

До окончания тестирования осталось 6 дней 8 часов 19 минут

# Задача 1. Розыгрыш резюме рьяными работниками

## Условие задачи

Ограничение времени, с	1
Ограничение памяти, МБ	64
Общее число попыток отправки	15

У HR Маши на столе лежат две стопки резюме, размерами  $n$  и  $m$ , в каждом из резюме указана зарплата, числа  $a[0..n-1]$  для одной стопки, и  $b[0..m-1]$  для второй. Нулевой индекс указывает на верхнее резюме в стопке.

Маша устанавливает значение  $s$  максимальной суммы зарплат и предлагает очень активному стажеру Саше сыграть в игру:

- За каждый ход Саша может взять одно *верхнее* резюме из любой стопки и забрать себе в работу
- Саша считает сумму всех зарплат из резюме, которые он взял. Он может брать новые резюме из стопок только таким образом, чтобы эта сумма не превышала  $s$
- Игра заканчивается, если Саша больше не может брать резюме

Нужно выяснить, какое максимальное количество резюме Саша мог бы забрать себе в работу, если бы тоже знал зарплату, указанные в каждом резюме.

## Входные данные (поступают в стандартный поток ввода)

Первая строка – целые числа  $n$ ,  $m$  и  $s$  через пробел ( $1 \leq n \leq 10\,000$ ,  $1 \leq m \leq 10\,000$ ,  $1 \leq s \leq 200\,000\,000$ )

Далее идут строки с зарплатами резюме в стопках. Всего строк столько, сколько резюме в большей из стопок, на каждой строке один из вариантов:

- два целых числа  $a$  и  $b$  через пробел ( $1 \leq a \leq 10\,000$ ,  $1 \leq b \leq 10\,000$ ),
- $a$  и символ – (если во второй стопке больше нет резюме) через пробел ( $1 \leq a \leq 10\,000$ )
- символ – (если в первой стопке больше нет резюме) и  $b$  через пробел ( $1 \leq b \leq 10\,000$ )

Все входные данные наших тестов всегда соблюдают указанные параметры, дополнительные проверки не требуются

## Выходные данные (ожидаются в стандартном потоке вывода)

Одно целое число, максимальное количество резюме

### Пример 1

Ввод:

```
3 4 11
1 1
2 2
3 3
- 4
```

Вывод:

```
5
```

Оптимальным алгоритмом здесь будет просто брать верхние резюме из каждой стопки  $1 + 1 + 2 + 2 + 3 = 9$ . Дальше резюме брать нельзя, потому что сумма станет выше 11, поэтому возвращаем 5.

### Пример 2

Ввод:

```
5 5 10
5 5 10
5 1
```

```
1 3
1 3
1 3
1 3
```

Вывод:

6

Здесь ситуация интереснее, и играет роль то, что Саша знает все зарплаты во всех резюме, оптимально для него будет взять сначала всю левую стопку по порядку  $5 + 1 + 1 + 1 + 1 = 9$ , а потом взять еще верхнее резюме из правой  $9 + 1 = 10$ . Итого 6 резюме.

### Пример 3

Ввод:

```
6 4 10
4 2
2 1
4 8
6 5
1 -
7 -
```

Вывод:

4

Этот пример похож на первый, просто показывает, как выглядит ввод для ситуации, когда вторая стопка меньше первой

## Примечания по оформлению решения

Возможно использование только стандартных библиотек языков, установки и использование дополнительных библиотек невозможны.

При отправке решений на Java необходимо назвать исполняемый класс `Main`. В решении не нужно указывать пакет.

### Примеры работы со стандартными потоками ввода и вывода

Для JS можно использовать `readline` и `console.log`:

```
const readline = require('readline').createInterface(process.stdin, process.stdout);
readline.on('line', (line) => {
  // Введенная строка в переменной line, тут можно написать решение и вывести его с помощью console.log
  ...
  console.log(String(result));
  readline.close();
}).on('close', () => process.exit(0));
```

в Python можно использовать встроенные функции `input()` и `print()`:

```
line = input()
...
print(result)
```

в Java можно использовать `java.util.Scanner` и `System.out.println`:

```
Scanner in = new Scanner(System.in);
String line = in.nextLine();
...
System.out.println(result);
```

Перед отправкой решения рекомендуем запустить тесты из раздела *Тестирование*, они помогут поймать синтаксические ошибки и ошибки выполнения.