```c
#include<stdio.h>
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<41){
int volume=length*width*height;
printf("%d\n",volume);
}}}
```

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

Passed all tests! ✓

```c
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
typedef struct{
    double area;
    int a,b,c;

}Triangle;
double calculate_area(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*x,const void*y){
    Triangle*t1=(Triangle*)x;
    Triangle*t2=(Triangle*)y;
    if(t1->area<t2->area)return -1;
    if(t1->area>t2->area)return 1;
    return 0;
}
int main(){
    int n;
    scanf("%d",&n);
    Triangle triangles[n];
    for(int i=0;i<n;i++){
        int a,b,c;
        scanf("%d %d %d",&a,&b,&c);
        triangles[i].a=a;
        triangles[i].b=b;
        triangles[i].c=c;
        triangles[i].area=calculate_area(a,b,c);

    }
    qsort(triangles,n,sizeof(Triangle),compare);
    for(int i=0;i<n;i++){
        printf("%d %d %d\n",triangles[i].a,triangles[i].b,triangles[i].c);
    }
    return 0;
}
```

	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! ✓

```c
/*
 * Complete the 'reverseArray' function below.
 *
 * The function is expected to return an INTEGER_ARRAY.
 * The function accepts INTEGER_ARRAY arr as parameter.
 */

/*
 * To return the integer array from the function, you should:
 *     - Store the size of the array to be returned in the result_count variable
 *     - Allocate the array statically or dynamically
 *
 * For example,
 * int* return_integer_array_using_static_allocation(int* result_count) {
 *     *result_count = 5;
 *
 *     static int a[5] = {1, 2, 3, 4, 5};
 *
 *     return a;
 * }
 *
 * int* return_integer_array_using_dynamic_allocation(int* result_count) {
 *     *result_count = 5;
 *
 *     int *a = malloc(5 * sizeof(int));
 *
 *     for (int i = 0; i < 5; i++) {
 *         *(a + i) = i + 1;
 *     }
 *
 *     return a;
 * }
 *
 */
#include<stdio.h>
#include<stdlib.h>
int* reverseArray(int arr_count, int *arr, int *result_count) {
int*result=(int*)malloc(arr_count*sizeof(int));
```

```
38 int*result=(int*)malloc(arr_count*sizeof(int));
39 if(result==NULL){
40     return NULL;
41 }
42 for(int i=0;i<arr_count;i++)
43 {
44     result[i]=arr[arr_count-i-1];
45 }
46 *result_count=arr_count;
47 return result;
48 }
49
```

	Test	Expected	Got	

✓	int arr[] = {1, 3, 2, 4, 5};
	int result_count;
	int* result = reverseArray(5, arr, &result_count);
	for (int i = 0; i < result_count; i++)
	printf("%d\n", *(result + i));
		5
		4
		2
		3
		1
			5
			4
			2
			3
			1
				✓

Passed all tests! ✓

```c
/*
 * Complete the 'cutThemAll' function below.
 *
 * The function is expected to return a STRING.
 * The function accepts following parameters:
 *  1. LONG_INTEGER_ARRAY lengths
 *  2. LONG_INTEGER minLength
 */

/*
 * To return the string from the function, you should either do static allocation or dynamic allocation
 *
 * For example,
 * char* return_string_using_static_allocation() {
 *     static char s[] = "static allocation of string";
 *
 *     return s;
 * }
 *
 * char* return_string_using_dynamic_allocation() {
 *     char* s = malloc(100 * sizeof(char));
 *
 *     s = "dynamic allocation of string";
 *
 *     return s;
 * }
 *
 */
 #include<stdio.h>
char* cutThemAll(int lengths_count, long *lengths, long minlength) {
long t=0,i=1;
for(int i=0;i<lengths_count;i++){
    t+=lengths[i];
}
do{
    if(t-lengths[lengths_count-1]<minlength){
        return "Impossible";
    }
```

	Test	Expected	Got	
✓	`long lengths[] = {3, 5, 4, 3};` `printf("%s", cutThemAll(4, lengths, 9))`	`Possible`	`Possible`	✓
✓	`long lengths[] = {5, 6, 2};` `printf("%s", cutThemAll(3, lengths, 12))`	`Impossible`	`Impossible`	✓

Passed all tests! ✓