

Методические рекомендации по самостоятельной  
подготовке студенческих команд к соревнованиям по  
спортивному программированию

Москва 2020

<b>Введение</b>	<b>8</b>
<b>1. Базовый глоссарий</b>	<b>9</b>
<b>2. Обзор основных индивидуальных и командных соревнований</b>	<b>14</b>
<b>3. Структура и география Чемпионата мира ICPC. Ключевое соревнование для студентов</b>	<b>25</b>
<b>3.1 Правила (проведения этапов) ICPC</b>	<b>26</b>
<b>3.2 Статистика ICPC</b>	<b>30</b>
<b>4. Теоретические и практические принципы обучения спортивному программированию</b>	<b>33</b>
<b>4.1. Индивидуальная подготовка</b>	<b>33</b>
○ 4.1.1. Изучение теории	34
○ 4.1.2. Практика	35
<b>4.2 Командная подготовка</b>	<b>41</b>
○ 4.2.1. Локальные тренировки	41
○ 4.2.2. Участие в сборах по спортивному программированию	41
<b>4.3 Техническая подготовка</b>	<b>43</b>
<b>4.4. Организационная подготовка при поездке на сборы</b>	<b>46</b>
<b>5. Развитие междисциплинарных гибких навыков</b>	<b>52</b>
<b>6. Приложения</b>	<b>59</b>
6.1. Пример спонсорского письма	59
6.2. Задачи для начального уровня (дивизион C)	61
6.3. Задачи для среднего уровня (B)	76
6.4. Задачи для продвинутого уровня (дивизиона A)	83
<b>7. Референсы</b>	<b>97</b>

## **Введение**

В этих методических рекомендациях вы найдете базовые рекомендации по самостоятельной командной подготовке к участию в соревнованиях по спортивному программированию.

Здесь дан базовый глоссарий, без которого не обойдется начинающий спортивный программист, сделан обзор основных индивидуальных и командных соревнований по спортивному программированию, а также приведено краткое, но емкое описание системы чемпионата мира по спортивному программированию для студентов (ICPC).

В документе приведены способы подготовки к ICPC и другим схожим по условиям чемпионатам, и также сформулированы главные теоретические и практические принципы освоения навыков спортивного программирования, кроме того, даны советы по организационной подготовке своего участия в соревнованиях и подготовительных сборах.

В качестве дополнения в рекомендации включен раздел по развитию гибких междисциплинарных навыков и примеры подготовительных задач.

Это пособие будет полезно студентам и старшим школьникам, интересующимся спортивным программированием, их родителям и начинающим тренерам.

# 1. Базовый глоссарий

## **Контекст**

Едиличное соревнование по спортивному программированию или основной тур соревнования; олимпиада.

Предполагает соревнование по программированию с двумя и более участниками (индивидуальными или командами) в рамках турнирной таблицы. Состоит в решении нескольких задач (проблем-сета, сета задач) в рамках определенных, заранее установленных правил. Как правило, победитель конкурса – это участник, правильно решивший максимальное количество задач за минимальное время.

## **Задача**

В математической формулировке, или сводимая к математической формулировке, проверяемая запуском на заданном наборе тестов ситуация, включающая в себя цель и условия, в которых она должна быть достигнута. Состоит из текста, примера набора входных данных и соответствующего набора выходных данных.

Основной целью является изобретение надежных программных решений для сложных, реально существующих проблем.

## **Попытка**

Отправка решения задачи на проверку. Каждая попытка принимается или отклоняется, и команда уведомляется о результате. Отклоненные попытки сопровождаются одной из пометок: run-time error; time-limit exceeded; wrong answer.

## **Тестирующая система**

Среда для запуска и оценки программных решений участников во время соревнований.

## **Дорешка**

Процесс самостоятельного решения всех заданий, которые не были решены во время контестов/практических занятий, важный пункт для дальнейшего профессионального развития.

## **Разборы**

Разборы бывают двух видов: устные и письменные

*Устный разбор* — это лекция тренера, где рассказывается о том, как нужно решать задачи, которые давались во время тура, например, основные алгоритмы, детали реализации и т.д.

*Письменный разбор* — это описание алгоритмов, которые автор задач применял для их решения.

Обычно устный разбор подробнее письменного. Во время тренировочных сборов в основном применяются оба вида разборов.

## **ICPC (The International Collegiate Programming Contest)**

Самый престижный международный студенческий чемпионат по спортивному программированию, в котором могут принимать участие команды, представляющие свое высшее учебное заведение, и состоящие из трех студентов или аспирантов в возрасте не старше 24 лет.

## **Дивизионы**

Участники (команды) условно распределяются по трем (иногда четырем) дивизионам: А, В, С, D.

Дивизион А (по рейтингу Codeforces — «красные») — рассчитан на самых опытных участников с многолетним опытом решения задач, тренирующихся ради участия в финале ICPC. Во время сборов, например, в этом дивизионе каждый день проходят пятичасовые контесты, разборы правильного решения задач и дорешки.

Дивизион В (по рейтингу Codeforces — «оранжевые» и верхняя когорта «фиолетовых») — для чуть менее опытных участников, готовящихся к

соревнованиям полуфинального уровня ICPC. В этом дивизионе во время сборов проводятся тематические конкурсы и лекции по сложным алгоритмам.

Дивизион C (по рейтингу Codeforces — «фиолетовые», «синие», «зеленые») — для начинающих. На этом уровне достаточно базовых знаний и опыта в программировании, занятия проводятся по более простым алгоритмам.

Дивизион D (по рейтингу Codeforces — «серые») — для совсем новичков, у которых нет команды. Часто на сборах участники из дивизиона D объединяются в команды и участвуют в дивизионе C.

## **Сборы**

Сборы — это серия интенсивных учебно-тренировочных мероприятий по спортивному программированию, на которых во время конкурсов имитируются условия ICPC, а также читаются лекции и делаются разборы задач. Обычно сборы длятся 1–2 недели в зависимости от программы и организатора. Сборов проводятся в разных местах России, стран СНГ и мира, так что это не только возможность отточить свои навыки по решению задач в команде, но также хороший шанс расширить горизонты, пообщаться с коллегами из других университетов и посмотреть новые города и страны.

Конкурс во время сборов — это основное 5-часовое мероприятие учебно-тренировочного дня, на котором команды решают задачи и соревнуются в рамках турнирной таблицы. В дивизионах A, B и C конкурсы — это командные соревнования, в дивизионе D — индивидуальный зачет.

Во время конкурсов тренеры сборов обычно находятся в аудитории и могут консультировать участников по их запросу, но не подсказывать решение. Тренеры команд на время конкурса удаляются в тренерскую, чтобы соревнование участников было честным.

## **Лекции**

Обычно читаются для всех дивизионов, кроме A. Лекции обычно длятся 2 академических часа, если иное не предусмотрено расписанием.

В особых случаях (например, при необходимости разобрать новый алгоритм) лекция может быть прочитана и для А-дивизиона.

### **Принимающая сторона (хост)**

Обычно вуз, индустриальная компания или общественная организация, занимающаяся размещением мероприятия (сборов, чемпионата) на заранее выбранной оргкомитетом локации.

### **Оргкомитет сборов**

В оргкомитет обычно входят исполнительный директор, руководитель сборов, ключевые сотрудники и координаторы сборов, технические специалисты, а также представители принимающей стороны.

### **Программный комитет сборов**

В Программный комитет входят главный методист, главный судья, приглашенные судьи и тренеры (преподаватели) сборов.

### **Тренер**

В рамках сборов существуют две категории тренеров. Первая — тренеры-преподаватели, которые читают программу участникам, то есть ведут лекции и разборы; вторая — тренеры команд-участниц. В каждом из двух случаев тренер — это специалист, руководящий подготовкой команд. С командой может работать несколько тренеров, на сборах тренером может быть член команды. Согласно правилам чемпионата ICPC участник команды не может выступать в качестве тренера, при этом заявленный тренер всегда должен быть аффилирован с вузом, который представляет команда.

### **Языки**

Официальными языками программирования Финала ICPC являются C/C++, Kotlin, Java и Python. На сборах по заявлению организаторов этот список может быть дополнен или сокращен. Чтобы новички могли получить пользу от участия в сборах и других видах обучения спортивному

программированию, им нужно знать минимум один из перечисленных выше языков, и иметь минимальную алгоритмическую подготовку.

C++ – оптимальный язык для участия в сборах. Python подходит для младших дивизионов, но его недостаточно для дальнейшего прогресса. Преподавателями рекомендуется иметь в базе хотя бы 2 языка, но эта рекомендация не является критерием участия в сборах.

### **Ключевые персоналии**

*Билл Паучер* — исполнительный директор ICPC, главная фигура Финала чемпионата.

*Алексей Малеев* — проректор по международным программам и технологическому предпринимательству Московского физико-технического института (МФТИ), основатель проекта тренировочных сборов Moscow Workshops и Discover.

*Владимир Парфенов* — декан факультета информационных технологий и программирования ИТМО, директор контестов Северо–Евразийского региона.

*Андрей Станкевич* — дважды призер ICPC, тренер 7 команд-чемпионов ICPC от ИТМО, замдиректора контестов и менеджер по регистрации Северо–Евразийского региона, к.т.н., доцент факультета информационных технологий и программирования, тренер сборов Moscow Workshops и Discover,

*Андрей Лопатин* — дважды чемпион ICPC, тренер 2 команд-чемпионов ICPC от СПбГУ, владелец награды Senior Coach Award, присуждаемой тренерам, чьи команды в течение 15 лет доходят до финала ICPC; преподаватель СПбГУ, тренер сборов Discover и Moscow Workshops.

*Олег Христенко (Снарк)*: программист, главный судья и модератор сборов Discover и Moscow Workshops, автор задач для контестов.



## **2. Обзор основных индивидуальных и командных соревнований**

Высокотехнологичные индустрии, государство, и, как следствие, образовательная система, все больше заинтересованы в формировании высококвалифицированных IT-кадров с перспективой на годы вперед. Так как спортивное программирование является одной из сфер, где активно формируются эти кадры, среда спортивного программирования очень активно развивается, ежегодно появляется все больше соревнований и площадок для их проведения: крупные компании сотрудничают с образовательными проектами и формируют свои собственные площадки для проведения конкурсов с целью развития своих сотрудников, привлечения новых, популяризации компании и увеличения сферы ее влияния.

Площадок — как очных, так и онлайн — существует огромное количество. Ниже мы разберем самые популярные в России и мире соревнования, которые проводятся не только для студентов, но и для школьников. Некоторые из них открыты для всех желающих.

### **Topcoder Open**

Доступ к ресурсу: [www.topcoder.com](http://www.topcoder.com)

Ежегодный индивидуальный профессиональный турнир по программированию, проводимый компанией Topcoder Inc.

Проводится с 2001 года, с 2003 года носит название Topcoder Open. Включает в себя четыре вида соревнований: Algorithm, Design, Development, Marathon Matches. В отборочных соревнованиях может принять участие любой человек, достигший возраста 18 лет. Ежегодно в турнире участвуют около 4000 программистов.

## Google Code Jam

Доступ к ресурсу: [codingcompetitions.withgoogle.com/codejam](https://codingcompetitions.withgoogle.com/codejam)

Международное соревнование по программированию, проводимое [Google](https://www.google.com). Стартовало в 2003 году с целью выявления лучших умов для последующей работы в Google. Соревнование состоит в решении определенного набора алгоритмических задач за фиксированное время. В отличие от большинства соревнований по программированию, участники здесь могут использовать любой [язык программирования](#) и [среду разработки](#).

Google Code Jam считается одним из самых массовых чемпионатов по программированию, так как участвовать может любой желающий. Так, в 2014 году было зарегистрировано почти 50 тысяч участников, среди которых больше половины прошли квалификационный раунд.

## ICPC

Доступ к ресурсу: [icpc.baylor.edu](https://icpc.baylor.edu)

Студенческий командный чемпионат мира по программированию ICPC (до 2017 — ACM ICPC). Считается самым престижным соревнованием по спортивному программированию.

ICPC вырос из турнира, который проводился в Техасском университете в 1970. Свою нынешнюю форму чемпионат принял в 1977 году, когда первый финал был проведен в рамках ежегодной конференции ACM по информатике. С тех пор турнир проводится ежегодно.

Начиная с 1989 года организацией соревнований занимается университет Бэйлора. В разное время спонсорами соревнований становились компании Apple, AT&T, Microsoft, с 1997 по 2017 год генеральным спонсором была компания IBM, начиная с 2018 года — компания JetBrains.

К настоящему моменту олимпиада превратилась во всемирное соревнование: в 2019 году в ней приняли участие 52 707 участников из 3 233

учебных заведений, представляющих 110 стран. 135 команд сошлись в борьбе за победу в финальном турнире.

### **Facebook Hacker Cup**

Доступ к ресурсу: [www.facebook.com/hackercup](http://www.facebook.com/hackercup)

Международное соревнование по программированию, проводимое Facebook. Было запущено в 2011 году, как средство по выявлению лучших умов для возможной работы в Facebook. Аналогично Google Code Jam, соревнование состоит из набора алгоритмических задач, которые должны быть решены за определенное время. Участники могут использовать любой язык программирования и среду разработки.

### **VK CUP**

Доступ к ресурсу: [vk.com/vkcup](http://vk.com/vkcup)

Ежегодный чемпионат по программированию, организованный ВКонтакте совместно с платформой Codeforces.

Раньше чемпионат проводился в формате командных соревнований. В 2020 году соревнование будет индивидуальным, призовой фонд составит около 688 000 рублей.

Сам конкурс состоит из четырех треков: ML, Engine, Mobile и Design. Каждый трек состоит из трех последовательных раундов: квалификационного, отборочного и финального. В каждом раунде участникам для решения дается одна или несколько задач.

Все этапы чемпионата суммарно по времени длятся около 8–9 месяцев, соревнования начинаются в сентябре.

В отборочный раунд каждого трека проходят участники, набравшие больше 0 баллов и не меньше того количества баллов, которое набрал участник,

находящийся на 256-м месте в квалификационном раунде соответствующего трека.

В финальный раунд каждого трека проходят 40 лучших участников. По результатам отборочного раунда организаторы могут пригласить в финал дополнительных участников.

## Codeforces

Доступ к ресурсу: [codeforces.com](https://codeforces.com)

Самая популярная в мире платформа для проведения соревнований по алгоритмике, существует с 2009 года. Поддерживается группой программистов из Университета ИТМО во главе с Михаилом Мирзаяновым. С 2013 года Codeforces опережает Topcoder по количеству активных пользователей.

Статистика 2019 года показывает ощутимый прирост платформы по всем основным метрикам (от 15% до 45%).

Страны-лидеры по количеству пользователей платформы: Индия, Китай, Россия, Бангладеш, Египет, США, Вьетнам, Япония, Бразилия, Казахстан.

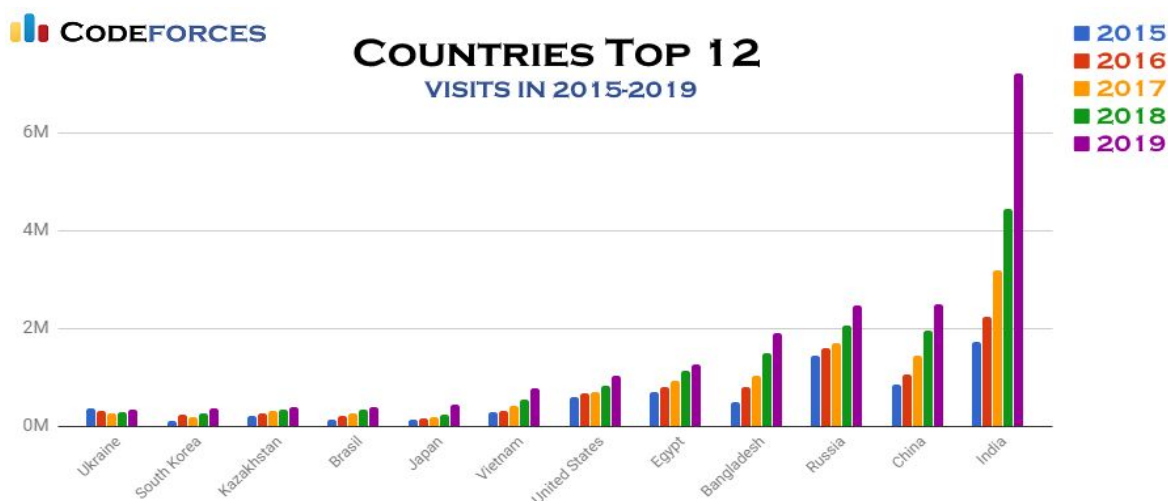


Рисунок 1. Статистика активности посетителей из разных стран.

Источник: <https://codeforces.com/blog/entry/73683?locale=ru>

У Codeforces много партнеров, таких как Telegram, Microsoft, Mail.Ru,

VK, JetBrains, Huawei, так что платформа регулярно проводит соревнования, организованные совместно с ними или для них.

Помимо крупных контестов, примерно раз в 1-2 недели на Codeforces проводятся свои «раунды», на которых участникам дается 5 задач на два часа.

Призы за победу в «раунде» не предусмотрены, но у каждого участника есть возможность повысить свой рейтинг, чтобы перейти в другой дивизион и соревноваться с более сильными участниками. Кроме того, попадание в высокорейтинговую когорту приносит узнаваемость, популярность, уважение со стороны IT-сообщества и заинтересованность крупных компаний-работодателей. Индустрия отслеживает показатели рейтинга участников Codeforces и при собеседовании/отборе на стажировку многих просят указать свой рейтинг на этой платформе, т.к. он является важным индексом подготовленности и уровня знаний кандидата.

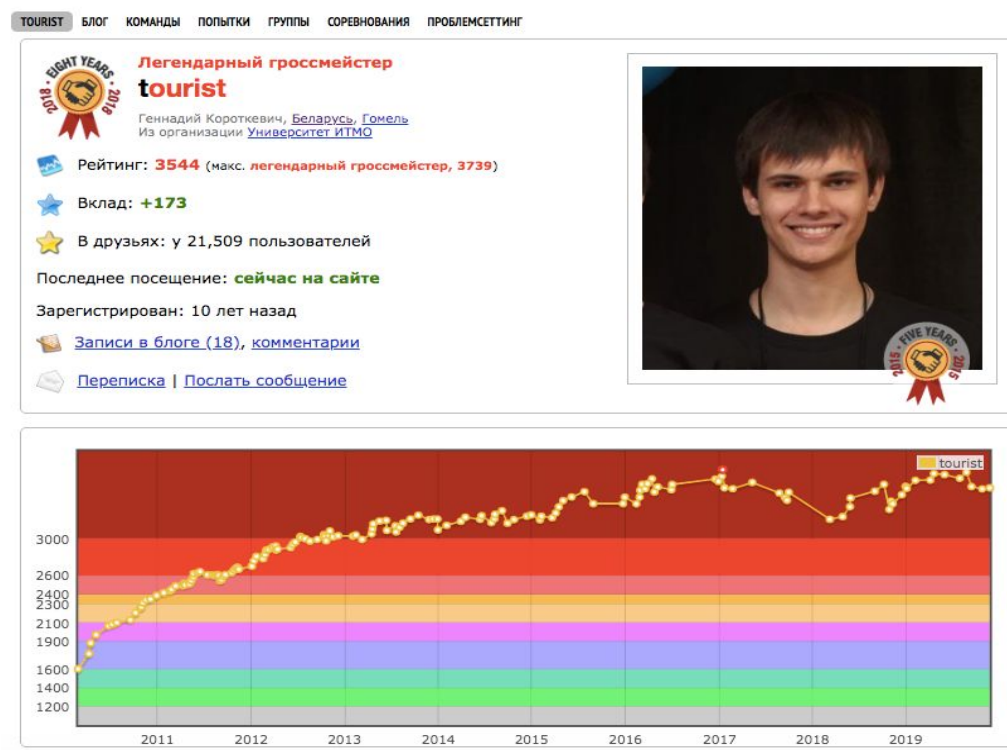


Рисунок 2. Скриншот личного профиля участника на платформе, отражающего рейтинг и прогресс участника.

Источник: <https://codeforces.com/profile/tourist>

## **AtCoder**

Доступ к ресурсу: [atcoder.jp](https://atcoder.jp)

Онлайн площадка для индивидуальных соревнований как для начинающих, так и для экспертов. Контесты проводятся раз в неделю. Задания здесь по сравнению с другими платформами некоторым участникам могут показаться более сложными из-за своего происхождения: задачи, созданные в традициях японской школы спортивного программирования, как правило, более сложные, чем в среднем по миру.

*Типы контестов, проводимых на площадке:*

AtCoder Beginner Contest (ABC) – задачи, предназначенные для новичков в спортивном программировании.

AtCoder Regular Contest (ARC) – некоторые задания похожи по уровню сложности на задания Grand Contest, однако решить их реально при наличии достаточного опыта.

AtCoder Grand Contest (AGC) – очный, финальный этап соревнований с самыми сложными заданиями, в который проходит только 8 человек из всех участников. Считается очень престижным соревнованием, т.к. туда очень трудно попасть.

## **CodeChef**

Доступ к ресурсу: [www.codechef.com](https://www.codechef.com)

Некоммерческая образовательная инициатива, созданная в 2009 году индийской компанией Directi, специализирующейся на разработке программного обеспечения. Это глобальное сообщество программистов, построенное на базе соревновательной платформы по спортивному программированию.

Ежемесячно на платформе проводятся онлайн–контесты на 35 языках программирования. Победители получают сладкие призы и приглашение на ежегодный CodeChef Cup. Кроме того, платформа, также как и Codeforces, открыта для партнерства, что позволяет крупным учреждениям и организациям по всему миру использовать CodeChef для проведения своих соревнований, получая поддержку со стороны платформы. Для образовательных учреждений сервис предоставляется бесплатно.

Проект также сотрудничает с образовательными учреждениями для создания локальных филиалов, ориентационных сессий и клубов по программированию.

## **IOI**

международная олимпиада по информатике среди школьников.

Доступ к ресурсу: [ioinformatics.org](http://ioinformatics.org)

Ежегодное соревнование по информатике среди школьников из более чем 80 стран мира. Впервые IOI была проведена в 1989 году.

Каждый год соревнование проходит в разных странах, 2 дня в летний или осенний сезон. На IOI участникам предлагается решить и запрограммировать алгоритмические задачи. По правилам в команде должно быть не более четырех участников. В сборную страны участники обычно отбираются по результатам национальных соревнований. В России команда формируется по итогам Всероссийской олимпиады по информатике и учебно–тренировочных сборов.

Состязание проводится в два тура. Итоги подводятся в индивидуальном зачете. При подведении итогов не более 50% участников награждаются медалями, так что соотношение золота, серебра, бронзы и отсутствия медалей было приблизительно 1:2:3:6 (то есть 1/12 часть участников получает золотые медали).

## **ВКОШП**

Всероссийская командная олимпиада школьников по программированию.

Доступ к ресурсу: [neerc.ifmo.ru/school/russia-team](http://neerc.ifmo.ru/school/russia-team)

Всероссийская очная командная олимпиада школьников по программированию, традиционно организуется осенью в Санкт–Петербурге и Барнауле. К участию также приглашаются школьники из других стран, площадки для которых организуются отдельно.

Олимпиада проводится в один компьютерный тур. На 5-часовом туре командам в составе из 3-х человек предоставляется компьютер и предлагается решить несколько задач. По итогам тура лучшие 15 команд награждаются медалями (4 золотых, 5 серебряных и 6 бронзовых). Остальные команды получают дипломы.

## **Технокубок**

Доступ к ресурсу: [technocup.mail.ru](http://technocup.mail.ru)

Ежегодная олимпиада по программированию, проводимая с 2015/2016 учебного года для учащихся 8–11 классов. Организаторы: МФТИ, МГТУ им. Н.Э. Баумана и компания Mail.Ru Group. Олимпиада проводится в два этапа (отборочный онлайн и финальный очный).

Олимпиада входит в перечень РСОШ и имеет I уровень, что позволяет победителям и призерам поступать в вуз без экзаменов, или получить 100 баллов ЕГЭ по информатике. Кроме того, победители получают ценные призы, а также привилегии при поступлении на образовательные проекты Mail.Ru Group.



## **OpenCup**

Доступ к ресурсу: [opencup.ru](http://opencup.ru)

Открытый Кубок им. Е.В. Панкратьева по программированию. Сезон соревнований начинается в сентябре и заканчивается в мае следующего года.

Участвовать в соревнованиях турнира могут участники трех категорий:

- SCHOOL (школьники) – учащиеся средних и средних специальных учебных заведений, и их аналогов в других странах;
- ACM – студенты высших учебных заведений, имеющие право выступать в Чемпионатах Мира ICPC в сезоне, начавшемся в текущем году;
- OPEN – все остальные.

Сезон OpenCup — это серия этапов, проводимых в соответствии с правилами, приближенными к ACM, называемых также Гран-при. Количество этапов каждый год может меняться.

Этапы проводятся в двух дивизионах: первом (основном) и втором (учебном). Разница дивизионов заключается в уровне сложности задач. Задачи первого дивизиона труднее задач второго дивизиона. Участники могут сами выбрать, задания какого дивизиона им решать.

## **Moscode Festival**

Доступ к ресурсу: [it-edu.com/pages/moscode2019](http://it-edu.com/pages/moscode2019)

Двухдневный студенческий чемпионат по спортивному программированию, объединяющий студентов и выпускников более чем из 20 стран мира. Организатором чемпионата выступает Центр развития ИТ-образования МФТИ. Обычно проводится в преддверии финала Чемпионата мира ICPC. Это возможность для студентов из России и зарубежных стран, чемпионов прошлых лет, а также молодых ИТ-специалистов, попробовать свои силы на соревновании высокого уровня в борьбе за кубок.

Участие в фестивале бесплатное, обучение и соревновательная часть проходят на английском языке. Событие проводится при поддержке крупных компаний–партнеров из индустрии таких как Мегафон, Acronis, Huawei, Яндекс и других.

В рамках соревнований команды в составе трех человек пишут 5-часовой констест, состоящий из 12 алгоритмических задач по правилам Чемпионата мира ICPC.

## **Rucode Festival**

Доступ к ресурсу: <https://rucode.net/>

Всероссийская программа интенсивной подготовки по спортивному программированию и искусственному интеллекту, созданной в партнерстве с Фондом развития Физтех–школ и при поддержке Фонда президентских грантов.

В рамках программы все желающие смогут повысить свой уровень знаний по искусственному интеллекту и спортивному программированию и принять участие в своих первых соревнованиях.

Студенческие команды–победители регионального чемпионата RuCode Festival за счет организаторов приглашаются в Москву в качестве зрителей финала ICPC.

Фестиваль RuCode проходит в несколько этапов:

- *Онлайн-курс «Быстрый старт в спортивное программирование»:* массовый открытый онлайн-курс, бесплатный и доступный для всех желающих. 4 модуля курса помогают войти в мир спортивного программирования и разобрать темы от базовых линейных алгоритмов до основ теории графов и их обходов. Также рассматриваются вопросы асимптотического анализа и оценки эффективности программы. Ссылка на курс: <https://stepik.org/64454>

- *Отборы на очные интенсивы:*

для дальнейшего участия в очных интенсивах участники проходят отбор в онлайн-формате. Зарегистрироваться на отборы можно на сайте фестиваля.

- *Очные трехдневные интенсивы:*

занятия, которые проводят ведущие тренеры из проекта Moscow Workshops в 10 городах России: Владивосток, Екатеринбург, Ижевск, Иннополис, Иркутск, Красноярск, Новосибирск, Пермь, Саратов, Чита. На интенсивы можно попасть только по результатам отборов.

- *Чемпионат RuCode Festival:*

финальный этап соревнований по искусственному интеллекту и спортивному программированию между командами дивизионов A/B и C/D. Проводятся очно, в 11 городах России. Принять участие может любой желающий, необходима предварительная регистрация на сайте фестиваля.

По всему миру существуют десятки разных соревновательных площадок. Мы перечислили только самые популярные.

Это могут быть только онлайн платформы, только очные чемпионаты, а также смешанные, состоящие из заочного отборочного и очного финального этапа.

Статистика показывает, что сфера спортивного программирования становится популярнее год от года. Индустрия вовлекается в процесс развития данного направления и старается больше поддерживать молодых специалистов. Крупные компании стали создавать собственные соревновательные площадки с призовым фондом в финале. Такой подход обеспечивает компании сильный интерес со стороны участников и увеличивает ее популярность.

### **3. Структура и география Чемпионата мира ICPC. Ключевое соревнование для студентов**

Основная миссия Чемпионата ICPC — предоставить студентам возможность взаимодействия с единомышленниками из других университетов, отточить свои навыки программирования, а также продемонстрировать умение решать сложные задачи и работать в команде. В рамках Чемпионата создается среда, где формируется новое поколение профессионалов в области компьютерных технологий со всего мира, стремящихся постоянно улучшать свои навыки.

ICPC — это командное соревнование. Между собой соревнуются команды разных университетов. В состав каждой команды входит 3 человека. По правилам ICPC участник на момент начала Чемпионата должен быть студентом или аспирантом и быть не старше 24 лет. Для участия в чемпионате команде необходимо зарегистрироваться на сайте чемпионата. У каждой команды должен быть тренер — человек, являющийся контактным лицом в течение всего чемпионата. При этом тренером команды не может быть ее участник. На протяжении всех этапов чемпионата состав команды не может изменяться, запасные игроки не допускаются.

Традиционно чемпионат ICPC состоит из трех этапов — четвертьфинал (региональный этап), полуфинал (финал подрегиона) и финал. Организаторы региональных соревнований могут проводить дополнительные квалификационные соревнования (1/8 финала) для привлечения более широкого круга участников и дальнейшего отбора. Все этапы имеют свои правила и особенности проведения.

Каждый из этапов чемпионата представляет собой пятичасовой контест, по итогам которого определяются призеры: 4 золотых, 4 серебряных и 4 бронзовых команды-медалиста. Иногда по решению организаторов может быть

присвоена дополнительная бронзовая медаль, при этом команде присваивается двенадцатое место. Команда с наилучшим результатом по итогам Финала Чемпионата становится Чемпионом Мира, награждается Кубком Чемпиона Мира и именной табличкой.

В ходе конкурса перед командами стоит цель решить максимально возможное количество из предложенного сета задач. Количество задач варьируется от года к году, обычно их от 8 до 13.

### **3.1 Правила (проведения этапов) ICPC**

В рамках чемпионата существует разделение на 8 подрегионов, которые включают в себя страны со всего мира, основываясь на территориальной принадлежности.

Для того, чтобы попасть в финал ICPC команды из разных стран проходят отборочные этапы в рамках чемпионата своего региона, по результатам которого лучшие участники попадают на финальный этап ICPC, где собираются команды со всего мира, прошедшие все ступени отбора. Учитывая географический фактор, в данной методичке будут рассмотрены правила проведения этапов чемпионата Северо-Евразийского региона (NEERC).

На чемпионат Северной Евразии отбор команд происходит посредством региональных соревнований. Северо-Евразийский подрегион делится на 16 регионов:

1. Восточно–Сибирский
2. Грузинский
3. Белорусский и Балтийский
4. Дальневосточный
5. Уральский
6. Центрально–Российский
7. Таврический (Крым и Украина)

8. Московский (Москва и Московская область)
9. Северо-Западный
10. Кыргызстанский
11. Казахстанский
12. Волжский и Южный
13. Армянский
14. Западно–Сибирский
15. Азербайджанский
16. Узбекистанский (Узбекистан и Таджикистан)

После проведения региональных соревнований команды упорядочиваются по результатам выступления в регионе.

Каждому региону присваивается своя квота — количество команд, которые могут быть допущены к выступлению на Чемпионате Северной Евразии по результатам регионального соревнования. Чем больше команд от региона принимает участие в региональном и квалификационном этапах, тем больше команд пройдет на более высокие этапы чемпионата.

Квота присваивается и каждому университету — это максимальное количество команд от университета, которые могут быть допущены к выступлению на Чемпионате Северной Евразии по результатам регионального соревнования. Квоты выделяются по аналогии с квотой региона — чем больше команд от университета принимает участие в региональном и квалификационном этапах, тем больше команд пройдет на более высокие этапы чемпионата.

Организаторы региональных соревнований могут проводить квалификационные соревнования (1/8 финала) для привлечения более широкого круга участников и их отбора. Квалификационный этап проходит на площадках региональных университетов.

Правила проведения квалификационного этапа могут отличаться от регионального по решению вуза, организующего площадку проведения. Русский язык допускается в формулировке условий, также принимаются задачи, написанные на дополнительных языках программирования (например, Pascal).

В основном правила проведения чемпионата едины на всех этапах соревнования, но на квалификационном и региональных этапах организаторы могут внести в правила некоторые изменения. Ниже приведено описание всех этапов чемпионата.

Характеристики этапов:

#### **Квалификационный этап — 1/8 ICPC (Qualifications)**

Языки программирования: C/C++.

Могут быть приняты задачи, написанные на Java, Python, Kotlin.

Условия задач: на местном языке.

К участию допускаются все желающие, зарегистрировавшиеся на сайте организатора.

Место проведения: площадки региональных университетов.

Нет ограничений по количеству участий.

#### **Четвертьфинал ICPC (Regionals)**

Языки программирования: C/C++. Могут быть приняты задачи, написанные на Java, Python, Kotlin.

Условия задач: на английском языке. В зависимости от региона может быть использован местный язык.

К участию допускаются все желающие (прошедшие квалификационный этап в случае его проведения), зарегистрировавшиеся на сайте организатора.

Место проведения: площадки региональных университетов.

В четвертьфинале можно участвовать не более 5 раз.

### **Полуфинал ICPC, регион Северной Евразии (NERC):**

Языки программирования: C/C++, Java, Python, Kotlin.

Условия задач: на английском языке.

Количество команд от университета: определяется квотой университета (от 2 до 6 команд включительно).

Количество команд от региона: определяется квотой региона (минимальное количество команд — 2 от региона).

Место проведения: Санкт–Петербург, Тбилиси, Алматы, Барнаул.

В полуфинале команда имеет возможность участвовать не более 5 раз.

### **Финал ICPC (World Finals):**

Языки программирования: C/C++, Java, Python, Kotlin.

Условия задач даются на английском языке.

Количество команд от университета: 1.

Количество команд от региона: меняется каждый год.

Место проведения: определяется на Финале предыдущего года.

В Финале команда может участвовать не более 2 раз.



### 3.2 Статистика ICPC



Рисунок 3. География участников ICPC 2019. Регионы полуфинала.

Самый многочисленный регион Западной Азии.

Создано на основе данных из <https://icpc.baylor.edu/>

К началу 2019 года международное сообщество ICPC – это:



Рисунок 4. Международное сообщество ICPC в цифрах на 2019 год.

Создано на основе данных из <https://icpc.baylor.edu/>

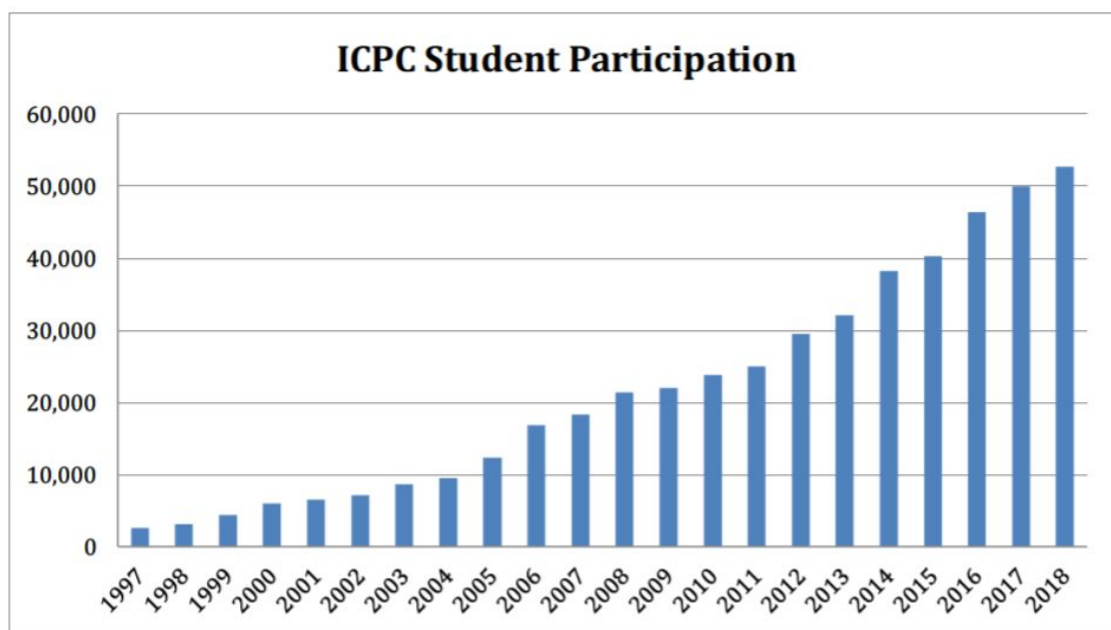


Рисунок 5. Статистика участников всех этапов ICPC в период 1997-2018 гг. Значительный ежегодный рост показателей участников.  
 Источник: <https://icpc.baylor.edu/worldfinals/pdf/Factsheet.pdf>

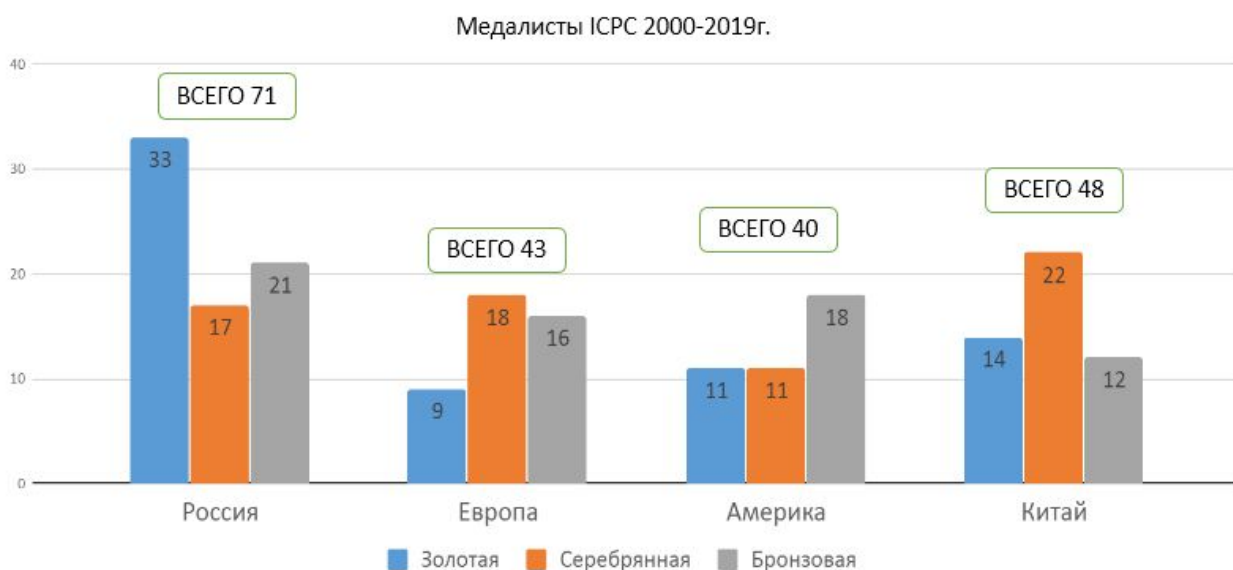


Рисунок 6. Медалисты Финала ICPC по странам, начиная с 2000 года. Россия стабильно удерживает лидерство, опережая Китай, США и страны Европы.

Источник: <https://icpc.baylor.edu/>

Резюмируем вышесказанное. Чемпионат ICPC состоит из трех этапов — четвертьфинал (региональный этап), полуфинал (финал субрегиона) и финал.

В рамках чемпионата между собой соревнуются команды разных университетов. В состав каждой команды может входить только 3 студента, возрастом не старше 24 лет.

Все этапы чемпионата представляет собой пятичасовой констест, состоящий обычно из 8-13 задач. В рамках награждения каждого этапа ICPC есть 12 призовых мест: 4 золота, 4 серебра, 4 бронзы. Лучшая команда получает титул чемпиона и кубок.

В рамках отборочных этапов всего чемпионата существуют квоты региона и квоты университетов. Чем больше команд от региона принимает участие в региональном и квалификационном этапах, тем больше команд пройдет на более высокие этапы чемпионата. Та же самая логика применима и для квот университетов.

Правила проведения квалификационного этапа могут отличаться от регионального по решению организаторов. Допускается использование другого языка в формулировках условий, а также написание программ с использованием языков из расширенного списка.

С каждым годом растет количество участников чемпионата. Российские команды с 2000 года держат лидерство в чемпионате среди более 100 стран-участниц. Чемпионат обещает стать еще более статусным. Крупные IT-компании со всего мира уделяют пристальное внимание подобным чемпионатам, рассматривая их как инструмент для поиска специалистов с высоким потенциалом.

## **4. Теоретические и практические принципы обучения спортивному программированию**

Прежде чем прийти в спортивное программирование, у вас должна быть первоначальная база, на которой будут строиться дальнейшее обучение и тренировки, а именно:

- Нужно знать язык программирования из списка: C, C++, Java. Даже знание языка Python будет достаточно для решения самых базовых задач, но следует учитывать, что этот язык медленнее, чем остальные;
- Необходимы базовая математическая подготовка и умение решать нестандартные задачи;
- Умение строить математическую модель задач;
- Опыт алгоритмического программирования;
- Знание английского для разных целей:
  - технического английского на уровне, достаточном для понимания задач и читаемых на сборах лекций. На начальном этапе в этом хорошо помогает работа со словарем;
  - разговорного английского для комфортного общения с иностранцами онлайн и офлайн.

Только с этим базисом рационально переходить к дальнейшей индивидуальной и последующей командной подготовкам.

### **4.1. Индивидуальная подготовка**

Для достижения успеха в спортивном программировании тренироваться необходимо, в первую очередь, самостоятельно. Даже при подготовке к командным соревнованиям базис участников закладывается только личными тренировками. На турнирах показывается только результат обучения — то, как соревнующиеся усвоили ранее полученный материал. Вся основная работа проводится вне рамок турниров. Только ваша личная прокачка при правильном

подходе превращает вас в сильное звено команды. Именно поэтому все, в том числе и очень титулованные программисты, большое количество времени уделяют самостоятельным тренировкам.

Такая подготовка включает в себя обязательную теоретическую и практическую части.

#### **4.1.1. Изучение теории**

Первый шаг в изучении теории — книги и обучающие статьи по теме. Наилучшим вариантом для старта является книга «Олимпиадное программирование» (автор Антти Лааксонен, издательство ДМК–Пресс, 2018г.).

Эта книга — идеальное справочное пособие для начинающих. В ней подробно описывается как проходят олимпиады, что требуется от участника, в чем цель соревнований и как к ним готовиться. Подробно разобраны базовые темы, трюки и алгоритмы. В ней также приведено много приемов проектирования алгоритмов, которые известны опытным олимпиадникам, но до сих пор обсуждались лишь на сетевых форумах и в блогах.

Существуют десятки книг по изучению спортивного программирования, однако именно эта книга на данный момент является самой актуальной и охватывающей широкий круг тем, поэтому она может быть полезна не только новичкам, но и опытным участникам.

Из онлайн источников одним из базовых является портал E-maxx. Доступ к ресурсу возможен по [ссылке](#). Его разработчиком является Иванов Максим под ником e-maxx aka maxdiver.

На портале можно найти много полезных материалов абсолютно бесплатно, включая электронные книги по программированию, список алгоритмов с описанием к ним и программ. Помимо этого на сайте есть форум, где участники обсуждают проблемы, алгоритмы, книги, новости из мира ICPC.



Рисунок 7. Интерфейс портала [e-maxx.ru](http://e-maxx.ru)

Для лучшего понимания теории алгоритмов мы рекомендуем пользоваться этим ресурсом, дополнительно находя в интернете статьи с пояснениями и авторскими решениями по каждому из алгоритмов. Точно полезными будут самостоятельные разборы изученных алгоритмов.

Следующий этап — переход к практике решения задач и разбора алгоритмов на онлайн-платформах.

#### 4.1.2. Практика

Следующий этап для эффективного усвоения полученной теоретической информации — ее применение на практике. Стратегия простая, но трудоемкая — решение как можно большего числа задач по изученным темам. Для поиска готовых задач существует много порталов с архивами констестов. Некоторые из них будут описаны ниже.

### Timus Online Judge

Доступ к ресурсу: [acm.timus.ru](http://acm.timus.ru)

Данный портал содержит в себе крупнейший архив задач по программированию разного уровня сложности. На сайте также имеется автоматическая проверяющая система, позволяющая мгновенно получить обратную связь по своему решению.

Задачи для архива берутся с соревнований Уральского федерального университета, Чемпионатов Урала, Уральских четвертьфиналов ICPC, Петрозаводских сборов по программированию. Портал предоставляет возможность участвовать в онлайн-турах, которые регулярно проходят в Уральском федеральном университете.

Понять уровень сложности задачи можно по проценту решивших ее на платформе. Если этот процент большой – задачи легкие. В целом сервис работает как «большая дорешка», что позволяет участникам тренироваться вне конкурсов и сборов.

Изначально рекомендуем решать задачи без учета времени. Это может длиться день или даже два, главное, дойти до результата. Следующий шаг — решение на скорость с учетом времени. Дальше уже необходимо переходить тренироваться на другие площадки. Кроме данного сервиса существует множество других площадок, которые позволяют не только найти архивы с задачами, но и участвовать в соревнованиях в личном зачете. Более подробно о них мы расскажем далее.

Личные соревнования — эффективный способ прокачки навыков по решению задач в рамках ограниченного времени и возможность продемонстрировать уровень своих знаний на широкую аудиторию. Как правило, на платформах такого типа существует рейтинговая система и деление на дивизионы, что дает возможность опытным не соревноваться с новичками, и наоборот.

## **Codeforces**

Доступ к ресурсу: [codeforces.com](https://codeforces.com)

Как уже было сказано ранее CodeForces считается самой популярной и активной среди других платформ. На ней очень часто проводятся конкурсы, реализованные совместно с партнерами или для них.

Кроме крупных конкурсов, на платформе часто проводятся «раунды», для решения которых участникам даются 5 задач на два часа. Проводятся 1-2 раза в неделю. Призов здесь не дают, но у каждого есть возможность повысить свой рейтинг, что дает свои преимущества, которые были описаны в разделе ранее.

В течение двух дней после каждого раунда публикуются краткие разборы задач. Рекомендуем вам их читать. Кроме того, после окончания раунда у вас есть возможность «дорешивать» задачи — отправлять их на проверку, просто чтобы дописать или исправить ошибки, которые у вас были. На результаты раунда это, конечно, уже не влияет, но позволяет разобраться в своих ошибках.

Кроме того, на платформе есть много другой полезной информации. Во-первых, там внедрен функционал блогов, и пользователи часто пишут различные тексты и статьи на темы, связанные с программированием. Во-вторых, на сайте есть раздел "тренировки", куда выкладываются задачи многих прошедших олимпиад и их можно использовать для самостоятельной подготовки.

## **TopCoder**

Доступ к ресурсу: [www.topcoder.com](https://www.topcoder.com)

Американская платформа, немного уступающая CodeForces по популярности, но не менее полезная для тренировок.

На платформе проводится 4 вида соревнований, среди которых наиболее популярное – турнир по быстрому решению задач на алгоритмы, где приз за 1 место составляет \$15,000 USD.



В финальных соревнованиях принимают участие до 72 человек: 48 в самом популярном Algorithm Competition (приз за 1 место = \$15 000) и по 8 человек в остальных.

В рамках турнира каждому участнику дается 3 задачи, разные по сложности, классифицируемые на 3 уровня. Каждая задача имеет свою максимальную стоимость в баллах. Баллы начисляются только за решения, признанные верными, частичные решения не учитываются.

Перед началом соревнования участников распределяют по виртуальным комнатам (до 20 человек). Такие матчи, называемые SRM (Single Round Match), проходят примерно раз в две недели. Кроме этого проводятся ежегодные турниры. Сам матч состоит из трех основных фаз: Coding, Challenging и System Testing.

## **SPOJ**

Доступ к ресурсу: [www.spoj.com](http://www.spoj.com)

Крупный англоязычный сайт с более чем 20000 задач на абсолютно разные темы: динамическое программирование, графы, структуры данных и т.д. Изредка проводятся контесты, которые представляют интерес, если вы живете в стране проведения.

## **Сервис дистанционной подготовки к информатике**

Доступ к ресурсу: [informatics.mccme.ru](http://informatics.mccme.ru)

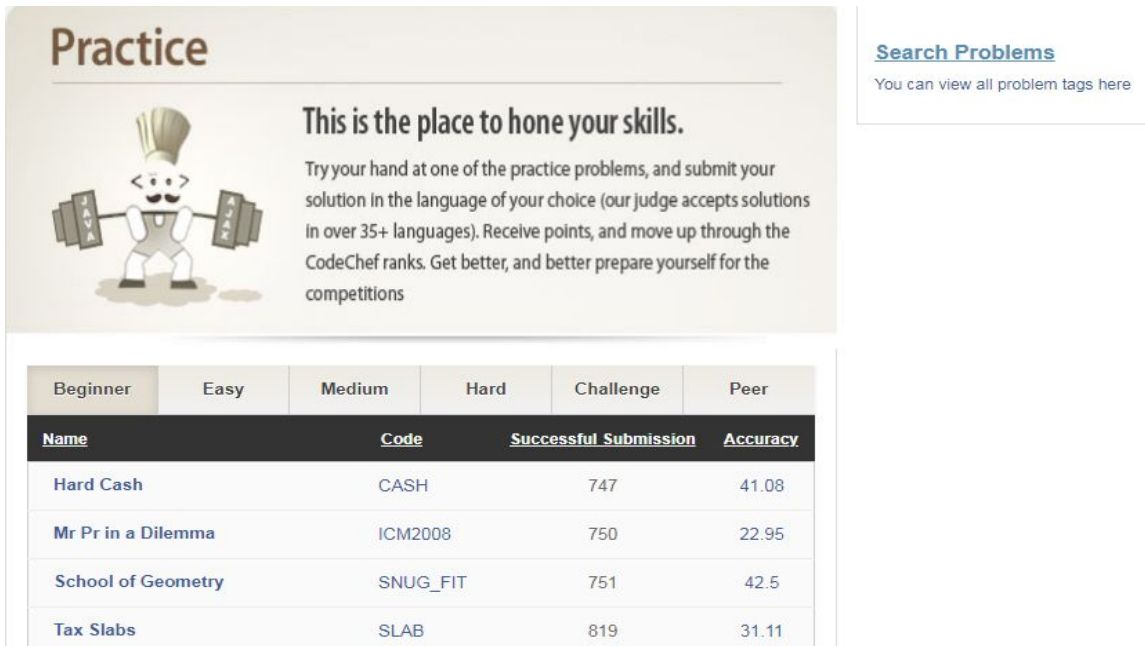
Платформа с большим количеством теоретических материалов и задач по соответствующим темам. Все очень удобно собрано по категориям и темам. Содержит большую базу задач с прошедших олимпиад школьников.

## **CodeChef**

Доступ к ресурсу: [www.codechef.com](http://www.codechef.com)

Менее крупный аналог Codeforces и TopCoder с огромным архивом задач разного уровня сложности и регулярными контестами.

Ежемесячно на платформе проводится три соревнования, победители которых получают сладкие призы от платформы. Также платформа предоставляет условия для организации турниров на базе своей платформы представителями образовательных учреждений и индустрии. В среднем на CodeChef ежемесячно проводится более 30 внешних контестов.



**Practice**

**This is the place to hone your skills.**

Try your hand at one of the practice problems, and submit your solution in the language of your choice (our judge accepts solutions in over 35+ languages). Receive points, and move up through the CodeChef ranks. Get better, and better prepare yourself for the competitions

[Search Problems](#)  
You can view all problem tags here

Beginner	Easy	Medium	Hard	Challenge	Peer
Name	Code	Successful Submission	Accuracy		
Hard Cash	CASH	747	41.08		
Mr Pr in a Dilemma	ICM2008	750	22.95		
School of Geometry	SNUG_FIT	751	42.5		
Tax Slabs	SLAB	819	31.11		

Рисунок 9. Главная страница портала CodeChef

## CodinGame

Доступ к ресурсу: [www.codingame.com/start](http://www.codingame.com/start)

Сайт, на котором программирование и видеоигры сливаются в единое целое. Здесь можно найти большую коллекцию задач по программированию, оформленных в виде видеоигр, что делает процесс тренировок веселее. Также тут изредка (раз в два месяца) проводятся контесты, содержащие в себе задачи на оптимизацию и ИИ, победители которых получают ценные призы. А если вы решите много задач, то на вас могут обратить внимание компании из индустрии.

## C Puzzles

Доступ к ресурсу: [www.gowrikumar.com/c/index.php](http://www.gowrikumar.com/c/index.php)

Подборка головоломок, специфичных для языка C, со всеми его причудами. Например, дан код, который, по логике, не должен работать, но он компилируется и даже правильно выполняет свою задачу. Надо понять, почему так? На этом сайте вы сможете приобрести навык отладки программ и чтения кода других.

---

The following C program segfaults of IA-64, but works fine on IA-32.

```
int main()
{
    int* p;
    p = (int*)malloc(sizeof(int));
    *p = 10;
    return 0;
}
```

Why does it happen so?

---

Рисунок 10. Пример головоломки, взятой с портала.

Источник [www.gowrikumar.com](http://www.gowrikumar.com)

Чем больше вы самостоятельно тренируетесь и осваиваете новые алгоритмы, тем более сильным вы становитесь для своей команды. Существует множество площадок и сервисов, где можно тренироваться. У каждой платформы есть свои особенности, форматы, зарубежная аудитория, плюсы и минусы. Тем самым вы не ограничены, у вас есть выбор и каждый сможет найти то, что ему по душе. Скучно не будет.

## 4.2 Командная подготовка

Чемпионат ICPC — это прежде всего командное соревнование, поэтому тем, кто хочет занять лидирующие места, необходимо отладить работу в команде. Каждый из участников должен понимать свою роль в группе, знать

свои сильные и слабые стороны, а также навыки товарищей. Если с индивидуальными навыками все проще (можно прокачать свой скилл в любой момент), то умение работать в команде приходит только с опытом. Есть два основных вида групповых тренировок, которые помогут команде стать более сплоченной.

#### **4.2.1. Локальные тренировки**

Метод подразумевает тренировку команды вместе со своим тренером на площадке университета. У каждого тренера свой подход к обучению, а некоторые тренеры даже едут с командой на сборы, чтобы взглянуть на них со стороны в менее привычной обстановке и почерпнуть новые фишки, техники, методы, чтобы в дальнейшем адаптировать их на локальных тренировках.

#### **4.2.2. Участие в сборах по спортивному программированию**

Данный метод не только позволит улучшить работу в команде, ведь большинство сборов проводятся по правилам ICPC или приближенным к ним, но и отточить свои собственные навыки программирования на теоретических занятиях. Сборы — отличная возможность прорешать как можно больше новых задач, разобрать новую для себя тему, а также испытать на команде стрессовые ситуации, с которыми участники регулярно встречаются на соревнованиях.

Все сборы предполагают определенный уровень подготовки команды и имеют большую географию проведения: команда может испытать силы в ежегодных сборах в Петрозаводске и Ижевске, принять участие в Чемпионате Урала или BSUIR Open, посетить сборы, организованные Moscow Workshops, которые проходят не только в российских городах (Москва, Иркутск, Владивосток), но и за рубежом — в Кыргызстане, Беларуси, Сингапуре, Латвии и других странах, что позволит дополнительно получить практику английского

языка. А самое ценное в сборах Moscow Workshops (MWs) и Discover — это возможность проконсультироваться у лучших тренеров, которые готовят чемпионов и медалистов финала ICPC, а также пообщаться с участниками из других регионов и стран. Программа сборов может включать теоретические занятия, а может состоять только из контестов. Команда всегда может выбрать для себя наиболее подходящую для себя программу и удобное место проведения.

### **4.3 Техническая подготовка**

Для того, чтобы иметь возможность полноценно решать задачи и работать с системой в рамках контестов или тренировок, необходима техническая подготовка рабочего компьютера. В зависимости от организатора и места проведения мероприятия существуют разные требования к настройке аппаратуры.

На очных турнирах начиная с ¼ этапа ICPC организаторы в обязательном порядке предоставляют компьютеры всем участникам, предварительно установив все необходимое программное обеспечение — это требование оргкомитета ICPC.

Если речь идет о локальных тренировках или сборах, на которые участники чаще всего приносят свои ноутбуки, вопрос самостоятельной настройки становится актуальным.

В рамках локальных тренировок в университете информацию по настройке аппаратуры можно получить у своего тренера и/или провести настройку самостоятельно.

При участии в тренировочных сборах требования к настройке аппаратуры различаются в зависимости от площадки и мероприятия. У каждого организатора площадки есть свои правила по настройке, обусловленные наличием своей тестирующей системы и поддерживаемыми ею языками.

С требованиями по подготовке к сборам вы сможете ознакомиться заранее, т.к. организаторы обычно заблаговременно размещают данную информацию на своем сайте, чтобы участник мероприятия был сразу готов приступить к пробному туру и решению задач.

Мы составили ряд рекомендаций и список необходимого программного обеспечения, которое нужно установить, чтобы быть готовым заранее и сконцентрироваться на обучении. Данные рекомендации также подойдут для организации личных тренировок.

Рекомендуемое программное обеспечение для подготовки ноутбука:

- Компилятор с поддержкой того языка, который будет на сборах.
  - В случае если вы программируете на C++, то вам необходимо установить компилятор GCC v7.3 и выше;
  - если на Java, то GCC v8.0 и выше;
- Code::Blocks — свободная кроссплатформенная среда разработки. Поддерживает языки программирования C, C++, D, Fortran;
- Geany — среда разработки программного обеспечения, написанная с использованием библиотеки GTK+. Предназначена для операционных систем: BSD, Linux, Mac OS X, Solaris и Windows. При этом, не включает в свой состав компилятор;
- CLion — интегрированная среда разработки для языков программирования Си и C++, разрабатываемая компанией JetBrains. Работает на операционных системах «Windows», «macOS», и «Linux». Для полноценной работы со средой необходимо решить вопрос с лицензией. Для университетов она бесплатная, поэтому вы можете обратиться к тренеру или иным ответственным лицам для решения этого вопроса;

- Microsoft Visual C++ — интегрированная среда разработки приложений на языке C++;
- Установленный браузер для взаимодействия с системой, т.к. практически все тестирующие системы взаимодействуют с участником через WEB-интерфейс.

Обычно на соревнованиях и сборах в большинстве случаев проводится пробный тур в начале мероприятия, что дает возможность познакомиться с тестирующей системой. **Важно!** Даже если вы уже работали в данной системе ранее, настоятельно рекомендуем вам пройти пробный тур, т.к. любая система со временем обновляется и в нее могут быть внесены новые функции.

Несмотря на то, что у каждого вуза или площадки может быть своя тестирующая система, основной функционал везде одинаков и включает в себя:

- возможность отправки решения на распечатку;
- возможность отправки задачи на проверку;
- возможность увидеть результат проверки;
- возможность ознакомиться с условиями и примерами;
- возможность получить архив тестов

Пошаговая стратегия поведения на пробном туре:

- 1) проверить, что ваш пароль позволяет вам войти на предоставленную рабочую станцию и тестирующую систему;
- 2) проверить, что вы освоили основной функционал тестирующей системы, а именно:
  - вы можете отправлять задачи на проверку;
  - вы получаете адекватные отзывы системы на ваши решения;
  - вы имеете возможность получить все возможные вердикты ошибок, чтобы избежать их в основном туре. Как это сделать, вам расскажет жюри. В случае, если что-то пошло не так — сообщайте об этом.

Самое главное: не надо скромничать, потому что пробный тур как раз и предназначен для выявления неполадок и знакомства с системой.

3) Проверив все вердикты, можно проверить дополнительные возможности тестирующей системы:

- онлайн-доступ к условиям (к которым он не закрыт по соображениям безопасности);
- возможность доступа к тестам, которые даны в качестве примеров;
- узнать каким образом отправляется на распечатку 1–2 из своих решений, чтобы убедиться, что процесс распечатки налажен: вам ее приносят или вы знаете, где ее забирать.

4) Если компилятор отличается от того, который вы используете между пробным и основным туром, изучите основы компилятора. Попробуйте послать код, который будет компилироваться неправильно, чтобы знать чего ожидать и к чему быть готовым.

Добиться того, чтобы все эти действия проходили так, как должны — ваша основная задача на пробном туре. Это необходимый ознакомительный этап, который применяется как к личным, так и к командным соревнованиям.

Выигрывать пробный тур и решать все предложенные там задачи можно, но не обязательно, потому что цель данного тура — знакомство и адаптация.

#### **4.4. Организационная подготовка при поездке на сборы**

Для того, чтобы развиваться в спортивном программировании и добиваться высоких результатов на контестах как и в любом спорте, необходимы регулярные тренировки. Важно работать над своим развитием системно, постоянно актуализируя имеющиеся знания и отрабатывая навыки решения задач в условиях конкурса. Один из самых эффективных видов



тренировок — это участие в тренировочных сборах. Если вы живете в большом городе, у вас, скорее всего, есть возможность участвовать в локальных сборах. При этом имеет смысл участвовать и в выездных сборах, чтобы менять окружение и соревноваться с большим количеством других команд. Если вы живете и учитесь в небольшом городе с не очень высокой контекстной активностью, вашим единственным вариантом отработки навыка участия в командных соревнованиях являются выездные сборы.

При организации участия в сборах встает ряд вопросов, которые необходимо решить для того, чтобы ваша команда смогла поехать на эти сборы.

### Финансирование

Существует несколько способов найти финансирование:

- 1) Самофинансирование (доступно далеко не всем)
- 2) Университетская поддержка
- 3) Поддержка региональных государственных учреждений
- 4) Поддержка крупными IT-компаниями, заинтересованными в развитии специалистов, которых они в дальнейшем наймут на работу
- 5) Инвестиционные фонды, например «Botan Investments»

Детально будут раскрыты только 2 пункта из списка, степень влияния на которые у вас как у команды выше всего. Речь идет о получении поддержки со стороны университета и привлечении финансирования из фондов типа «Botan Investments».

На текущий момент многие государственные организации и крупные IT-компании заинтересованы в развитии спортивного программирования в своих регионах и готовы оказывать разного рода поддержку в организации мероприятий и при подготовке команд.

Университеты по всему миру также стараются поддерживать развитие спортивного программирования. В зависимости от успехов университетских

команд администрация вуза определяет, сколько средств из бюджета заложить на развитие этого направления в течение следующего года, включая такие статьи расходов как:

- посещение командами университета тренировочных мероприятий (в том числе и международных);
- разработка и проведение локальных образовательных программ по спортивному программированию в рамках вуза;
- популяризация спортивного программирования в рамках университета и в регионе;
- организация отборочных соревнований на площадке университета;
- оплата труда тренеров, командировочные и т.д.

Уровень вовлеченности и заинтересованности студентов, тренеров, и их результативность на соревнованиях (региональных, федеральных и международных), оказывают влияние на объем выделяемого университетом бюджета финансирования на весь следующий год в рамках направления. При этом практика показывает, что даже если бюджет в университете есть, существует еще ряд шагов, которые необходимо предпринять для того, чтобы получить максимально возможную поддержку.

Зачастую ответственность за организацию поездки команды на сборы берет на себя тренер команды. Также встречаются университеты, у команд которых по каким-то причинам нет тренера, либо он имеет высокую нагрузку, не позволяющую ему уделять время команде в том объеме, в каком хотелось бы. В этом случае ответственность за организационный процесс частично или полностью перекладывается на участников команды.

В некоторых университетах существует третья сторона, которая помогает командам и тренеру организовать поездку на сборы. Речь идет об ассистентах, кураторах направления или иных ответственных лицах из административного блока.

Задача команды — максимально ускорить процесс решения формальных вопросов. Воспримите как естественный ход вещей то, что вы проявите себя самостоятельными и активными студентами.

Алгоритм действий тренера или участников команды при получении финансирования со стороны университета:

1. Определить внутри команды степень готовности каждого участника поехать на сборы, запланированные на конкретные сроки. Сообщить о намерениях своему тренеру, если вопрос участия не оговаривался заранее.
2. Связаться с ответственными лицами (деканат / ректорат / куратор направления / отдел образовательных проектов при университете) через тренера или самостоятельно для решения вопроса о выделении средств на частичное или полное финансирование поездки.
3. В случае необходимости запросить у организаторов сборов подписанные приглашения на всю вашу команду. Чтобы приглашение было сделано оперативно, необходимо сразу прислать организаторам данные ректора или лица, временно исполняющего обязанности ректора. Передать распечатанное подписанное приглашение ответственными лицами для дальнейшего одобрения.
4. Через тренера или самостоятельно получить контакт лица, которое будет курировать все процессы, касающиеся поездки, и при необходимости передать его организаторам сборов для дальнейшего взаимодействия.
5. Глобальная задача для всех сторон — соблюсти сроки проведения внутренних приказов вуза, т.к. регламент проведения приказов у каждого учебного заведения уникальней. Существуют примеры, когда команда была готова ехать на сборы, финансовую поддержку университет готов был оказать, однако при этом провести все формальные операции в рамках университета в нужный срок не удалось из-за особенностей

проведения локальных процедур согласования. Это приводило к тому, что команда не ехала на сборы, что негативно отражалось на уровне ее подготовки к чемпионату.

Тайм-менеджмент — это не только практика эффективного управления своим временем, это также практика эффективного управления временем других участников процесса. Таким образом, крайне важно делать все заранее, примерные сроки для организации поездки от 1-1,5 месяца и больше, в зависимости от университета. К примеру, командам из университетов некоторых зарубежных стран стоит начинать совершать первые шаги из списка за 2-3 месяца до начала сборов в силу необходимости получения виз и других разрешительных документов на пересечение границ.

Целеполагание — важная составляющая любого процесса. Важно понимать, к чему вы стремитесь, и что нужно сделать, чтобы этого достигнуть. Исходя из поставленных целей вы будете понимать, на какие сборы вам нужно поехать, в каких дивизионах участвовать, когда следует обращаться в ВУЗ за поддержкой.

Составьте годовой план нужных вам мероприятий и предупредите об этом вуз заранее, в момент когда бюджет на будущий год только формируется. Такой подход однозначно увеличит ваши шансы на получение финансирования. Чаще всего бюджет вузовских подразделений создается в октябре-ноябре текущего года на следующий календарный год, однако некоторые вузы согласуют его поквартально. В любом случае, чем раньше вы заговорите с материально ответственными людьми о вашем намерении представлять вуз на международном чемпионате и о необходимости оказать вам для этого финансовую помощь, тем выше вероятность, что на вашу просьбу откликнутся.

Альтернативный способ получения финансовой поддержки — обращение в фонд «Botan Investments». Основателем фонда является выходец из ICPC — Виктор Шабуров. На работу в свои компании он среди прочих привлекает

спортивных программистов, а с некоторыми из них реализует успешные совместные проекты, именно поэтому его фонд очень часто оказывает частичную или полную финансовую поддержку командам, заявляющим свое желание поехать на сборы.

Для получения помощи необходимо подать заявку на сайте [botaninvestments.com](http://botaninvestments.com) и написать мотивационное письмо от лица вашей команды. Для вашего удобства команда Moscow Workshops подготовила пример спонсорского письма, которое можно отправить потенциальным спонсорам. Его можно найти в Приложении. Всё что нужно сделать — внести данные о вашей команде в поля шаблона, описать ваши достижения и указать, почему вам важно попасть на эти сборы. **Важно!** Не повторяйте текст примера, пишите о себе сами, своим языком, иначе в Botan Investments будут приходить одинаковые письма, и вы рискуете получить отказ. По опыту Moscow Workshops команды, пишущие аргументированные мотивационные письма, регулярно ездят на сборы за счет Botan Investments. При этом период одобрения заявки фондом намного меньше, чем при одобрении того же запроса университетом.

## **5. Развитие междисциплинарных гибких навыков**

Гибкие навыки (soft skills) — те знания и умения, которые не относятся напрямую к специфическим профессиональным навыкам, например, к разработке или тестированию, но необходимы для успешного взаимодействия в команде.

Швейцарская организация «Всемирный экономический форум» (ВЭФ), наиболее известная проведением ежегодных встреч в Давосе, регулярно опрашивает руководителей кадровых служб и стратегов ведущих мировых работодателей, и, базируясь на этой информации, готовит аналитический отчет «The Future of Jobs». В этом отчете выделяются главные тенденции мирового рынка труда, и прогнозируются будущие тренды.

По последним данным экспертов ВЭФ, работодатели ожидают более широкого сочетания компетенций, поэтому растет важность тандема профессиональных технологических и междисциплинарных гибких навыков. Поскольку автоматизация и алгоритмизация создают новые рабочие места и уничтожают другие, спрос на такую комбинацию навыков, скорее всего, будет только увеличиваться.

Исследования платформы LinkedIn, сервиса для поиска и установления деловых контактов, оперирующего в 150 странах мира с 630 миллионами зарегистрированных пользователей (на июнь 2019 года), показывают, что после креативности, самые востребованные гибкие навыки — это убеждение, сотрудничество, адаптивность и тайм-менеджмент. Эксперты ВЭФ считают, что востребованность этих компетенций, вероятно, будет развиваться с течением времени, наряду со способностями к активному обучению и социальному влиянию.

Эксперты советуют помнить, что залог успешного развития гибких навыков — это их ежедневное применение и развитие по системе SMART, где

S — Specific/Конкретность;

M — Measurable/Измеримость;

A — Attainable/Достижимость;

R — Relevant/Уместность;

T — Time-bound/Ограниченность во времени.

Очевидно, что если вы будете развивать не только свои профессиональные навыки в программировании, но и навыки, связанные с умением гибко перестраиваться, грамотным тайм-менеджментом, способностью эффективно работать в команде и адаптироваться в изменяющихся условиях, то ваша востребованность на рынке труда и успешная самореализация будут гарантированы.

### **Тайм-менеджмент**

Индивидуально.

Тренируйте скорость набора кода и скорость набора текста в целом, это будет полезно для развития скорости выполнения любых задач, в том числе задач по спортивному программированию.

Командно.

Научитесь распределять задачи по силам. Во время конкурса не тратьте время на одну задачу, если вы послали ее 2-3 раза и она не прошла, нужно не заикливаться, а решать другую. Если останется время, вернетесь к ней.

Умейте переходить на более простые задачи со сложных — это ценный навык. Над сложной задачей можно работать втроем одновременно в том случае, когда остальные задачи вы уже решили: один вычитывает задачу, второй — пишет код, третий участник придумывает тестовые случаи.

Если времени и нерешенных задач осталось мало — один из участников может присоединиться к пишущему сокоманднику, чтобы снизить вероятность совершить ошибку при усиливающемся из-за недостатка времени стрессе.

При работе в команде отработать навык тайм-менеджмента поможет только практика совместной работы.

## **Командообразование**

На этапе подбора команды для себя и себя для команды, метод проб и ошибок — это то, что вполне работает.

Тимбилдинг на ранних стадиях тоже может сработать. Делайте совместную работу и просто проводите время вместе, чтобы лучше друг на друга настроиться.

Так или иначе, все прорабатывается только практикой: чем больше вы как команда вместе прошли тренировок и контестов, тем лучше.

В ходе совместной работы вы учитесь понимать сильные и слабые стороны друг друга, и, что важно, — читать код друг друга.

Иногда отладить рабочее взаимодействие может помочь использование метода включенного пассивного наблюдения (человеком извне команды, способным проанализировать происходящее), например, с использованием протокола наблюдения по схеме Джеймса Спрэдли.

Допустим, команда пишет контест и наблюдатель фиксирует по контрольному листу Спрэдли следующие пункты:

- 1) пространство
- 2) действующие лица
- 3) деятельность
- 4) объекты (все находящиеся в данном пространстве предметы)
- 5) акты (единичное действие)
- 6) события
- 7) время
- 8) цели
- 9) чувства/эмоции



*Пример:*

- 1) пространство: компьютерный класс университета;
- 2) действующие лица: Леша, Саша и Костя, + наблюдатель Антон;
- 3) деятельность: решение такой-то задачи конкурса, взятой с такого-то ресурса;
- 4) объекты: 8 столов, 8 кресел, 8 компьютеров, фикус в кадке;
- 5) акты: в 20.11 Костя и Саша вошли в класс, в 20.12 Леша вошел в класс, в 20.12 Леша сказал «Привет», и т.д.;
- 6) события: в 20.54 в класс заглянула уборщица Екатерина Петровна и спросила, когда они закончат, так как ей нужно вымыть пол и закрыть помещение;
- 7) время: вторник, 1 декабря, наблюдаемое событие длилось в течение 68 минут, с 20.10 до 21.18
- 8) цели: проработка решения задач типа X...
- 9) чувства: во время решения задачи Саша неоднократно проявлял агрессию по отношению к Косте, это выражалось в том, что... Костя в этот момент был крайне недоволен, это проявлялось в том, что...

Затем все зафиксированное обсуждается с командой, анализируется, после сделанных выводов — работа над ошибками.

Ремарка: если вам длительное время не удастся собраться на тренировки втроем, и чаще это происходит по не очень серьезным причинам, — это может быть признаком того, что вы не очень мотивированы работать вместе именно в таком составе. Возможный вариант решения — общение с другими людьми, среди которых вы подберете себе подходящую команду.

Основной совет здесь — больше ездить по сборам и решать много туров.

## **Креативность, когнитивная гибкость и критическое мышление**

Честный самоанализ — всегда полезная вещь. Анализ способа действий и пристрастий сокомандников — тоже полезная вещь.

Старайтесь отслеживать себя на два типа крайностей:

- не нравится определенный тип задач, но пишу их лучше всех в команде: в этом случае практикой можно наработать привычку спокойно решать задачи такого нелюбимого формата, это будет только на пользу всей команде; всегда возможен вариант, что во время контеста не будет задач того типа, в которых вы сильны и которые вам нравятся, поэтому умение решать задачи разного типа будет преимуществом и индивидуальным, и командным, во время любого контеста вы сможете задействовать свои навыки;
- можно, наоборот, быть фанатом какого-то типа задач, но не быть самым сильным в команде в решении задач этого типа; очевидно, что забирать такие задачи себе во время турнира нерационально, так что учитесь отпускать ситуацию и работать на командный результат.

Если вы всей командой не можете решить задачу, учитесь решать не ее, а другую. Всегда старайтесь трезво оценить ситуацию и найти оптимальный вариант.

Идеально, если члены команды разбираются во всех темах, и при этом каждый человек имеет свои специализации, чтобы не было потенциальных разногласий о том, кто какие задачи будет решать.

И помните, что сложные задачи всегда легче решать совместно.

Совет тот же: прорабатывайте все на практике, участвуя в как можно большем количестве сборов и турниров.

## **Принятие решений, управление конфликтами и переговорные компетенции**

Дойти до высокого этапа любых серьезных соревнований при наличии внутри команды нерешенных проблем и без умения взаимодействовать, даже если команда состоит из трех очень сильных одиночных участников, будет сложно.

Можно даже выйти в полуфинал или финал серьезного турнира, но так как на высоких этапах задачи обычно сложные и вязкие, при отсутствии практики совместной работы победить будет трудно.

В ходе совместной практики команды должны учиться:

- вместе работать над ошибками;
- читать и понимать код друг друга;
- прорабатывать все разногласия до выхода на контеcт;
- знать свои сильные и слабые стороны как команды;
- распределять порядок чтения задач (договариваться заранее, кто берет задачи с начала, кто — середину, кто — задачи в конце);

Оценивайте вероятность того или иного конфликта, распишите правила поведения во время контеcта на каждый случай: может быть полезен регламент замен за компьютером и регламент действий в случае обнаружения ошибки.

Прогресс приходит только с практикой.

## **Управление своими ресурсами**

Нарабатывайте базовые навыки на личных тренировках, а не в команде. Не превращайте командную тренировку в личную, оттачивайте взаимодействие между собой, создавайте комфортную обстановку внутри группы до выхода на этапы серьезных турниров.

Во время контеста отслеживайте свое состояние и состояние сокомандников, не допускайте избыточной усталости и переутомления; стремитесь к наиболее оптимальному распределению ресурса времени и сил в соотношении с ожидаемым результатом.

На тренировках решайте как можно более сложные задачи, чтобы отработать тактику и оценить емкость своего потенциала.

Как и в любой другой деятельности, для эффективной работы и высокой результативности вам нужен отдых от тренировок и соревнований.

Проблема выгорания существует, мониторьте свое состояние и обсуждайте с сокомандниками в случае необходимости.

Лето — время, которое можно использовать для нормализации своих личностных ресурсов, а еще для изменения команды, в случае надобности. Правильно использованное лето — залог готовности к новому сезону, который стартует осенью.

## 6. Приложения

### Пример спонсорского письма

Ниже представлен пример спонсорского письма, который вы можете использовать для получения спонсорской поддержки. Необходимо адаптировать его под себя.

---

*Кому:*

*декану своего вуза*

*генеральному директору компании N*

*губернатору*

*Уважаемый Борис Григорьевич!*

*Меня зовут Иван Иванов, я студент 4 курса факультета прикладной математики Саратовского государственного университета. Как Вы знаете, многие из самых успешных программистов современности в студенчестве участвовали и побеждали в олимпиадах и соревнованиях по спортивному программированию. Один из хороших примеров — сооснователь “ВКонтакте” Николай Дуров, который в 2000 году стал чемпионом старейшего и самого престижного из соревнований по спортивному программированию International Collegiate Programming Contest или сокращенно ICPC.*

*Чемпионат ICPC ежегодно объединяет более 50 тысяч студентов всего мира в гонке за медали. Российские команды с 2000 года уже завоевали 33 медали, Китай — всего 13, США — 12. Последние 8 лет россияне становились абсолютными лидерами соревнования. Эти успехи связаны с тем, что олимпиадное движение в России очень развито, и существуют программы подготовки, способствующие росту и заметному успеху на мировой арене. Сезон 2019-2020 станет для олимпиадных программистов особенно ответственным, потому что финал ICPC в 2020 году состоится в Москве, а*

*его проведение поддержал Председатель Правительства России Дмитрий Анатольевич Медведев.*

*Самый большой проект, который готовит будущую элиту мира программирования к чемпионату ICPC зародился в России. Он называется Moscow Workshops, и в нем преподают чемпионы ведущих IT-соревнований. В 2019 году 11 из 12 команд-победителей финала ICPC тренировались на этих сборах. Всего в программе обучились 2200 студентов 55 стран (205 вузов мира). В этих сборах я бы хотел принять участие и поднять свой уровень навыков в алгоритмах.*

*В числе моих заслуг в области спортивного программирования — победа на заключительном этапе Всероссийской олимпиаде школьников по информатике в 2016 году, третье место на Чемпионате урала в 2017 году в составе команды UU2, выход в полуфинал ICPC региона Северная Евразия. Я занимаюсь исследованиями в области машинного обучения, и создал свой проект, который востребован у 100 пользователей.*

*//Я нацелен на развитие в области соревнований по программированию, что поможет мне показывать высокие результаты и повышать престиж вуза на международных соревнованиях.*

*//Я нацелен на развитие в области соревнований по программированию и хотел бы реализовать свои навыки в проектах Вашей компании.*

*//Я нацелен на развитие в области соревнований по программированию и уверен, что высокие достижения позволят достойно представить наш регион.*

*В связи с этим прошу Вас поддержать мою инициативу о поездке на сборы по спортивному программированию [вставить название] для подготовки к чемпионату ICPC и другим важнейшим соревнованиям.*

*С уважением,*

*Иван Иванов*

*(дата, подпись)*

---

## Задачи для начального уровня (дивизиона С)

Новички, обычно относящиеся к дивизиону С на тренировочных сборах Moscow Workshops, прежде чем приступить к задачам слушают тематические лекции тренеров по конкретным темам. После лекций команды приступают к решению контестов и закреплению пройденного материала.

Ниже приведен пример лекции на тренировках в Москве при МФТИ одного из преподавателей в формате видео, а также представлен ряд задач по изученным темам.

Ссылка на лекцию по теме «Динамическое программирование на подотрезках»: <https://clck.ru/MNK38>

Задачи по теме:

## E23. Свертка

Язык	Ограничение времени	Ограничение памяти	Ввод	Вывод
Все языки	0.2 секунды	256Mb	стандартный ввод или folding.in	стандартный вывод или folding.out
Python 3.2	1.5 секунд	256Mb		
Python 2.7	1.5 секунд	256Mb		
Oracle Java 8	0.5 секунд	256Mb		
Oracle Java 7 x32	0.5 секунд	256Mb		

Петя хочет сократить запись последовательности, состоящей из заглавных латинских букв. Для этого он может свернуть ее повторяющиеся подпоследовательности. Например, последовательность АААААААААВАВАВССD может быть записана как 10(A)2(BA)B2(C)D. Формальной определением свернутой последовательности и соответствующей ей операции развертки дается следующим образом:

- Последовательность, которая содержит единственный символ от 'A' до 'Z' представляет из себя свернутую последовательность. При развертке такой последовательности получается она сама.
- Если  $S$  и  $Q$  — свернутые последовательности, то  $SQ$  также свернутая последовательность. Если при развертке строки  $S$  получается строка  $S'$ , а при развертке  $Q$  получается  $Q'$ , то при развертке  $SQ$  получается строка  $S'Q'$ .

- Если  $S$  — свернутая последовательность, то  $X(S)$  также свернутая последовательность, где  $X$  это десятичное представление целого числа большего единицы. Если при развертке строки  $S$  получается строка  $S'$ , то при развертке  $X(S)$  получается строка  $S'$ , повторенная  $X$  раз.

Петя хочет свернуть заданную последовательность таким образом, чтобы результат содержал наименьшее число символов.

## Формат ввода

Входной файл содержит непустую строку, состоящую из заглавных латинских букв. Длина строки не превышает 100 символов.

## Формат вывода

В выходной файл выведите одну строку, содержащую наименьшую последовательность развертка которой даст строку, заданную во входном файле. Если ответов несколько — выведите любой из них.

## Пример

Ввод	Вывод
AAAAAAAAAABABABCCD	9(A)3(AB)CCD



## A23. Газета

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Выход	стандартный вывод или output.txt

«Sam Torpe был найден мертвым в своей квартире» — большими буквами было написано на обложке местной газетки, которую так любил читать дома по вечерам Мистер Холмс. «Уже четвертый иностранец за месяц», — грустно подумал он. — «Завтра будет много работы». Холмс выключил свет и, повалившись на кровать от усталости, сладко уснул.

Мистер Холмс работает детективом, и, конечно же, именно ему поручено исследовать дело об этом серийном убийце. «Никаких улик, никаких следов борьбы. Как тут вообще что-то можно найти? Даже медики не могут определить, как они умерли», — думал он, сидя на работе. Никакой зацепки не было, можно было надеяться только на чудо.

Проведя весь день на работе, безуспешно пытаясь найти хоть какую-нибудь маленькую улику, мистер Холмс, грустный и уставший, потопал домой. Шел он, как обычно, через лавку с газетами, чтобы купить свежий выпуск его любимой газеты. «Здравствуйте, Чед, мне как обычно» — «Вы выглядите неважно, мистер Холмс. Возьмите отпуск, отдохните. Вот, держите газету». «Спасибо за совет», — ответил Холмс, взял газету и побрел домой.

Пожинав, Холмс сел на свое любимое кресло перед камином, и, достав свою любимую газету, расслабился после тяжелого рабочего дня. «Что-то не так», — подумал он, увидев в своей привычной газете другой шрифт и формат текста. Он почувствовал, что и бумага недостаточно мягкая, как полагается его любимой газете. «Что мне подсунули? Это не та газета!» — снова напрягся Холмс. ««Stopgame». Ну что за заголовок? Что это за странная газета? Ненавижу этот день!». Холмс бросил газету в камин, и отправился спать.

Под утро у детектива не вылетало из головы название газеты. «Stopgame... Stopgame... Как будто я уже недавно это где-то видел...» — думал он, идя на работу.

«Точно!» — громко воскликнул Холмс в полицейском участке, чем привлек внимание всех вокруг. Детектив быстро взял какую-то бумажку, написал на ней «Stopgame», порвал ее на несколько частей и сложил их. «Sam Torpe», — обрадовался Холмс. — «Но почему? Может, это случайность?». Детектив выскочил из участка и побежал к продавцу газет.

«Вы мне вчера дали не ту газету!» — запыхавшись, сказал Холмс — «Дайте мне такую же, пожалуйста, на ней написано «Stopgame»». Чед хотел было извиниться, но, видя сияющие глаза Холмса, понял, что, кажется, он ему только помог. Чед протянул Холмсу газету, тот выхватил ее из рук и принялся спешно читать ее.

“Какая-то статья про зависимых от компьютерных игр детей... Ничего интересного. Может, просто совпадение?” — подумал Холмс, взял газету, и пошел обратно в офис.

Перечитав газету, Холмс не нашел ничего интересного. Посмотрел данные о самой газете — уже более десяти лет она продается в ближайших районах. Потеряв надежду, Холмс отбросил газету на дальний край своего стола и принялся снова искать зацепки.

Прошло еще 3 дня. Холмс потерял. “Их убивает что-то изнутри. Что-то просто отнимает у них жизнь. И уже довольно скоро появится новая жертва. Все они умирают вечером, все во вторник. А завтра вторник. Нужно срочно что-то делать”. Холмс увидел отброшенную газету, и его посетила одна гениальная мысль.

Холмс рванул в дома, где жили иностранцы. У каждого из них он нашел эту газету последнего выпуска, которую еще могли купить убитые. “Бинго! Зацепка!” — возрадовался Холмс. — “Имя каждого из убитых — подпоследовательность букв из заголовка газеты!”.

“Кто же это мог быть? Как эти имена так удивительно могут образовывать другие слова?”. Холмс решил проверить сотрудников редакции газеты. Оказалось, главный по заголовкам в последнее время работает на дому. “Мы его видели за странным делом”, — говорят сотрудники, — “он нарезал бумагу и клеил в странную кожаную потертую тетрадку”.

Холмс работал всю ночь напролет. Он выяснил, кто этот сотрудник, и узнал, что всех этих иностранцев он знал — они вместе воевали. Под утро Холмс решил навестить его. Но не тут-то было: его не было дома.

Газета выходила каждый вторник утром. Холмс решил сходить за ней. Взяв газету, он передал ее своему напарнику, а сам решил выяснить, кто может быть следующей жертвой. Холмс, всю ночь провозившись с делом странного сотрудника редакции газеты, быстро справился с этой задачей, после чего раздался громкий звонок телефона Холмса.

“Он в своем доме, делает что-то странное”, — сообщили ему коллеги. И тут Холмс понял, что у него совсем мало времени. Но Холмс хочет знать точно, сколько времени убийца потратит на разрезание названия газеты.

“Эй, Ватсон, посчитай мне быстро, сколько разрезов ему нужно сделать, чтобы собрать имя и фамилию убитого?”

Сегодня Ватсон — это вы. Помогите Холмсу определить минимальное количество времени, которое уйдет у убийцы на разрезание названия газеты.

## Формат ввода

Длину строки  $q$  обозначим как  $|q|$  ( $|abac| = 4$ ,  $|sam| = 3$ ).

Во входном файле содержатся 3 строки  $s$ ,  $t_1$ ,  $t_2$  ( $|s| = |t_1| + |t_2|$ ,  $1 \leq |t_1|, |t_2| \leq 1\,000$ ). Все строки содержат только маленькие латинские буквы.

## Формат вывода

Подстрокой  $p[l, r]$  назовем строку, состоящую из символов  $p_l, p_{l+1}, \dots, p_{r-1}, p_r$ , где  $p_i$  —  $i$ -тый символ строки  $p$  (« $cde$ » является подстрокой строки « $abcdef$ », « $bcd$ » — не является).

Тогда любую строку  $p$  можно представить в виде разбиения на непересекающиеся подстроки  $p[x_0 + 1, x_1] + p[x_1 + 1, x_2] + p[x_2 + 1, x_3] + \dots + p[x_{n-1} + 1, x_n]$ , где

$0 = x_0 < x_1 < x_2 < \dots < x_{n-1} < x_n = |p|$ . При этом длиной разбиения будем называть количество чисел  $x_i$ , то есть число  $n$ .

В данной задаче требуется найти разбиение строки  $s$  такое, чтобы из получившихся подстрок можно было сложить строки  $t_1$  и  $t_2$ , не нарушая порядка, в котором подстроки шли в исходной строке. Например, из строки « $adbc$ » нельзя будет выложить слова « $cad$ » и « $b$ », так как порядок подстрок « $ad$ » и « $c$ » будет нарушен.

Программа должна выдавать на экран одно число — минимально возможную длину такого разбиения, то есть количество разрезов, которые потребуются сделать убийце.

Гарантируется, что существует разбиение строки  $s$  хотя бы одним способом, чтобы получились строки  $t_1$  и  $t_2$ .

### Пример 1

#### Ввод

stopgame  
sam  
topge

#### Вывод

3

### Пример 2

#### Ввод

ababbaba  
abbb  
abaa

#### Вывод

4

## Примечания

В первом примере из условия строку можно разрезать следующим образом: stopgame  $\rightarrow$  s | topg | am | e.

Во втором примере: ababbaba  $\rightarrow$  ab | abb | a | b | a.

## V23. Восстановление HTML-файла

Ограничение времени	2 секунды
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Петя недавно скачал поврежденный HTML-файл. Он выглядит как обычный HTML-файл, но в нем есть некоторые несоответствия тэгов. Ваша задача — удалив минимальное количество открывающихся и закрывающихся тэгов, сделать так, чтобы структура тэгов стала правильной. Более формально, HTML-файл состоит из произвольных символов с ASCII кодами из диапазона от 32 до 126, а также Linux-style переводов строки (символов с кодом 10). Тэги открываются следующим образом: `<Имя тэга Параметры>` и закрываются следующим образом: `</Имя тэга>`. Имя тэга — строка, состоящая из больших и маленьких латинских букв, которые считаются различными. *Имя тэга* отделяется от *Параметров* как минимум одним пробелом (но не переводом строки). Параметры могут содержать произвольные допустимые ASCII символы кроме `<`, `>` и переводов строк. Также допускаются открывающиеся тэги без параметров и записываются в следующей форме: `<Имя тэга>`.

HTML-файл считается правильным, если каждому открывающемуся тэгу можно привести в соответствие следующий далее в файле закрывающийся тэг таким образом, чтобы часть файла между этими тэгами также представляла собой правильный HTML-файл, и аналогично можно привести в соответствие каждому закрывающемуся тэгу ровно один открывающийся тэг, идущий ранее в файле. HTML-файл, не содержащий тэгов, также является правильным.

### Пример 1

**Ввод**

```
<a href=http://it-edu.mipt.ru>
<b someone has corrupted this file>
It was a good file before...
</a>
</b>
```

**Вывод**

```
2
```

### Пример 2

**Ввод**

```
<a><b>That's good</b></a>
```

**Вывод**

```
0
```

## C23. Подготовка к олимпиаде

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Тимур хочет выиграть открытую олимпиаду ЛОШ. Чтобы подготовиться, Тимур хочет порешать задачи с одного известного сайта с автоматической системой проверки задач.

Сайт устроен следующим образом: у каждого пользователя есть свой рейтинг  $R$ . У каждой задачи на этом сайте есть сложность  $s_i$ . Задача доступна для сдачи только тем пользователям, у которых рейтинг не меньше сложности этой задачи ( $R \geq s_i$ ). Это нужно для того, чтобы новичок не принялся решать сложную задачу. После решения задачи рейтинг участника увеличивается на  $p_i$ . Каждую из задач можно только один раз.

Тимур изучил все задачи и для каждой оценил число дней  $t_i$ , которое ему потребуется для ее решения. Теперь Тимур хочет спланировать оставшееся до олимпиады время таким образом, чтобы достигнуть максимально возможного рейтинга. Сейчас у Тимура рейтинг  $R_0$ , и до олимпиады осталось  $T$  дней.

Зная эти данные, помогите Тимур выбрать оптимальный порядок решения задач, приводящий его к максимально возможному рейтингу.

### Формат ввода

В первой строке входных данных задано три целых числа:  $n$ ,  $T$  и  $R_0$  — количество задач на сайте, количество дней до олимпиады и рейтинг Артура на текущий момент.

В следующих  $n$  строках задано по три целых числа  $s_i$ ,  $p_i$  и  $t_i$  — сложность задачи, увеличение рейтинга за решение задачи и количество дней, которое требуется на её решение.

$n \leq 1000$ ;  $T, t_i \leq 1000$ ;  $R_0, s_i \leq 10^9$ ;  $p_i \leq 10^6$ . Все числа во входных данных положительные.

### Формат вывода

В первой строке выходного файла выведите максимальный рейтинг, который может иметь Тимур через  $T$  дней.

Во второй строке выведите номера задач, разделенные пробелами, в том порядке, в котором необходимо решать задачи, чтобы получить максимальный рейтинг. Задачи нумеруются от 1 до  $n$  в том порядке, в котором они заданы во входных данных.

## Пример 1

Ввод	Вывод
4 10 1	20
10 10 1	2 4 1
1 5 5	
7 3 1	
2 4 4	

## Пример 2

Ввод	Вывод
4 10 1	13
11 10 1	2 4 3
1 5 5	
7 3 1	
2 4 4	

## Пример 3

Ввод	Вывод
3 4 3	9
3 3 2	1 2
3 3 2	
3 5 3	

Лекция по декартовым деревьям (по явному ключу)

Ссылка на лекцию часть 1: <https://clck.ru/MNL7G>

Ссылка на лекцию часть 2: <https://clck.ru/MNL8N>

Задачи по теме приведены ниже.

## A27. Двоичное дерево поиска

Ограничение времени	2 секунды
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Реализуйте сбалансированное двоичное дерево поиска.

### Формат ввода

Входной файл содержит описание операций с деревом, их количество не превышает  $100000$ . В каждой строке находится одна из следующих операций.

- *insert x* – добавить в дерево ключ  $x$ . Если ключ  $x$  уже в дереве, то ничего делать не надо.
- *delete x* – удалить из дерева ключ  $x$ . Если ключа  $x$  в дереве нет, то ничего делать не надо.
- *exists x* – если ключ  $x$  есть в дереве, выведите «*true*», иначе «*false*»

Все числа во входном файле целые и по модулю не превышают  $10^9$ .

### Формат вывода

Выведите последовательно результат выполнения всех операций *exists*. Следуйте формату выходного файла из примера.

### Пример

Ввод	Вывод
insert 2	true
insert 5	false
insert 3	
exists 2	
exists 4	
delete 5	

## В27. Двоичное дерево поиска 2

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Реализуйте сбалансированное двоичное дерево поиска.

### Формат ввода

Входной файл содержит описание операций с деревом, их количество не превышает  $10^5$ . В каждой строке находится одна из следующих операций.

*insert*  $x$  – добавить в дерево ключ  $x$ .

*delete*  $x$  – удалить из дерева ключ  $x$ . Если его там нет, то ничего делать не надо.

*exists*  $x$  – если ключ  $x$  есть в дереве, выведите «true», иначе, «false»

*next*  $x$  – выведите минимальный элемент в дереве, строго больший  $x$  или «none», если такого нет.

*prev*  $x$  – выведите максимальный элемент в дереве, строго меньший  $x$  или «none», если такого нет.

*next*  $x$  – выведите минимальный элемент в дереве, строго больший  $x$  или «none», если такого нет.

*prev*  $x$  – выведите максимальный элемент в дереве, строго меньший  $x$  или «none», если такого нет.

*kth*  $x$  – выведите  $k$ -тый по величине элемент(нумерация с единицы). Если такого не существует, то выведите «none».

Все числа во входном файле не превышают  $10^9$ .

### Формат вывода

Выведите последовательно результат выполнения всех операций *exists*, *next*, *prev*. Следуйте формату выходного файла из примера.



## Пример

<b>Ввод</b>	<b>Вывод</b>
insert 2	true
insert 5	false
insert 3	5
exists 2	3
exists 4	none
next 4	3
prev 4	2
delete 5	none
next 4	
prev 4	
kth 1	
kth 3	

## C27. Декартово дерево

Ограничение времени	1 секунда
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Вам даны пары чисел  $(a_i, b_i)$ . Вам необходимо построить декартово дерево, такое что  $i$ -ая вершина имеет ключи  $(a_i, b_i)$ , вершины с ключом  $a_i$  образуют бинарное дерево поиска, а вершины с ключом  $b_i$  образуют кучу.

### Формат ввода

В первой строке записано число  $N$  — количество пар. Далее следует  $N$  ( $1 \leq N \leq 50\,000$ ) пар  $(a_i, b_i)$ . Для всех пар  $|a_i|, |b_i| \leq 30\,000$ .  $a_i \neq a_j$  и  $b_i \neq b_j$  для всех  $i \neq j$ .

### Формат вывода

Если декартово дерево с таким набором ключей построить возможно, выведите в первой строке YES, в противном случае выведите NO. В случае ответа YES, выведите  $N$  строк, каждая из которых должна описывать вершину. Описание вершины состоит из трёх чисел: номер предка, номер левого сына и номер правого сына. Если у вершины отсутствует предок или какой-либо из сыновей, то выводите на его месте число 0. Если подходящих деревьев несколько, выведите любое.

### Пример

Ввод	Вывод
7	YES
5 4	2 3 6
2 2	0 5 1
3 9	1 0 7
0 5	5 0 0
1 3	2 4 0
6 6	1 0 0
4 11	3 0 0

## D27. Слоники

Ограничение времени	6.5 секунд
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Это интерактивная задача.

Добро пожаловать в самый большой в мире рынок слоников! Так как вы новичок на этом рынке, вашей целью является научиться анализировать текущее состояние рынка. Вы должны понимать, сколько денег вы можете заработать в любой момент времени, исходя из текущих предложений, совершая любые покупки и продажи, какие вы хотите. Но вы не будете торговать в реальности. Только анализировать.

Рынок слоников имеет следующую модель: продавцы могут сами назначить цену на слоника, а покупатели – запросить слоника за определенную цену. Цены меняются постоянно, как только торговцам это взбредет в голову.

Мы подразумеваем, что вам дана текущая ситуация рынка, и вы можете купить несколько слоников и сразу их продать. Вы покупаете и продаете слоников в течение минуты. Вы не заинтересованы в хранении слоников. Так же вы можете покупать и продавать столько слоников, сколько вы захотите!

На открытии рынка слоников никто не хочет покупать или продавать слоников. Известно лишь, какие изменения происходят на рынке. Для простоты будем считать, что изменения имеют следующий вид: "buy 10 100". Это значит, что спрос на слонов стал на 10 слонов больше с ценой спроса 100. Или "sell -3 99", что обозначает, что предложение слонов стало на 3 слона меньше, а цена продажи стала 99.

Напишите программу, которая вычисляет максимальную прибыль, которую вы можете получить после перепродажи слонов. Заметьте, что вы не совершаете никаких продаж и покупок, это значит, что при появлении нового предложения (или изменения количества слоников с такой ценой), все предыдущие предложения еще действительны.

### Формат ввода

Ваша программа должна принимать предложения с рынка последовательно, в следующем формате: каждая строка входных данных обозначает предложение, и начинается с типа предложения (одна из строчек "buy", "sell" или "end"), которая обозначает, что изменилось количество покупаемых или продаваемых слоников, либо изменения закончились. Предложение "end" больше ничего не содержит, и после него ваша программа должна непременно закончить работу.

Другие предложения содержат по 2 числа, разделенных пробелом  $D$  и  $P$ , где  $D$  – изменение количества продаваемых или покупаемых слоников по цене  $P$  за каждого. ( $-10^6 \leq D \leq 10^6$ ,  $1 \leq P \leq 10^9$ ). Суммарное число всех предложений не превышает  $10^5$ . Для лучшего понимания, смотрите примеры.

Гарантируется, что сумма цен всех за всех продаваемых слоников и сумма цен за всех покупаемых слоников на каждый момент времени не превышает  $2^{62}$ , и что число слоников для каждой цены для покупки или продажи никогда не отрицательно.

## Формат вывода

Для каждого “buy” или “sell”, вы должны вывести число в отдельной строке: максимальную прибыль в пиастрах, которую вы можете получить, занимаясь перепродажей слоников. Не забывайте делать перевод строки и flush.

## Примечания

Вход от интерактора

```
buy 10 100
```

```
sell 4 98
```

```
buy -7 100
```

```
buy 2 99
```

```
sell 1 97
```

```
end
```

Ответы программы участника:

```
0
```

```
8
```

```
6
```

```
7
```

```
9
```

## E27. И снова сумма...

Ограничение времени	2 секунды
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Реализуйте структуру данных, которая поддерживает множество  $S$  целых чисел, с которым разрешается производить следующие операции:

- $add(i)$  — добавить в множество  $S$  число  $i$  (если он там уже есть, то множество не меняется);
- $sum(l, r)$  — вывести сумму всех элементов  $x$  из  $S$ , которые удовлетворяют неравенству  $l \leq x \leq r$ .

### Формат ввода

Исходно множество  $S$  пусто. Первая строка входного файла содержит  $n$  — количество операций ( $1 \leq n \leq 300\,000$ ). Следующие  $n$  строк содержат операции. Каждая операция имеет вид либо «+  $i$ », либо «?  $l$   $r$ ». Операция «?  $l$   $r$ » задает запрос  $sum(l, r)$ .

Если операция «+  $i$ » идет во входном файле в начале или после другой операции «+», то она задает операцию  $add(i)$ . Если же она идет после запроса «?», и результат этого запроса был  $y$ , то выполняется операция  $add((i + y) \bmod 10^9)$ .

Во всех запросах и операциях добавления параметры лежат в интервале от 0 до  $10^9$ .

### Формат вывода

Для каждого запроса выведите одно число — ответ на запрос.

### Пример

Ввод	Вывод
6	3
+ 1	7
+ 3	
+ 3	
? 2 4	
+ 1	
? 2 4	

## Задачи для среднего уровня (дивизиона В)

Ниже приведен еще один ряд тематических задач дивизиона В, а также лекция по ним, прошедшая на московских тренировках под руководством тренера Moscow Workshops Филиппа Руховича.

Ссылка на лекцию по теме «Продвинутые оптимизации в ДП»:  
<https://clck.ru/MPEf5>

Задачи по теме:

### A25. Египетские дни рождения

Ограничение времени	0.5 секунд
Ограничение памяти	600Mb
Ввод	стандартный ввод или birthday.in
Вывод	стандартный вывод или birthday.out

В недавней археологической экспедиции в Египте Василий Ильич и его коллеги нашли каменную плиту, на которой было много похожих записей. По ряду причин археологи решили, что это — списки рабов, задействованных на строительстве пирамид: их имена, дни рождения, родные города. Василий Ильич очень интересуется, как же надсмотрщики различали рабов, если у них совпадали все эти параметры. Для начала он хочет определить, сколько среди записей на плите различны. Помогите ему.

#### Формат ввода

Первая строка содержит число  $n$  - количество записей на плите. Далее  $n$  строк, содержащих записи. Каждая строка состоит из маленьких латинских букв и не является пустой.

Гарантируется, что суммарная длина строк не превосходит  $5 \cdot 10^6$ .

#### Формат вывода

Выведите количество различных записей.

## Пример 1

<b>Ввод</b>	<b>Вывод</b>
4	3
a	
aa	
aab	
aa	

## Пример 2

<b>Ввод</b>	<b>Вывод</b>
3	1
a	
a	
a	

## B25. К-я строка

Ограничение времени	1 секунда
Ограничение памяти	64Mb
Ввод	стандартный ввод или kthstr.in
Вывод	стандартный вывод или kthstr.out

Реализуйте структуру данных, которая поддерживает следующие операции:

- добавить в словарь строку  $S$ ;
- найти в словаре  $k$ -ю строку в лексикографическом порядке.

Известно, что изначально словарь пуст.

### Формат ввода

Первая строка входного файла содержит натуральное число  $N$  — количество команд ( $N \leq 10^5$ ). Последующие  $N$  строк содержат по одной команде каждая.

Команда записывается либо в виде числа  $k$ , либо в виде строки  $S$ , которая может состоять только из строчных латинских букв. Гарантируется, что при запросе  $k$ -й строки она существует. Также гарантируется, что сумма длин всех добавляемых строк не превышает  $10^5$ .

### Формат вывода

Для каждого числового запроса  $k$  выходной файл должен содержать  $k$ -ю в лексикографическом порядке строчку из словаря на момент запроса. Гарантируется, что суммарная длина строк в выходном файле не превышает  $10^5$ .

### Пример

Ввод	Вывод
7	tolstoy
pushkin	gogol
lermontov	
tolstoy	
gogol	
gorkiy	
5	
1	



## C25. Шифр Бэкона

Ограничение времени	1 секунда
Ограничение памяти	256Mb
Ввод	стандартный ввод или bacon.in
Вывод	стандартный вывод или bacon.out

Программисту Васе не повезло — вместо отпуска его послали в командировку на научную конференцию. «Надо повышать уровень знаний», — сказал начальник, «Важная конференция по криптографии, проводится во Франции —а там шифровали еще во времена Ришелье и взламывали чужие шифры еще во времена Виета.»

Вася быстро выяснил, что все луврские картины он уже где-то видел, вид Эйфелевой башни приелся ему еще раньше, чем мышка стерла его с коврика, а такие стеклянные пирамиды у нас делают надо всякими киосками и сомнительными забегаловками. Одним словом, смотреть в Париже оказалось просто не на что, рыбу половить негде, поэтому Васе пришлось посещать доклады на конференции.

Один из докладчиков, в очередной раз пытаясь разгадать шифры Бэкона, выдвинул гипотезу, что ключ к тайнам Бэкона можно подобрать, проанализировав все возможные подстроки произведений Бэкона. «Но их же слишком много!» — вслух удивился Вася. «Нет, не так уж и много!» — закричал докладчик, — «Подсчитайте, и вы сами убедитесь!»

Тем же вечером Вася нашел в интернете полное собрание сочинений Бэкона. Он написал программу, которая переработала тексты в одну длинную строку, выкинув из текстов все пробелы и знаки препинания. И вот теперь Вася весьма озадачен — а как же подсчитать количество различных подстрок этой строки?

### Формат ввода

На входе дана непустая строка, полученная Васей. Строка состоит только из строчных латинских символов. Ее длина не превосходит 2 000 символов.

### Формат вывода

Выведите количество различных подстрок этой строки.

## Пример

**Ввод**

**Вывод**

---

ааба

8

---

## D25. Витя и странный урок

Ограничение времени	2 секунды
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Сегодня на уроке Витя изучал очень интересную функцию — *mex*. *Mex* набора чисел — минимальное неотрицательное число, не присутствующее в наборе чисел. Например,  $mex([4, 33, 0, 1, 1, 5]) = 2$ , а  $mex([1, 2, 3]) = 0$ .

Витя очень быстро разобрался со всеми задачами учителя, а сможете ли вы?

Даны массив, состоящий из  $n$  неотрицательных целых чисел, и  $m$  запросов. Каждый запрос характеризуется одним числом  $x$  и заключается в следующих последовательных шагах:

- Выполнить операцию побитового сложения по модулю 2 (*xor*) каждого элемента массива с числом  $x$ .
- Найти *mex* полученного массива.

Учтите, что после каждого запроса массив изменяется.

### Формат ввода

Первая строка содержит два целых положительных числа  $n$  и  $m$  ( $1 \leq n, m \leq 3 \cdot 10^5$ ), обозначающие количество элементов в массиве и количество запросов соответственно. Следующая строка содержит  $n$  неотрицательных целых чисел  $a_i$  ( $0 \leq a_i \leq 3 \cdot 10^5$ ), представляющих элементы исходного массива.

Каждая из следующих  $m$  строк содержит запрос — одно неотрицательное целое число  $x$  ( $0 \leq x \leq 3 \cdot 10^5$ ).

### Формат вывода

Для каждого запроса выведите ответ на него в отдельной строке.

### Пример 1

Ввод	Вывод
2 2	1
1 3	0
1	
3	

## Пример 2

<b>Ввод</b>	<b>Вывод</b>
4 3	2
0 1 5 6	0
1	0
2	
4	

## Пример 3

<b>Ввод</b>	<b>Вывод</b>
5 4	2
0 1 5 6 7	2
1	0
1	2
4	
5	

## Задачи для продвинутого уровня (дивизиона А)

### Problem A. A day in the woods

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

People from the city really like to spend their weekends at the forester's range. A bus would take them to the forester's hut and they'd hike to forest clearings nearby to run barbecue parties. But sometimes it happened so that they lost their way and roamed from one clearing to another, taking different trails.

Alex, the head forester, decided to help the silly citizens. He put numbers on all clearings and put arrows on trails so that a citizen walking from the forester's hut along the arrows, passing on to other clearings no more than once, would find himself on clearings with increasing numbers. If this citizen decided that he wants to return, he'd have to go against the direction of the arrows. The decreasing numbers of the clearings would mean he is going in the right direction. After renumbering all the clearings should have different numbers from 1 to  $n$ .

Alex has a map of his forest where all the clearings and the trails linking the clearings are marked. Any two clearings can be linked with one or more trails. The forester put the arrows correctly, but now he is having trouble with the clearing numbers. Help him to renumber the clearings according to his plan.

#### Input

The input contains several testcases.

The first line of each test set contains three integers  $n$ ,  $m$ , and  $s$ , — the number of clearings, the number of trails, and the number of the clearing where the forester's hut is located, respectively ( $1 \leq n \leq 1000$ ,  $0 \leq m \leq 1000$ ,  $1 \leq s \leq n$ ). All clearings are enumerated with integers from 1 to  $n$ .

The next  $m$  lines describe the trails — each of them contains two integers, numbers of the clearings a trail connects in the direction of the arrow put by the forester. Note that loop trails are possible, i.e. in some cases these numbers can be equal.

The sum of  $n$  in all tests is not greater than 1000. The sum of  $m$  in all tests is not greater than 1000. The input ends with the line with three zeroes, which shouldn't be processed.

#### Output

For each test case print the numbering of the clearings in a separate line. If no acceptable numbering exists, the output must contain the phrase "No solution".

#### Example

standard input	standard output
3 3 1	1 2 3
1 2	2 3 4 1
2 3	No solution
1 3	
4 4 4	
4 1	
1 2	
2 3	
3 1	
4 6 1	
1 2	
1 3	
1 4	
2 3	
3 4	
4 2	
0 0 0	

## Problem B. Bus Stops

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

A core part of the decision of where to live in a city like Bytesburg is the availability of transport links to interesting places. This is particularly interesting to Barbara, who enlivens her stressful life as an organiser of programming camps by making frequent sightseeing travels around town in a yellow-blue bus.

Barbara's idea of a good time is a visit to a spot that takes exactly one bus journey to get to. She is considering moving her house to be near one specific spot along her favourite bus route—how many other scenic spots can she reach from there (assuming that on a given trip she can choose a new bus route each time)?

### Input

- The first line of input contains three integers: Barbara's starting stop,  $m$  ( $1 \leq m \leq s$ ), the number of buses,  $b$  ( $1 \leq b \leq 50$ ), and the number of stops,  $s$  ( $1 \leq s \leq 50$ ).
- The next  $b$  lines contain the bus routes, each written as a string of  $s$  characters where having the  $i$ th character as '1' denotes that this bus route has a stop at  $i$ , and '0' denotes that it does not.

### Output

Output the maximum number of other stops Barbara can reach from the starting stop by taking exactly one bus.

### Example

standard input	standard output
1 3 5 01100 10011 10111	3
2 2 3 101 101	0

## Problem C. Campus for SIT

Input file: *standard input*  
 Output file: *standard output*  
 Time limit: 2 seconds  
 Memory limit: 512 mebibytes

Carol is the urbanist, and now she is working on the project of the new campus for Schlaffhausen Institute of Technology (SIT). Territory of the campus can be represented as  $n$  by  $m$  grid, with beautiful lakes in some cells and forest in another. Carol wants the campus buildings to fulfill the next conditions:

1. Buildings shall be put to forest cells only, no more than one building per forest cell.
2. To make campus compact enough, each forest cell must have a building directly on it, or adjacent to it.
3. For the environmental purposes no two buildings can be adjacent to each other.

Two cells are adjacent if they share a side.

Find any placement of buildings that satisfies these constraints.

### Input

The first line of the input consists of two integers  $n$  and  $m$  ( $1 \leq n, m \leq 100$ ). The following  $n$  lines each contain a string of length  $m$  consisting only of the characters 'F' (forest) and 'L' (lake). This is the map of the land for the campus. It is guaranteed that the map contains at least one forest cell.

### Output

Output a copy of the map, where some of the land cells have been replaced with the letter 'B', meaning that a building was placed on the corresponding land cell. This placement should satisfy the constraints above. If there are many solutions, any one will be accepted.

### Examples

standard input	standard output
5 6 FFFFLF FFFFLF LFFFFF FFFFFF LLFFFL	BFFBLF FFBFLB LFFBFF FBFFFB LLBFBL
10 16 LLLLLLLFLFFFLL LLLLLLFFFFFFL LLLLLFFFFFFL LLLLLFFFFFFL LLLFFFFFFL LLFFFFFFL LLFFFFFFL LLFFFFFFL LLFFFFFFL LLFFFFFFL LLFFFFFFL LLFFFFFFL LLFFFFFFL LLFFFFFFL LLFFFFFFL	LLLLLLLBLFBFLL LLLLLLBFBFLL LLLLLBFBFLL LLLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL LLBFBFLL

## Problem D. Data Consistency

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Diana is a system administrator in the datacenter of the Byteland Astronomical Institute. The datacenter is planning to buy the special workstation for loseless real-time data communication with the satellites. Each satellite sends block of  $D$  mebibytes of the data to the Earth per session. All this data must be kept.

Workstation supports the SSD modules of  $S$  mebibytes each. Modules can be installed in multiple layers, one module per layer. When module is full, it must be immediately replaced with the empty one, but replacement takes some time and cause data loss, so no SSD module can be replaced until transaction of current block is finished.

To prevent data loss, Diana use very recent solution from Acronis called Acronis Infinite Space. It works in the next way. Each new block of the data is written to SSD in the first layer. When the SSD runs out of the free space and current block is still not received, last part of the block is written on the SSD in the second layer; if this SSD runs out of the free space while receiving this part, remaining part goes to the SSD in third layer and so on. When the block is received, all full SSD's are moved for the further processing and replaced with empty ones.

To perform the initial setup of the Acronis Infinite Space Diana wants to know minimal number of layers  $L$  needed to prevent the data loss.

### Input

The input consists of a single line containing the two integers  $S$  and  $D$  ( $1 \leq D \leq S \leq 10^{10}$ ).

### Output

Output the smallest integer  $L$  such that  $L$  layers will be enough to prevent the data loss. If it is impossible to prevent the data loss with any number of layers, print  $-1$  instead.

### Example

standard input	standard output
31 6	4

### Note

In the sample, at each 6'th session the first layer SSD is full and  $6 \cdot 6 - 31 = 5$  mebibytes go to second layer SSD. At each 6·7 session the second layer SSD is full, and  $7 \cdot 5 - 31 = 4$  mebibytes go to third layer SSD. At each 6·7·8'th session the third layer SSD is full, and  $8 \cdot 4 - 31 = 1$  mebibyte goes to fourth layer SSD. After each 6·7·8·31 time the fourth SSD is full right at the end of transaction, and no fifth SSD is needed.



## Problem E. Evenly Divided

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 second  
Memory limit: 512 mebibytes

The Byteland ICPC Community oversaw a sharp resurgence in membership this year, and must now face the inevitable strains of growth: the group photo they usually take can no longer fit everyone in one long row.

Edward, the photographer, asks members to split into two groups: Tall and Short, so that the picture can be doubled up with taller people standing behind shorter people in two rows of  $\frac{n}{2}$  each.

The ICPC Community annual meetings is an opportunity for the members to meet people. Many new joiners were assigned a mentor from the members who had already signed up before they joined. And Edward wants to choose a way of arranging the rows such that nobody is standing directly in front of or behind their mentor, assuming they have one.

Find a way of arranging the two rows such that this is possible. The number of tall people is always the same as the number of short people.

### Input

The input consists of:

- a line consisting of the number of members in the mountaineering society, which is a positive even integer  $m$  ( $1 \leq m \leq 10^5$ ).
- $m$  further lines, with the  $i$ th line ( $1 \leq i \leq m$ ) consisting of an integer indicating whether the  $i$ th member is short (0) or tall (1), then the number of the  $i$ th member's mentor,  $t_i$  ( $0 \leq t_i \leq m$ ). When  $t_i = i$ , this indicates that the  $i$ th member did not have a mentor.

### Output

If an arrangement is possible, output 2 lines of  $\frac{n}{2}$  numbers each to show which member should stand where.

Every number of type 1 should occur somewhere on the first row, and every number of type 0 should occur somewhere on the second row. Nobody should share a column with their mentor.

Otherwise, output *impossible*.

### Examples

standard input	standard output
4 0 1 1 1 1 2 0 3	3 2 1 4
4 0 1 1 1 0 1 1 1	<i>impossible</i>
10 0 1 1 1 1 1 1 1 0 1 0 4 0 6 1 1 0 7 1 2	10 8 3 2 4 1 6 7 9 5

## Problem F. Formula-1

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Those of you, who watch Formula-1 races, know that for several seasons Acronis supports one of teams: they started with Toro Rosso and now they are IT partners of Williams team.

Traditionally, the information for the drivers is shown by teams using special boards with big letters on it. Usually its time gap to the next (or previous) driver. Sure, this information is sent to the onboard electronics by the radio channel... but sometimes radio does not work properly.

Farid, one of the engineers of Williams come to Acronis with brilliant idea: let in this cases the video recognition software does all the work and reads the data from the tables.

Important part of this module is the recognition of the single letters. Your task is to implement it.

### Input

First line of the input contains one integer  $n$  ( $1 \leq n \leq 100$ ) — number of the possible letters.

Then the  $n$  letter descriptions come. Description of  $i$ -th letter starts with the two integers  $h_i$  and  $w_i$  ( $1 \leq h_i, w_i \leq 200$ ) — numbers of rows and columns in the letter shape. Each of the next  $h_i$  lines contains exactly  $w_i$  characters. '.' means that pixel is white, while 'X' means that pixel is black. It is guaranteed that each letter is 4-connected, i.e. between any two 'X' pixels exists a path consisting entirely from 'X' pixels where two consecutive pixels share a side, and that any two letters are pairwise distinct and cannot be transformed one to another by a rotation of multiple of 90 degrees.

Then come description of the picture of the board with message. The first line of the description contains two integers  $h$  and  $w$  — number of rows and columns of the picture. Each of the next  $h$  lines contains exactly  $w$  characters '.' and 'X'. You may assume that:

- Each 'X' pixel in the messages are parts of exactly one letter.
- Letters does not overlap neither touch by sides (but they may touch diagonally).
- All the letters in the message are the letters described in the previous section; they may be rotated by 0, 90, 180 or 270 degrees.

### Output

For each letter in order they are listed in the input data, print number of its occurrences in the message.

## Examples

standard input	standard output
<pre> 2 5 8 ...XXXX ...XX.XX ..XX..XX .XXXXXXX XX....XX 5 6 XXXXXX ..XX.. ..XX.. ..XX.. ..XX.. 11 13 .....X.... .....X.... .....XXXXX .....XXXXX .....X.... .....X.... ...XXXX.... ...XX.XX.... ..XX..XX.... .XXXXXXX.... XX....XX.... </pre>	<pre> 1 1 </pre>
<pre> 2 3 1 X X X 5 7 .XXXXX. XX...XX X.....X XX...XX .XXXXX. 5 7 .XXXXX. XX...XX X.XXX.X XX...XX .XXXXX. </pre>	<pre> 1 1 </pre>

## Problem G. Generator

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Gregor is an engineer. He now is working on the project of new generator for water power plants. One of important parts of generators is a water wheel.

A water wheel is a device for converting the potential energy of water into the kinetic energy of rotation. We have attached buckets along our otherwise perfectly-balanced wheel. Each bucket has a specific weight when empty, and a specific weight when full of water, both in kilograms.

Whenever a bucket reaches the top of the water wheel, it is filled to its maximum capacity. When the bucket reaches the bottom, diametrically opposite, it is emptied again.

The acceleration of the wheel depends on the horizontal distance between the centre of mass of the wheel and its axis of rotation. Centre of mass of the wheel at some time is defined as the sum of the co-ordinates of the buckets at that time multiplied by their weight.

Our water wheel has a radius of exactly 1 metre.

To calculate effectivity of his construction Gregor wants to know maximum positive  $x$ -component of centre of mass achieved by the wheel at any angle. Help him to find it.

### Input

- The first line of input contains the number of buckets on the wheel,  $n$  ( $2 \leq n \leq 10^5$ ).
- The remaining  $n$  lines of input each contain a description of a bucket, in angular order:
  - the decimal clockwise angle of the  $i$ th bucket in degrees,  $d_i$  ( $0.0 \leq d \leq 360.0$ ),
  - the decimal weights of this bucket when empty and full,  $e_i$  and  $f_i$  ( $0 \leq e_i \leq f_i \leq 10^4$ ).

### Output

Output the maximum horizontal component of the centre of mass of the wheel over all possible angles. Your answer should be within an absolute or relative error of at most  $10^{-6}$ .

### Examples

standard input	standard output
9 0.000 0.5 1.8 22.50 1.0 1.7 90.00 0.5 1.0 115.0 1.0 1.2 135.0 1.0 1.2 180.0 1.0 1.2 225.0 1.0 1.2 270.0 1.0 1.2 295.0 1.0 1.2	1.65664252966349700991

### Note

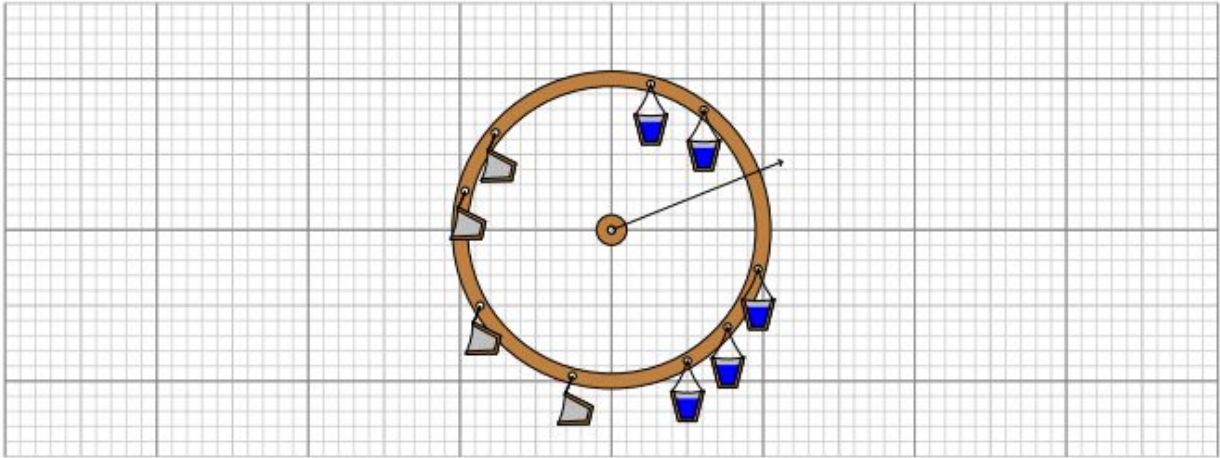


Illustration of Sample Input 1 at approximately 15.0 degrees clockwise from the start. The centre of mass is drawn as a vector from the origin.

## Problem H. Hacker's Heist

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Hatiko is a skilled hacker and works for the Byteland Intelligence Service.

She tricked her way into the server of Berland embassy through their local WiFi and managed to copy the files the informant told her about. Unluckily, the Berland authorities caught the informant, so they know what Hatiko did and sealed off the city. Hatiko needs to upload the data to the internet before they catch her.

Luckily the city is full of coffee shops with free wifi and Hatiko researched their details before she started her heist. The only downside is that they have strange opening hours and she may need to move between them. When travelling between coffee shops Hatiko can't upload any data at all.

Given the list of coffee shops, their opening hours and Wifi speed, what is the earliest time when all of important data can be uploaded?

### Input

- The first line contains an integer  $d$  ( $1 \leq d \leq 10^9$ ), the size of the data in megabytes.
- The second line contains an integer  $n$  ( $1 \leq n \leq 100$ ), the number of cafes.
- The next  $n$  lines each contain:
  - The integer opening and closing times of the cafe in seconds,  $o$  and  $c$  ( $0 \leq o \leq c \leq 24 * 60 * 60$ ).
  - The wifi speed of the cafe  $w$  ( $1 \leq w \leq 1000$ ) in megabytes per second.
- $n$  lines, each with  $n$  integers, the time to travel to each cafe from the  $n^{th}$  cafe, in seconds ( $0 \leq d_i \leq 24*60*60$ ).

Hatiko may start in any single cafe when it opens and start using wifi the moment she arrives at a cafe.

### Output

One line containing an integer number of seconds, the smallest integer time by which all of the data can be uploaded.

### Examples

standard input	standard output
200 2 10 20 15 10 40 5 0 5 5 0	35

## Problem I. Interesting Game

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Igor and Ilona, both enjoying playing games very much, have discovered a new type of game: tree game. In this game, first player chooses a vertex of a tree. Then players (beginning with second player) alternately choose a neighbor of the last chosen vertex that was not chosen before, until a player can't make a move. This player is then declared loser and the other one is the winner. Igor moves first but unfortunately Ilona is a very experienced player and she will never make a mistake. Therefore Igor has asked you for help.

Our tree has  $N$  vertices conventionally numbered  $1 \dots N$  and exactly  $N - 1$  edges connecting them. Write a program, which determines all vertices in which Igor can start the game and win despite Ilona playing perfectly.

### Input

On the first line, there is a single number  $N$ , ( $1 \leq N \leq 2 \cdot 10^6$ ), which is equal to the number of nodes in the tree.  $N - 1$  lines follow, on the  $i$ -th line is a single integer  $a_i$  ( $1 \leq a_i \leq i$ ), which means there is an edge in the tree connecting the vertex  $i + 1$  with the vertex  $a_i$ .

### Output

Output consists of several lines, on each one there is a single number of a node, where Igor can start the game and win, regardless of how Ilona would play. Numbers in the output should be sorted in ascending order.

## Problem J. Jackpot

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

The thunderingly fun new game show of 'Jackpot' has bounded onto televisio screens up and down the land and is the next big thing. It runs along a simple premise: there are a number of doors, behind one of which is hidden a monetary prize and behind all of the others of which are hidden goats. Each contestant chooses how many doors will be opened before they pick a door to look at. The prize decreases as each door is opened. If they manage to open the door and find the money they get to keep it!

Not content with guessing, Jessica decided to try and work out the best number of doors to open to maximise her expected profit, where expected profit is the probability of winning multiplied by the remaining prize fund.

Jessica managed to find out that the remaining prize fund  $P_R$  is a function of the initial prize fund  $m$ , the number of doors opened  $d$  and a scaling factor  $f$ , ie.

$$P_R = m - (d \cdot f)^2$$

Given the number of doors, initial prize fund and the scaling factor can you work out how many doors should be opened before Jessica will try to pick a door to maximise her expected profit?

### Input

One line of input containing:

- One integer  $n$  ( $1 \leq n \leq 10^{18}$ ), the number of doors.
- One integer  $m$  ( $1 \leq m \leq 10^{18}$ ) the initial prize fund.
- One floating point  $f$  ( $0 < f \leq 1$ ), the scaling factor.

There will always only be one door with the best value.

### Output

One line containing one floating point number, the value of the expected payout. The output must be accurate to an absolute or relative error of at most  $10^{-6}$ .

### Example

standard input	standard output
4 100 1	91.0
20000 100500 0.8	5.0254931



## Problem K. Kings

Input file: *standard input*  
Output file: *standard output*  
Time limit: 2 seconds  
Memory limit: 512 mebibytes

Kevin invented the new game, called “chess-tidying”. The game starts with a messy board containing several king pieces strewn across the cells. The player’s job is to put all of the pieces into a line along the primary diagonal of the board, which runs along black squares.

In each move, unlike in normal chess, one king may be moved one place either horizontally or vertically (but not both) to another unoccupied cell.

Kevin wants, given an instance of the game, to know how many moves you will need in order to finish it. Help him to do that

### Input

- One line with the number of rows and columns,  $n$  ( $1 \leq n \leq 500$ ).
- Each of the following  $n$  lines contains the two-dimensional integer coordinates  $c$  and  $r$  ( $1 \leq c, r \leq n$ ), the position of one of the kings.

Each of the kings starts at a unique position.

### Output

Output the minimum number of moves necessary to cover the main diagonal ( $r = c$ ) with kings.

### Examples

standard input	standard output
3 1 1 2 3 3 2	2
8 6 4 6 8 5 5 5 4 4 8 5 7 7 4 3 7	28

### Note

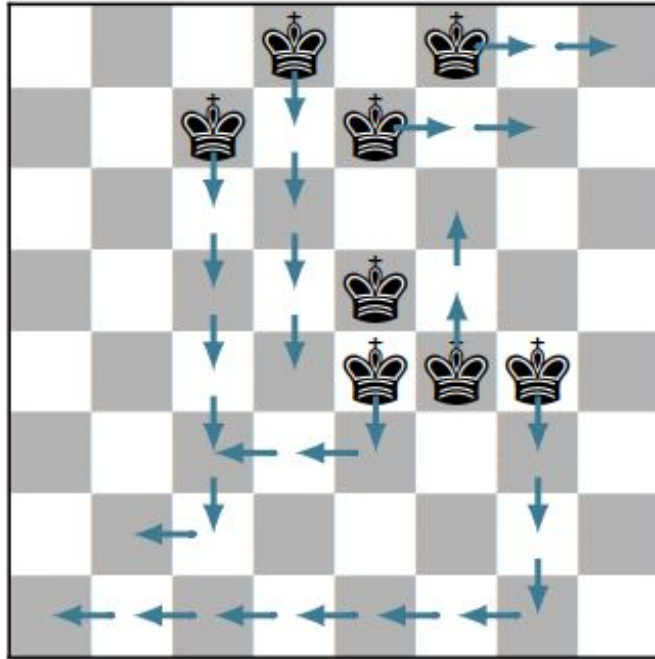


Illustration of Sample Input 2. In this case, the minimum number of moves necessary to put all of the kings along the black diagonal is 28 as pictured.

## 7. Референсы

- ICPC Foundation <http://nac.icpc.global/>
- ICPC by Baylor University ICPC Headquarter <https://icpc.baylor.edu/>
- Spradley, James P. (1980). Participant observation. The USA: Holt, Rinehart and Winston
- The Future of Jobs Report 2018. Switzerland: World Economic Forum, 2018
- Интервью с Олегом Христенко и Владиславом Невструевым, Москва, Долгопрудный, 19 и 20 февраля 2020 г.
- Анастасия Рылова. Top-10 soft skills, и почему их стоит в себе развивать? TEDxGorkyLibrary, 31 августа 2018  
<https://www.youtube.com/watch?v=zisPBjvT7LQ>
- Сергей Зуев. Навыки для глобального мира будущего, которым не учат в школах. TEDxYouth@Vladivostok, 13 ноября 2017  
<https://www.youtube.com/watch?v=cIClkaxKhno>
- Правила проведения отборочных этапов ICPC  
<https://neerc.ifmo.ru/information/selection-rules.html>
- Сайты для тренировки решения задач по спортивному программированию  
<https://tproger.ru/experts/8/>
- Важные ссылки и сервисы по олимпиадному программированию  
<https://codeforces.com/blog/entry/512?locale=ru>
- Тренировочные сборы Moscow Workshops и Discover  
<https://discover.it-edu.com/>
- Интервью с Андреем Станкевичем про спортивное программирование  
<https://habr.com/ru/post/446658/>
- Статистика ICPC <https://icpc.baylor.edu/worldfinals/pdf/Factsheet.pdf>