Question **1**
Correct
Flag question

You are transporting some boxes through a tunnel, where each box is a parallelepiped, and is characterized by its length, width and height.

The height of the tunnel ***41*** feet and the width can be assumed to be infinite. A box can be carried through the tunnel only if its height is strictly less than the tunnel's height. Find the volume of each box that can be successfully transported to the other end of the tunnel. Note: Boxes cannot be rotated.

Input Format

The first line contains a single integer ***n***, denoting the number of boxes.

***n*** lines follow with three integers on each separated by single spaces - $length_i$, $width_i$ and $height_i$ which are length, width and height in feet of the ***i***-th box.

Constraints

$1 \leq n \leq 100$

$1 \leq length_i, width_i, height_i \leq 100$

Output Format

For every box from the input which has a height lesser than ***41*** feet, print its volume in a separate line.

Sample Input 0

```
4
5 5 5
1 2 40
10 5 41
7 2 42
```

Sample Output 0

```
125
80
```

Explanation 0

The first box is really low, only ***5*** feet tall, so it can pass through the tunnel and its volume is ***5 x 5 x 5 = 125***.

The second box is sufficiently low, its volume is ***1 x 2 x 4= = 80***.

The third box is exactly ***41*** feet tall, so it cannot pass. The same can be said about the fourth box.

**Answer:** (penalty regime: 0 %)

**Answer:** (penalty regime: 0 %)

```c
#include<stdio.h>
#define TUNNEL_HEIGHT 41
int main()
{
    int n;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        int length,width,height;
        scanf("%d %d %d",&length,&width,&height);
        if(height < TUNNEL_HEIGHT)
        {
            int volume=length*width*height;
            printf("%d\n",volume);

        }
    }
    return 0;
}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 4<br>5 5 5<br>1 2 40<br>10 5 41<br>7 2 42 | 125<br>80 | 125<br>80 | ✓ |

Passed all tests! ✓

Question **2**

Correct

Flag question

You are given ***n*** triangles, specifically, their sides $a_i$, $b_i$ and $c_i$. Print them in the same style but sorted by their areas from the smallest one to the largest one. It is guaranteed that all the areas are different.

The best way to calculate a volume of the triangle with sides ***a***, ***b*** and ***c*** is Heron's formula:

$S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$ where $p = (a+b+c)/2$.

Input Format

First line of each test file contains a single integer ***n***. ***n*** lines follow with $a_i$, $b_i$ and $c_i$ on each separated by single spaces.

Constraints

$1 \le n \le 100$

$1 \le a_i, b_i, c_i \le 70$

$a_i + b_i > c_i$, $a_i + c_i > b_i$ and $b_i + c_i > a_i$

Output Format

Print exactly ***n*** lines. On each line print ***3*** integers separated by single spaces, which are $a_i$, $b_i$ and $c_i$ of the corresponding triangle.

Sample Input 0

```
3
7 24 25
5 12 13
3 4 5
```

Sample Output 0

```
3 4 5
5 12 13
7 24 25
```

Explanation 0

The square of the first triangle is ***84***. The square of the second triangle is ***30***. The square of the third triangle is ***6***. So the sorted order is the reverse one.

**Answer:** (penalty regime: 0 %)

```c
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
double calculateArea(int a,int b,int c)
{
    double p=(a+b+c)/2.0;
    return sqrt(p*(p-a)*(p-b)*(p-c));
}
int compare(const void *t1,const void *t2)
{
    int *triangle1=(int *)t1;
    int *triangle2=(int *)t2;
    double area1=calculateArea(triangle1[0],triangle1[1],triangle1[2]);
    double area2=calculateArea(triangle2[0],triangle2[1],triangle2[2]);
    if(area1<area2)
    {
        return -1;
    }
    if(area1>area2)
    {
        return 1;
    }
    return 0;
}
int main()
{
    int n;
    scanf("%d",&n);
    int triangles[n][3];
    for (int i=0;i<n;i++)
    {
    scanf("%d %d %d",&triangles[i][0],&triangles[i][1],&triangles[i][2]);}
    qsort(triangles,n,sizeof(triangles[0]),compare);
    for (int i=0;i<n;i++)
    {
        printf("%d %d %d\n",triangles[i][0],triangles[i][1],triangles[i][2]);
    }
    return 0;

}
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 3<br>7 24 25<br>5 12 13<br>3 4 5 | 3 4 5<br>5 12 13<br>7 24 25 | 3 4 5<br>5 12 13<br>7 24 25 | ✓ |

Passed all tests! ✓