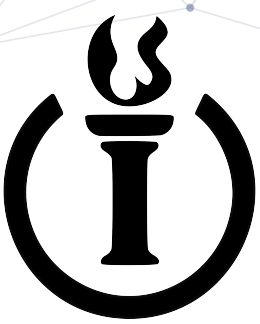
International Computer Science Competition

Лист задач отборочного раунда



Задачи

Задачи ICSC состоят из концептуальных задач и задач по программированию. Тип задачи обозначается следующими символами:

- 💡 Концептуальная задача: Требует логического или математического мышления, решается вручную.
- </> Задача по программированию: Требует реализации решения в коде.

Всего ты можешь набрать до 25 баллов. Уровень сложности варьируется от одной звезды (самый простой) до трех звезд (самый сложный).

Мы подготовили обучающие материалы, которые помогут тебе с концепциями, относящимися к предложенным задачам. Ты можешь получить к ним доступ здесь: https://icscompetition.org/training.

Твое решение

Ты должен предоставить письменное описание своего решения для **всех задач** в документе с решением, который ты отправляешь. Обязательно объясни свое решение, включая любой написанный тобой код. Ты можешь предоставить рукописные или напечатанные решения.

Для задач по программированию ты также можешь по желанию загрузить свои файлы с кодом, чтобы получить специальную почетную отметку в своем сертификате. Ты можешь предоставить свои решения по программированию на Python (3.9), Java (OpenJDK 11), C (C99) или C++ (GCC 15.1).

Информация о подаче

Когда ты отправишь свое решение, автоматически будет создана учетная запись участника, чтобы помочь тебе управлять своей заявкой. Ты можешь получить доступ к своей учетной записи и заявке, используя учетные данные, которые ты установил во время процесса подачи: https://icscompetition.org/login.

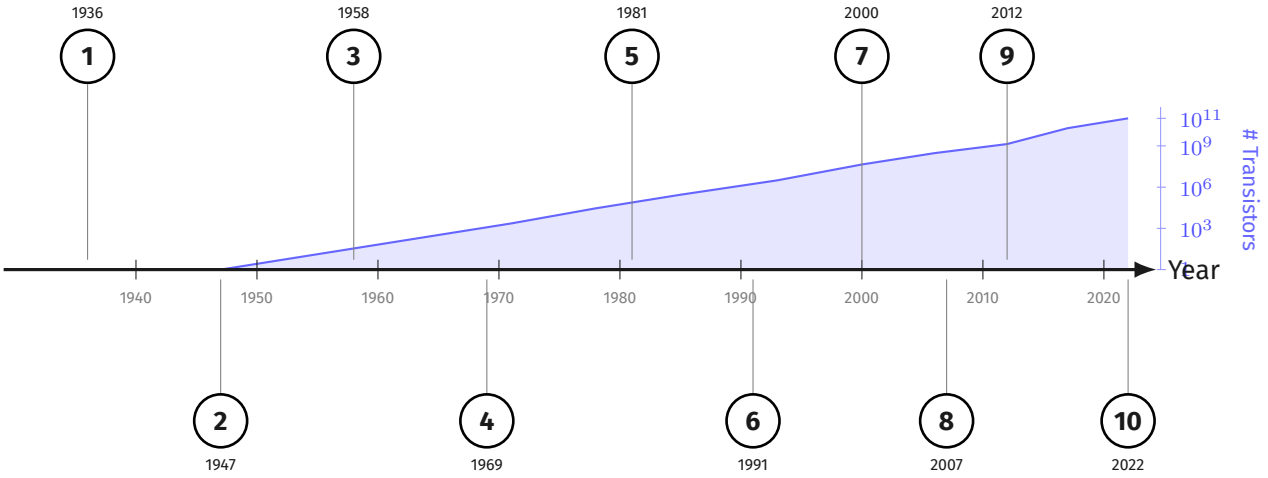
Пожалуйста, убедись, что твои файлы с решением и кодом загружены правильно. Любые проблемы с выполнением твоего (необязательного) кода будут отмечены в твоей учетной записи участника. Ты должен отправить свое решение онлайн до *Sunday, 5 July 2026, 23:59 UTC+0* по адресу https://icscompetition.org/submission.

Чтобы пройти в Pre-Final Round, ты должен набрать не менее 12/14/17 баллов. Если у тебя есть вопросы или комментарии, не стесняйся обращаться к нам по электронной почте: info@icscompetition.org.

Надеемся, тебе понравится решать задачи. Удачи!

Problem A [5 points]

Ниже представлена временная шкала, на которой отмечены десять знаковых событий в истории вычислительной техники и технологий, сопоставленных с ростом количества транзисторов в коммерческом микропроцессоре.[1]



Определи и назови каждое пронумерованное событие, отмеченное на временной шкале выше (может быть несколько правильных ответов):

1. (1936): .....

2. (1947): .....

3. (1958): .....

4. (1969): .....

5. (1981): .....

6. (1991): .....

7. (2000): .....

8. (2007): .....

9. (2012): .....

10. (2022): .....

[1] Транзистор — это переключатель или усилитель, который регулирует поток тока в цепи, что делает его фундаментальным строительным блоком всех современных компьютеров и электроники.

Problem B [5 points]

Ты можешь скачать шаблон кода для этой задачи здесь.

Pixel art — это вид цифрового искусства, где изображения создаются путем заполнения отдельных ячеек в сетке. Твоя задача — написать функцию, которая генерирует определенные геометрические узоры на сетке $N \times N$, где каждая ячейка либо 1 (заполнена), либо 0 (пуста).

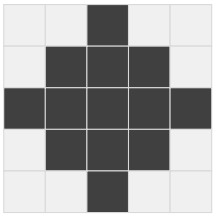
Тебе нужно реализовать следующие две фигуры:

1. Checkerboard (`"checkerboard"`): Ячейки чередуются между заполненными и пустыми, как на шахматной доске. Верхняя левая ячейка всегда 0. Пример для $N = 4$:

```
0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0
```

2. Diamond (`"diamond"`): Фигура в форме ромба, центрированная в сетке. Для этой фигуры N всегда должно быть нечетным. Пример для $N = 5$:

```
0 0 1 0 0
0 1 1 1 0
1 1 1 1 1
0 1 1 1 0
0 0 1 0 0
```



Чтобы реализовать эти фигуры, напиши следующую функцию:

```
list<list<int>> generate_shape(int n, string shape)
```

- n: Размер сетки ($N \times N$ сетка, $5 \leq N \leq 51$, для фигуры diamond всегда нечетное).
- shape: Либо `"checkerboard"`, либо `"diamond"`.
- Функция должна возвращать двумерный список/массив целых чисел (0 или 1).

Формат вывода: Выведи сетку в виде N строк, каждая из которых содержит N значений, разделенных пробелом (0 или 1).

Problem C [5 points]

В задаче A ты мог заметить, что количество транзисторов в коммерческом микропроцессоре растет не линейно. Еще в 1975 году Гордон Мур заявил, что количество транзисторов удваивается примерно каждые **2 года**; эта тенденция сохраняется уже более пяти десятилетий.

Это дает следующую формулу:

$$T(t) = T_0 \times 2^{t/2}$$

где T_0 — количество транзисторов в базовый год, а t — количество лет, прошедших с этого базового года.

Intel 4004 был выпущен в 1971 году как первый в мире коммерческий одночиповый микропроцессор с количеством транзисторов $2{,}300$.

(a) Представь, что ты в 1971 году. Используя формулу выше, предскажи количество транзисторов в микропроцессоре в 2026 году.

(b) Сравни свой прогноз с реальным процессором 2026 года, например, Apple M5. Насколько хорошо закон Мура подтверждается на протяжении 55 лет?

(c) Учитывая реальное количество транзисторов в 2026 году, которое ты нашел в пункте (b), какой период удвоения (в годах) лучше всего соответствует росту от Intel 4004 до сегодняшнего дня?

Problem D [5 points]

Ты можешь скачать заготовку кода для этой задачи здесь.

Ты работаешь над сервером для онлайн-файтинга в 2D. Два игрока на разных ПК отправляют свои действия через интернет на центральный сервер каждый кадр (1/60 секунды).

Идет Grand Finals. Матч заканчивается победой Player 1. Player 2 в ярости и утверждает, что его решающий удар пришелся ровно в тот же кадр, что и удар Player 1, а значит, это должен был быть двойной KO. Согласно правилам игры, все события в одном и том же кадре должны быть применены до проверки на KO. Чтобы разрешить спор, ты заново проигрываешь всю игру через функцию `processGame`, чтобы пересчитать итоговое HP, и подозреваешь, что в ней есть баг. Вот её псевдокод:

processGame

Input: events: список (player, frame, attack_value) в произвольном порядке (из-за сетевых задержек)
Output: player1_hp, player2_hp

```
1: stable sort events by frame number
2: for all event in events do
3:     if player1_hp ≤ 0  OR  player2_hp ≤ 0 then
4:         break
5:     end if
6:     opponent_hp ← opponent_hp - event.attack_value
7: end for
8: return player1_hp, player2_hp
```

Более того, логи событий боя показывают:

- `(1, 512, 20)`
- `(2, 512, 20)`

при `player1_hp = 10` и `player2_hp = 15` к моменту, когда повтор доходит до 512 кадра.

(a) Что не так с вышеуказанным псевдокодом? В чем именно заключается проблема?

(b) Реализуй исправленную функцию `processGame` на C, C++, Python или Java. Она принимает полный лог событий одного матча, устанавливает обоим игрокам начальное HP равное H, проигрывает каждое событие в порядке кадров и возвращает итоговое HP каждого игрока.

```
list<int> processGame(list<tuple<int,int,int>> events, int H)
```

- `events`: Список кортежей `(player, frame, attack_value)`, где `player` — это 1 или 2, `frame` — неотрицательное целое число, а `attack_value` — положительное целое число.
- H: Начальное HP для обоих игроков (положительное целое число).
- Верни список из двух целых чисел: `[hp1, hp2]`, каждое из которых не может быть меньше 0.

Problem E [5 points]

В 1998 году PageRank вывел Google вперед всех других поисковых систем. Идея: страница важна, если на нее ссылаются важные страницы. Формально, PageRank страницы p_i равен:

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

где $M(p_i)$ — это множество страниц, которые ссылаются *на* p_i, $L(p_j)$ — это количество исходящих ссылок со страницы p_j (ее исходящая степень), N — это общее количество страниц, а $d = 0.85$ — это **коэффициент затухания**.

Сегодня "мусор", сгенерированный ИИ, наводняет интернет. Когда эти страницы смогут захватить поисковый алгоритм?[2]

(a) Объясни, как работает PageRank: интерпретация случайного серфера, роль d, почему уравнение рекурсивно и как оно решается на практике.

(b) Рассмотрим N страниц, разделенных на h страниц, созданных человеком, и $a = N - h$ страниц, созданных враждебным ИИ. Страницы, созданные человеком, ссылаются на k страниц, выбранных равномерно случайным образом из всех N страниц. Страницы ИИ ссылаются **только** на другие страницы ИИ. Пусть $\alpha = a/N$.

Выведи выражение в замкнутой форме для $S_A = \sum_{p \in A} PR(p)$, общего PageRank всех страниц ИИ A, в терминах α и d.

[2] Google и другие современные поисковые системы сегодня используют сотни сигналов для ранжирования страниц. Наш фиктивный сценарий фокусируется на поисковой системе, которая использует исключительно PageRank, и, таким образом, предполагает упрощенный сценарий.

(c) Найди переломную долю α^*, при которой $S_A = S_H$ (т.е. страницы ИИ в совокупности имеют столько же PageRank, сколько страницы, созданные человеком). Затем обсуди переломную долю при $d = 0.85$ и объясни проблемы PageRank, которые раскрывает наш сценарий.