

Structures and Unions

Question 1

Boxes Through Tunnel

Problem Statement

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, width; and height; which are length, width and height in feet of the i -th box.

Constraints

$$1 \leq n \leq 100$$

$$1 \leq \text{length, width, height} \leq 100$$

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

555

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 $5 \times 5 \times 5 = 125$.

The second box is sufficiently low, its volume is $1 \times 2 \times 4 = 80$.

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 %)

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #define MAX_HEIGHT 41
4
5 struct box
6 {
7     int len;
8     int wid;
9     int hgt;
10 };
11
12 typedef struct box box;
13
14 int get_volume(box b){
15     return b.len * b.wid * b.hgt;
16 }
17 int is_lower_than_max_height(box b){
18     return b.hgt < MAX_HEIGHT;
19 }
20 int main()
21 {
22     int n;
23     scanf("%d",&n);
24     box*boxes=(box *)malloc(n*sizeof(box));
25     for(int i=0;i<n;i++){
26         scanf("%d %d %d",&boxes[i].len,&boxes[i].wid,&boxes[i].hgt);
27         if(is_lower_than_max_height(boxes[i])){
28             printf("%d\n",get_volume(boxes[i]));
29         }
30     }
31 }
32 free(boxes);
33 return 0;
34 }
```

	Input	Expected	Got	
✓	4	125	125	✓
	5 5 5	80	80	
	1 2 40			
	10 5 41			
	7 2 42			

Passed all tests! ✓

Question 2

Small Triangles , Large Triangles

Problem Statement

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:

$S = \sqrt{p * (p - a) * (p - b) * (p - c)}$ 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

$$1 \leq n \leq 100$$

$$1 \leq a, b, c \leq 70$$

$$a_i + b_i > c_i, a_i + c_i > b_i \text{ and } b_i + c_i > a_i$$

Output Format

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

Sample Input 0

3

7 24 25

5 12 13

345

Sample Output 0

345

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 %)

```
1 #include<stdio.h>
2 #include<math.h>
3 struct triangle
4 {
5     int a;
6     int b;
7     int c;
8 };
9 typedef struct triangle triangle;
10
11 double area(triangle t){
12     double s=(t.a+t.b+t.c)/2.0;
13     return sqrt(s*(s-t.a)*(s-t.b)*(s-t.c));
14 }
15 void sort_triangles_by_area(triangle*tr,int n){
16     for(int i=0;i<n-1;i++){
17         for(int j=0;j<n-1-i;j++){
18             if (area(tr[j])>area(tr[j+1])){
19                 triangle temp=tr[j];
20                 tr[j]=tr[j+1];
21                 tr[j+1]=temp;
22             }
23         }
24     }
25 }
26 int main(){
27     int n;
28     scanf("%d",&n);
29     triangle tr[n];
30     for(int i=0;i<n;i++){
31         scanf("%d %d %d",&tr[i].a,&tr[i].b,&tr[i].c);
32     }
33     sort_triangles_by_area(tr,n);
34     for(int i=0;i<n;i++){
35         printf("%d %d %d \n",tr[i].a,tr[i].b,tr[i].c);
36     }
37     return 0;
38 }
39
```

	Input	Expected	Got	
✓	3 7 24 25 5 12 13 3 4 5	3 4 5 5 12 13 7 24 25	3 4 5 5 12 13 7 24 25	✓

Passed all tests! ✓