Внимание: Согласно правилам олимпиады, каждая команда может использовать только **один** компьютер. Вы можете использовать дополнительные компьютеры **только** для связи с сокомандниками и для чтения условий задач.

В любой момент времени может быть только один активный компьютер, на котором вы можете писать, компилировать, запускать программы. Активный компьютер может переключаться во время тура, например чтобы следующую задачу писал другой член команды. Однако нельзя проявлять активность больше чем на одном компьютере одновременно.

Использовать интернет разрешается.

Использовать предварительно написанный код разрешается.

Погрузка

1 Условие

Есть *склад* диковинных товаров, собранных исследователями. План склада представляется прямоугольным клеточным полем M на N клеток. Склад обнесён глухим забором, так что выходить за пределы поля нельзя. В P заданных клетках поля находятся двусторонние порталы в *другой мир*. Во всех остальных клетках поля изначально лежит по одному ящику с грузом.

Изначально в другом мире находится K погрузочных челноков. Челноки проникают на склад через порталы, чтобы транспортировать ящики со склада. Ваша задача — составить план действий для челноков, позволяющий перенести все $(MN - P)$ ящиков со склада в другой мир.

Время и пространство в задаче дискретны.

Если челнок находится на складе в клетке с порталом, он может воспользоваться этим порталом для перемещения в другой мир, потратив на это единицу времени. Когда челнок попадает в другой мир, он мгновенно освобождается от груза и готов к следующей поездке. Если челнок находится в другом мире, то он может воспользоваться любым порталом по своему выбору для перемещения на склад, потратив на это единицу времени. В результате такого перемещения челнок оказывается в той клетке склада, где расположен соответствующий портал. Каждый портал пропускает в единицу времени максимум один челнок либо туда, либо обратно.

Находясь на складе, за единицу времени челнок может сделать один шаг в соседнюю клетку по горизонтали или вертикали, или может никуда не ходить и остаться на месте. Если свободный челнок шагает в клетку с ящиком, то он автоматически берёт этот ящик себе на голову. Далее челнок должен пронести ящик в другой мир через какой-нибудь портал исключительно аккуратно, нигде не опуская. Если челнок уже несёт ящик, то шагать в клетку с другим ящиком ему запрещено.

Каждую единицу времени все K челноков выполняют свои действия одновременно.

Однако мир нестабилен, поэтому челнокам нельзя сталкиваться. Ни в какой момент времени два челнока не могут находиться в одной клетке склада. Кроме того, встречное движение двух челноков в одну единицу времени также запрещено: то есть челноки не могут за один ход поменяться местами, находясь в смежных клетках. С другой стороны, поскольку все челноки ходят одновременно, то челнок может шагнуть в клетку с другим челноком, если другой челнок в этот момент уходит из этой клетки.

Рассчитайте план движения челноков, чтобы доставить все ящики со склада в другой мир за минимальное время.

2 Формат входных данных

На вход вашей программе в первой строке подаются четыре целых числа — размеры склада M, N, количество челноков K и количество порталов P, открытых на складе ($1 \leqslant M, N \leqslant 100, 1 \leqslant K \leqslant 100, 1 \leqslant P \leqslant 10$).

В следующих P строках заданы координаты порталов r_i, c_i — номер строки и столбца, в которых находится i-ый портал ($0 \leqslant r_i < M, 0 \leqslant c_i < N, 0 \leqslant i < P$).

Порталы нумеруются от 0 до $P-1$ в порядке их описания. Гарантируется, что все порталы находятся в разных клетках.

3 Формат выходных данных

В первую строку выходного файла необходимо вывести целое число T — сколько единиц времени/шагов в вашем плане. Каждая из следующих K строк должна содержать T символов, определяющих действия соответствующего челнока в каждый момент времени. Каждое действие челноков описывается одним символом:

- L — челнок совершил шаг влево от своей предыдущей позиции;
- R — челнок совершил шаг вправо от своей предыдущей позиции;
- U — челнок совершил шаг вверх от своей предыдущей позиции;
- D — челнок совершил шаг вниз от своей предыдущей позиции;
- W — челнок остался на текущей клетке и ничего не сделал;
- *ind* (цифра) — челнок вошел на склад через портал с номером *ind* ($0 \leqslant ind \leqslant P-1$);
- E — челнок воспользовался порталом, чтобы покинуть склад.

Регистр букв (большая или маленькая) значения не имеет. Вместо символа `W` разрешается выводить символ точки. Поле ориентировано таким образом, что шаг вниз увеличивает номер строки, а шаг вправо увеличивает номер столбца.

Максимально допустимая длительность плана ограничена. Суммарное количество действий, совершенных всеми челноками, равное ($T \cdot K$), не должно превышать 10^7.

Если план некорректен по любой причине, то решение получает за тест вердикт `WA`.

4 Пример

input.txt	output.txt
2 2 2 1 0 0	10 ORLEORDULE WODWWUEWWW

Вы можете посмотреть визуализацию этого плана действий, если вы откроете визуализатор из архива материалов сразу после скачивания.

5 Ограничения

Ограничение по времени составляет **2 секунды**.

Ограничение по памяти составляет **256 мегабайт**.

6 Отправка решений в систему тестирования

Участник может отправлять своё решение в систему тестирования NSUts во время тура. В системе решение тестируется на наборе из 100 тестов.

За каждый тест решение получает один из следующих вердиктов:

- `A: Accepted` — решение успешно прошло тест, все ящики вынесены со склада.
- `P: Presentation Error` — решение вывело некорректный план действий.
- `W: Wrong Answer` — решение вывело некорректный план действий.
- `T: Time Limit Exceeded` — решение не уложилось в ограничение по процессорному времени.
- `D: Deadlock` — решение не уложилось в ограничение по астрономическому времени.
- `M: Memory limit exceeded` — решение не уложилось в отведённое ограничение по памяти.
- `R: Run-time error` — решение вернуло код ошибки, отличный от нуля (скорее всего “упало”).
- `S: Security violation` — решение совершило действие, запрещённое правилами.

Если решение получило вердикт `A: Accepted`, то оно получает за тест вещественное число баллов в диапазоне от 0 до 1. При любом другом вердикте решение получает за тест 0 баллов.

7 Система оценивания

Баллы решения за тест вычисляются по формуле:

$$Score = \frac{T_{best}}{T}$$

Здесь T — количество ходов в выведенном участником плане, а T_{best} — наименьшее количество ходов на данном тесте среди всех участников соревнования.

Баллы решения по всем тестам суммируются, и в результате получается вещественное число в диапазоне от 0 до 100. Команды упорядочиваются по убыванию этой суммы: первое место занимает команда, у которой сумма максимальна.

8 Просмотр результатов в системе тестирования

Список ваших посылок в систему тестирования NSUts доступен на вкладке «Результаты».

Рядом с каждой посылкой есть ссылка «Открыть отчёт о тестировании». Данная страница содержит полную информацию о результатах тестирования: вердикт, количество ходов в плане и текущее количество баллов по каждому тесту. Также по каждому тесту есть ссылка для запуска визуализатора, в котором можно посмотреть выведенный план действий. Описание визуализатора приведено в разделе 13.

Кроме того, рядом с каждой посылкой есть ссылка «Скачать результаты тестирования» для скачивания zip-архива. В этом архиве содержатся выходные данные по всем тестам, полученные при автоматическом запуске программы участника системой.

9 Предварительное и финальное тестирование

В течение тура вы можете отправлять в систему тестирования сколько угодно решений. Кроме того, вам доступен текущий предварительный рейтинг, результаты которого не идут в зачёт. Этот рейтинг строится по описанным выше правилам, при этом учитываются только результаты последнего на текущий момент скомпилировавшегося решения от каждого участника.

Набор тестов, на котором тестируются решения во время тура, полностью включён в архив материалов, который вы можете скачать. В архиве также есть генератор тестов и скрипт генерации, с помощью которых этот набор был создан. В скрипте генерации в качестве *seed* используется число 123.

После конца тура будет выполнено **финальное тестирование**. От каждого участника в финальное тестирование проходит одно решение — **последнее успешно скомпилированное решение**, отправленное в систему NSUts. По результатам работы решений будет построен финальный рейтинг по тем же самым правилам. Место каждого участника в финальном рейтинге определяет результаты участника в первой номинации и учитывается далее при суммировании номинаций для подведения итогов всей олимпиады.

Набор тестов в финальном тестировании **будет изменён**. Жюри поставит другое значение *seed* в скрипт генерации и перегенерирует все тесты. В результате перегенерации параметры M, N, K, P на каждом тесте **не** изменятся, но изменится размещение порталов.

10 Материалы

В новостях в системе тестирования доступна ссылка для скачивания архива с материалами (`materials.zip`).

Прежде всего рекомендуется:

1. Распаковать архив материалов, перейти в директорию с содержимым.
2. Перейти в папку `js_vis` и запустить визуализатор примера, открыв `index.html` в браузере.
3. Скомпилировать чекер `check.cpp`, который проверяет ответ на тесте.
4. Написать и скомпилировать свое решение, например, `Task.exe`
5. Запустить `python runner.py Task.exe` — выполняет локальное тестирование на 100 тестах в директории `tests`.
6. Запустить `python runner.py Task.exe -i 42` — то же самое, но запускает только один тест под номером 42.
7. Перейти в папку `js_vis` и открыть `index.html` в браузере, чтобы посмотреть выведенный план для 42-ого теста.
8. Отправить решение в систему тестирования.
9. Посмотреть результаты посылки в системе.

В архиве присутствуют:

Файл или директория	Описание содержимого
`runner.py`	Простой скрипт для локального прогона решения на всех тестах.
`check.cpp`	Проверяющая программа.
`gen_random.cpp`	Программа для генерации тестов.
`gen.cmd`	Скрипт для генерации тестов, которым создали набор, используемый на предварительном тестировании.
`tests/*.in`	Набор тестов для предварительного тестирования, полученные запуском `gen.cmd`.
`js-vis/script.js`, `js-vis/index.html`	Визуализатор входных/выходных данных теста.

`js-vis/res.js`	Файл, из которого визуализатор загружает входные/выходные данные.
`gameround.pdf`	Это условие задачи.
`testlib.h`	Служебный файл, используемый для компиляции генератора и чекера.
`check.exe`	Предсобранный исполняемый файл для `check.cpp`.
`gen_random.exe`	Предсобранный исполняемый файл для `gen_random.cpp`.

11 Чекер и генератор

Проверяющую программу и генератор можно скомпилировать примерно так:

```
cl check.cpp /O2 /EHsc
cl gen_random.cpp /O2 /EHsc

g++ check.cpp -O2 -o check
g++ gen_random.cpp -O2 -o gen_random

clang++ check.cpp -O2 -o check
clang++ gen_random.cpp -O2 -o gen_random
```

Проверяющая программа запускается с тремя параметрами:

```
check  input.txt output.txt output.txt
```

Первые два параметра — входной и выходной файл, а третий параметр должен указывать на любой файл (его содержимое не используется). Результаты проверки выводятся в stderr, то есть в консоль.

Генератор запускается с пятью параметрами:

```
gen_random  N M K P seed   >input.txt
```

Первые четыре параметра — это размеры теста N, M, K, P, описанные выше в условии. Последний параметр нужен для инициализации генератора случайных чисел. При повторных запусках с одинаковыми параметрами генератор должен выдавать одинаковые данные. Последний элемент `>input.txt` перенаправляет вывод в файл `input.txt` (без этого генератор пишет входные данные для теста в `stdout`, то есть в консоль).

12 Скрипт прогона тестов

Скрипт `runner.py` запускает заданное решение на тестах, которые лежат в директории `tests`.

Чтобы запустить решение `my_mega_sol.exe`, нужно выполнить команду:

```
python runner.py  my_mega_sol.exe
```

Чтобы запустить решение `my_mega_sol` на Linux, следует добавить dot-slash:

```
python runner.py  ./my_mega_sol
```

Запустить решение на Java с классом `my_java_sol` можно так:

```
python runner.py  "java my_java_sol"
```

Внутри кавычек можно указывать любые другие параметры, например `-Xmx256M` и подобные. Аналогично запускаются решения на Python и Kotlin.

Скрипт перебирает файлы `tests/k.in` для $k = 1 \dots 100$ и запускает на каждом тесте решение и чекер. Выходные данные сохраняются в файл `tests/k.out`. В конце работы скрипт выводит количество ходов по тестам в виде мини-таблицы: в первой строке показывается количество ходов для тестов $1 - 10$ по порядку, во второй — по тестам $11 - 20$, и так далее.

Кроме того, можно запустить решение на одном заданном тесте, добавив параметр `-iX`, например:

```
python runner.py  my_mega_sol.exe -i17
```

После этого можно посмотреть выведенный решением план в визуализаторе, открыв `js_vis/index.html` в браузере. Скрипт автоматически копирует входные и выходные данные в `js_vis/res.js` каждый раз, когда он запускает решение.

13 Визуализатор

С помощью визуализатора можно посмотреть план действий челноков, которые решение вывело в выходной файл. Чтобы запустить визуализатор, нужно открыть файл `js_vis/index.html` в браузере, предварительно вставив входные и выходные данные в файл `js_vis/res.js`. Вы можете копировать данные вручную, либо использовать `runner.py` с указанием номера теста.

Визуализатор показывает поле следующим образом. Фон клетки серый, если в ней есть портал, и белый, если нет. Если в клетке есть челнок, то он изображён цветным прямоугольником. Каждый челнок окрашен своим цветом и пронумерован. Чёрная рамка в клетке показывает, что в ней есть ящик, либо что челнок в этой клетке несёт ящик. При наведении курсора на клетку склада отображаются ее координаты (нумерация с нуля), сообщение о наличии ящика и челнока в ней.

В нижней части визуализатора находится строка статуса, которая показывает, сколько ходов уже прошло, сколько ящиков остаётся на складе, действия роботов на последнем ходу. Кроме того, там же находится одна или две полосы прокрутки, с помощью которых можно изменять номер текущего хода. Также работают клавиши: стрелки влево/вправо, клавишы Home и End. Клавиша Space запускает автоматическое проигрывание плана. Чтобы перейти к заданному моменту времени, достаточно подправить номер хода в адресной строке браузера.