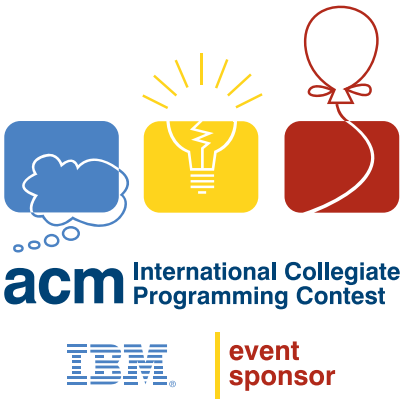
The German Collegiate Programming Contest 2016

Practice Session

GCPC Practice 2016



Problems

A Hello, GCPC!
B The Buffet Wars
C Packing Cases

Do not open before the contest has started.

This page is intentionally left blank.

Problem A
Hello, GCPC!

Alice is the new General Code Performance Caretaker (GCPC) at her company. Her job is to check the code written by the developers for bad performance and improve it if possible. The code is often strange since a company-wide policy is to write every piece of code in three programming languages. This is to ensure that the code runs on as many platforms as possible although Alice doubts that this is the best way to do it. Either way, she found the following code written in C++, Java, and Python and wants to improve it.

C++:

```cpp
#include <algorithm>
#include <iostream>
#include <vector>

using namespace std;

int main() {
  int n;
  cin >> n;

  vector<long long> a(n);
  for(int i = 0; i < n; i++) cin >> a[i];

  vector<long long> p;
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
      for(int k = 0; k < n; k++) {
        p.push_back(a[i]*a[j]*a[k]);
      }
    }
  }

  sort(p.begin(), p.end());

  cout << p[p.size()-1] << endl;
  return 0;
}
```

Java:

```java
import java.util.Collections;
import java.util.LinkedList;
import java.util.Scanner;

public class HelloGCPC {
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);

```

```
9       int n= s.nextInt();
10
11      long[] a = new long[n];
12      for(int i = 0; i < n; i++) a[i] = s.nextLong();
13
14      LinkedList<Long> p = new LinkedList<>();
15      for(int i = 0; i < n; i++) {
16        for(int j = 0; j < n; j++) {
17          for(int k = 0; k < n; k++) {
18            p.add(a[i]*a[j]*a[k]);
19          }
20        }
21      }
22
23      Collections.sort(p);
24
25      System.out.println(p.get(p.size()-1));
26    }
27  }
```

Python:

```
1  n = int(raw_input())
2  a = map(long, raw_input().split())
3
4  p = []
5  for i in range(n):
6    for j in range(n):
7      for k in range(n):
8        p.append(a[i]*a[j]*a[k])
9  p.sort()
10
11 print p[len(p)-1]
```

Can you help her rewrite the code in one of the languages such that it prints the same result for all inputs but always runs within a second?

Input

The input consists of:

- one line with an integer n ($1 \leq n \leq 10^4$), where n is the number of integers read by the code;
- one line with n integers $a_1, \ldots, a_n$ ($0 \leq a_i \leq 10^6$ for all $1 \leq i \leq n$), where $a_1, \ldots, a_n$ are the numbers used by the code above.

Output

Output the same as the code above does.

Sample Input 1	Sample Output 1
3 1 2 3	27

Sample Input 2	Sample Output 2
5 344 823 3950 167 4784	109489762304

This page is intentionally left blank.

Problem B
The Buffet Wars

Today is the yearly grand gala ball for cooks - the Global Congress for Phenomenal Cooking (GCPC), where many important people give long speeches about the importance of Cooking to modern society until finally, the festive part begins. As expected, all guests storm the buffet and begin fighting over the most precious dishes with forks in hand.

Last year, Herbert was still engaged in some smalltalk when the buffet was opened, and all the good things were already gone when he finally showed up. This year, he came prepared. As soon as the last speech ended, he sprinted to the buffet and now he is among the first guests at the buffet and gets to fill his plate with whatever exquisite food he wants. But alas, the selection is huge and Herbert does not even know where to start.

As everyone knows, enjoyment of food is measured in noms. To get the maximum amount of noms, Herbert can fill his plate with many servings of any dish (even taking the same dish multiple times). Can you help him choose which dishes to fill his plate with?

Input

The input consists of:

- one line with two integers $n(1 \leq n \leq 1000)$ and $m(1 \leq m \leq 1000)$, where n is the amount of different dishes at the banquet and m is amount of servings Herberts plate can hold;
- n lines describing the different dishes. Each dish is described by:
  - one line with a string a, an integer s $(1 \leq s \leq 100)$, a real number t $(0 \leq t \leq 10)$, indicating that there are s servings available of the dish called a with a deliciousness value of t noms per serving.

All names are unique and consist of at most 30 letters from "A" to "Z" and "a" to "z".

Output

Output the maximum amount of noms that Herbert can load onto his plate. The answer should be correct up to an absolute or relative error of at most 10^{-3}.

Sample Input 1	Sample Output 1
5 6 KingPrawns 3 8.5 Bibimbap 10 5.8 FriedChicken 20 2.2 Lobster 2 9 IceCream 3 7.2	50.700000

Sample Input 2	Sample Output 2
4 10 FettuciniAlfredo 10 3.5 SpaghettiCarbonara 8 4.5 Saltimbocca 1 9 Lasagna 5 2.4	48.500000

Problem C
Packing Cases

Just recently, during Lea's visit at her uncle's house, she was reminded that while some people are quite tall, sadly she is not. She could not even reach the glasses that were stored in the topmost shelf in the kitchen. Luckily for her, there were a lot of packing cases lying around and she could use them to build a tower and then climb on it to reach the glasses.

Building such a tower is of course a very shaky endeavour, and Lea does not want to fall. So she imposed the following restriction on the tower: Given two packing cases a and b with dimensions x_a, y_a, z_a and x_b, y_b, z_b, case a may only be stacked onto case b if $x_a < x_b$ and $y_a < y_b$. Please remember that a case can be rotated to fit that restriction.

Lea now has to figure out whether it is possible to reach the desired height if she stacks the cases optimally, or not.

Input

The input consists of:

- one line with two integers h ($1 \leq h \leq 5 \cdot 10^6$) and n ($1 \leq n \leq 10^3$), where h is the height the tower should reach and n is the number of case types;
- n lines describing the case layouts. Each case layout is described by:
  - one line with three integers x, y, and z ($1 \leq x, y, z \leq 4 \cdot 10^4$), where x, y, and z are the dimensions of the case layout.

Lea has exactly 5 boxes of each type at her disposal.

Output

Output "`possible`" if Lea can build a tower of height at least h according to the constraints, otherwise output "`impossible`".

Sample Input 1

```
7 2
4 2 2
3 1 5
```

Sample Output 1

```
possible
```

Sample Input 2

```
8 2
1 6 6
2 2 2
```

Sample Output 2

```
impossible
```

This page is intentionally left blank.