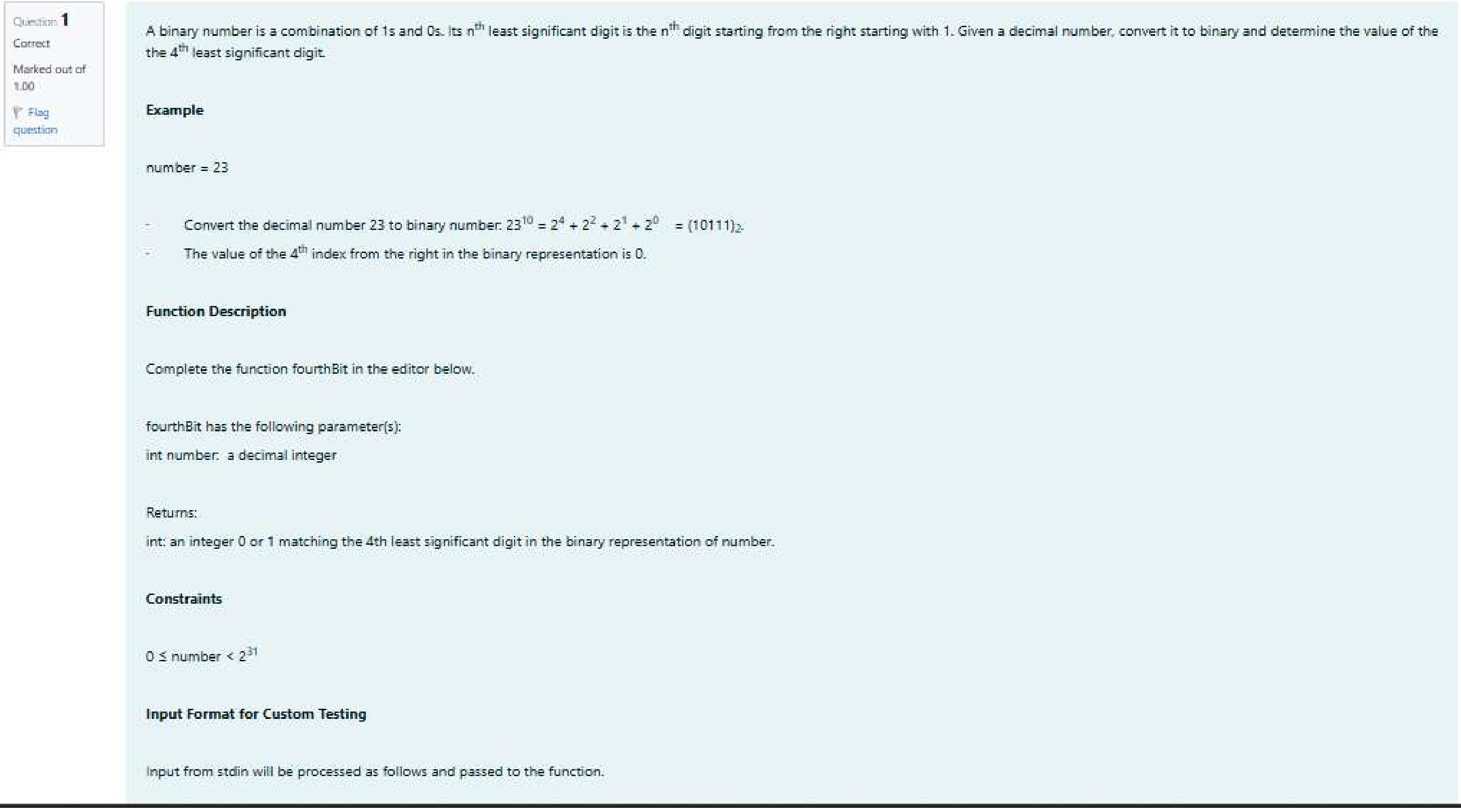


Question **1**

Correct

Marked out of 1.00

Flag question

A binary number is a combination of 1s and 0s. Its nth least significant digit is the nth digit starting from the right starting with 1. Given a decimal number, convert it to binary and determine the value of the the 4th least significant digit.

Example

number = 23

- Convert the decimal number 23 to binary number: $23^{10} = 2^4 + 2^2 + 2^1 + 2^0 \quad = (10111)_2$.
- The value of the 4th index from the right in the binary representation is 0.

Function Description

Complete the function fourthBit in the editor below.

fourthBit has the following parameter(s):
int number: a decimal integer

Returns:
int: an integer 0 or 1 matching the 4th least significant digit in the binary representation of number.

Constraints

$0 \leq$ number $< 2^{31}$

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

Sample Input 0

```
STDIN  Function
-----  --------
32   → number = 32
```

Sample Output 0

```
0
```

Explanation 0

- Convert the decimal number 32 to binary number: 32_{10} = $(100000)_2$.
- The value of the 4th index from the right in the binary representation is 0.

Sample Case 1

Sample Input 1

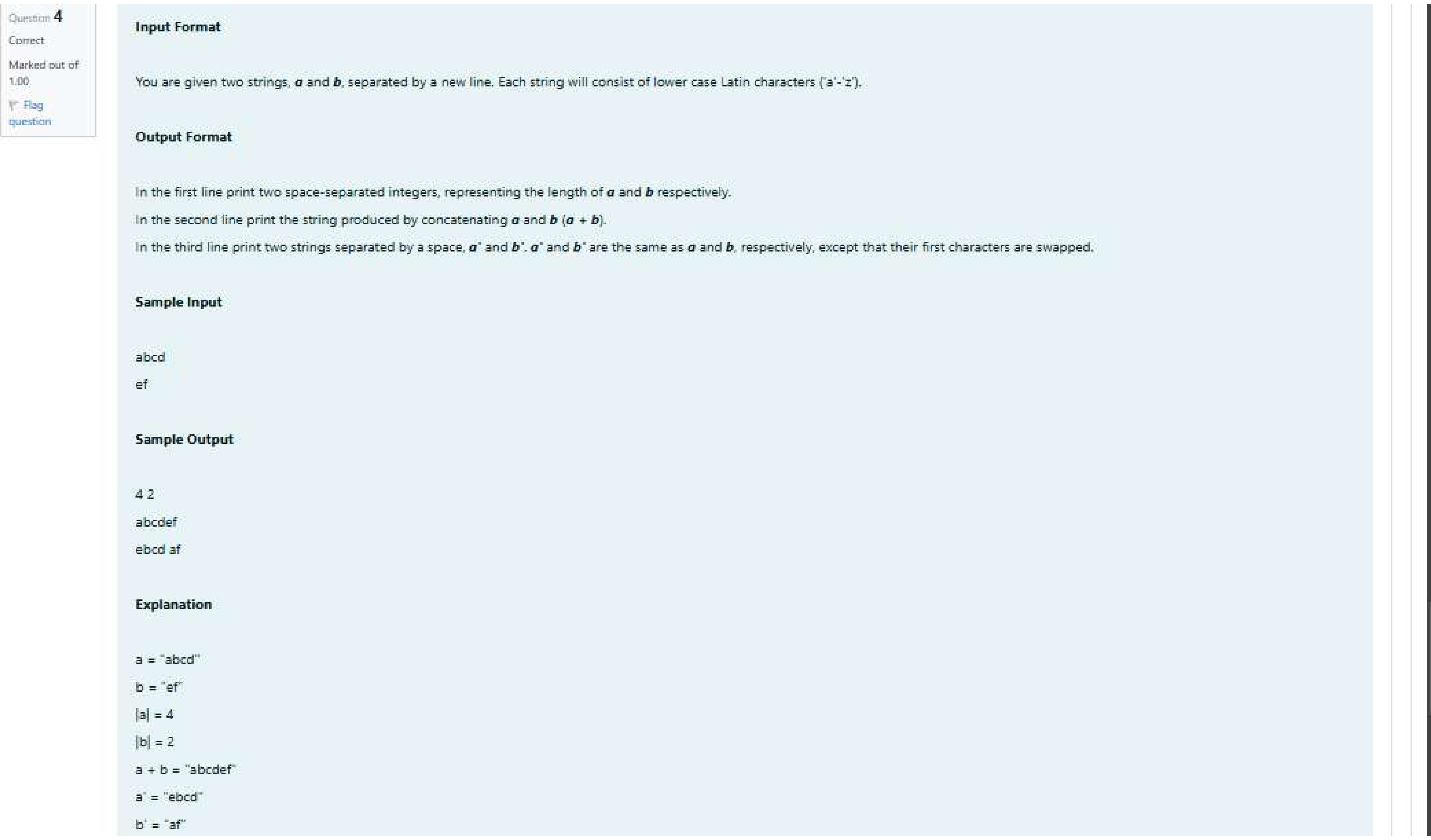
```
STDIN   Function
-----   --------
77   →  number = 77
```

Sample Output 1

```
1
```

Explanation 1

- Convert the decimal number 77 to binary number: 77_{10} = $(1001101)_2$.



Question 4

Correct

Marked out of 1.00

Flag question

Input Format

You are given two strings, ***a*** and ***b***, separated by a new line. Each string will consist of lower case Latin characters ('a'-'z').

Output Format

In the first line print two space-separated integers, representing the length of ***a*** and ***b*** respectively.

In the second line print the string produced by concatenating ***a*** and ***b*** (***a*** + ***b***).

In the third line print two strings separated by a space, ***a'*** and ***b'***. ***a'*** and ***b'*** are the same as ***a*** and ***b***, respectively, except that their first characters are swapped.

Sample Input

```
abcd
ef
```

Sample Output

```
4 2
abcdef
ebcd af
```

Explanation

a = "abcd"

b = "ef"

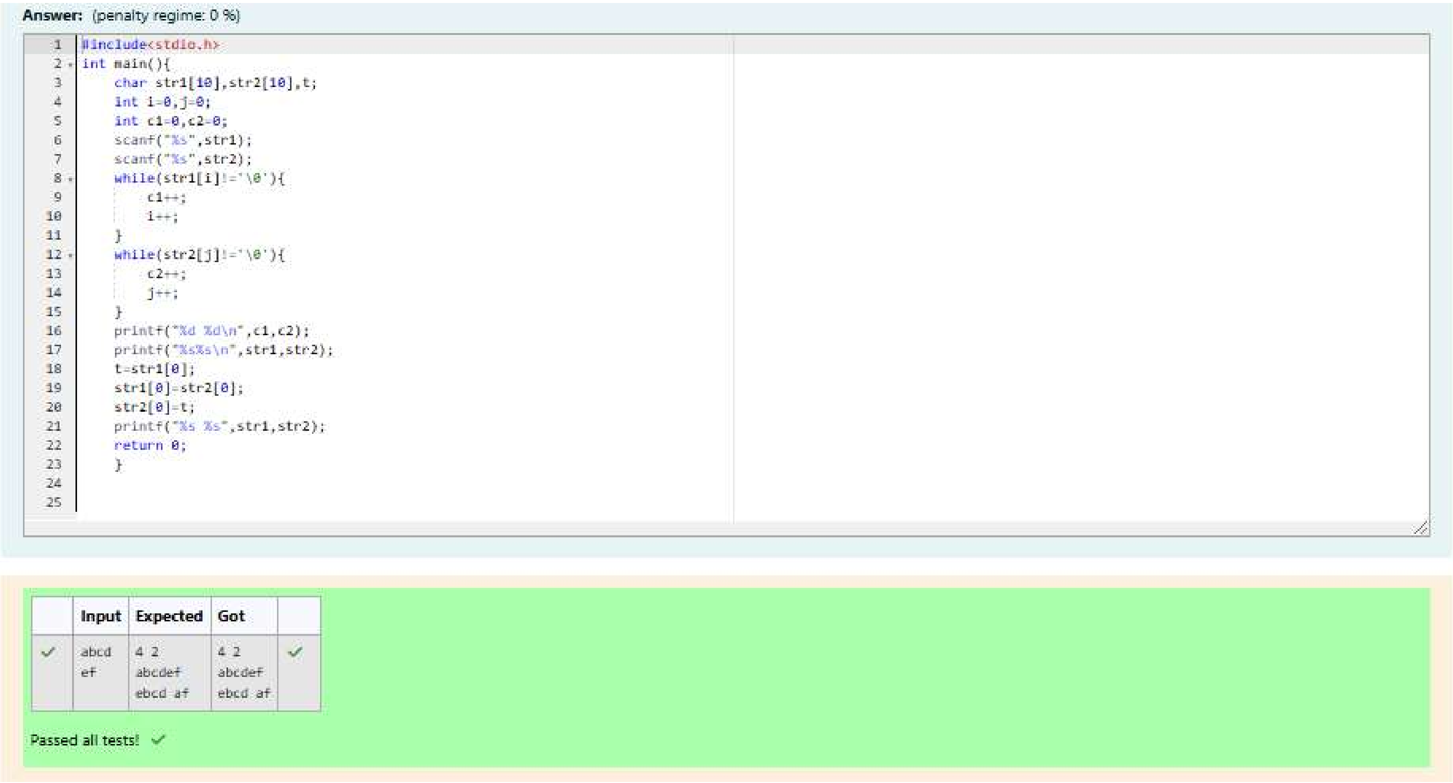
|a| = 4

|b| = 2

a + b = "abcdef"

a' = "ebcd"

b' = "af"

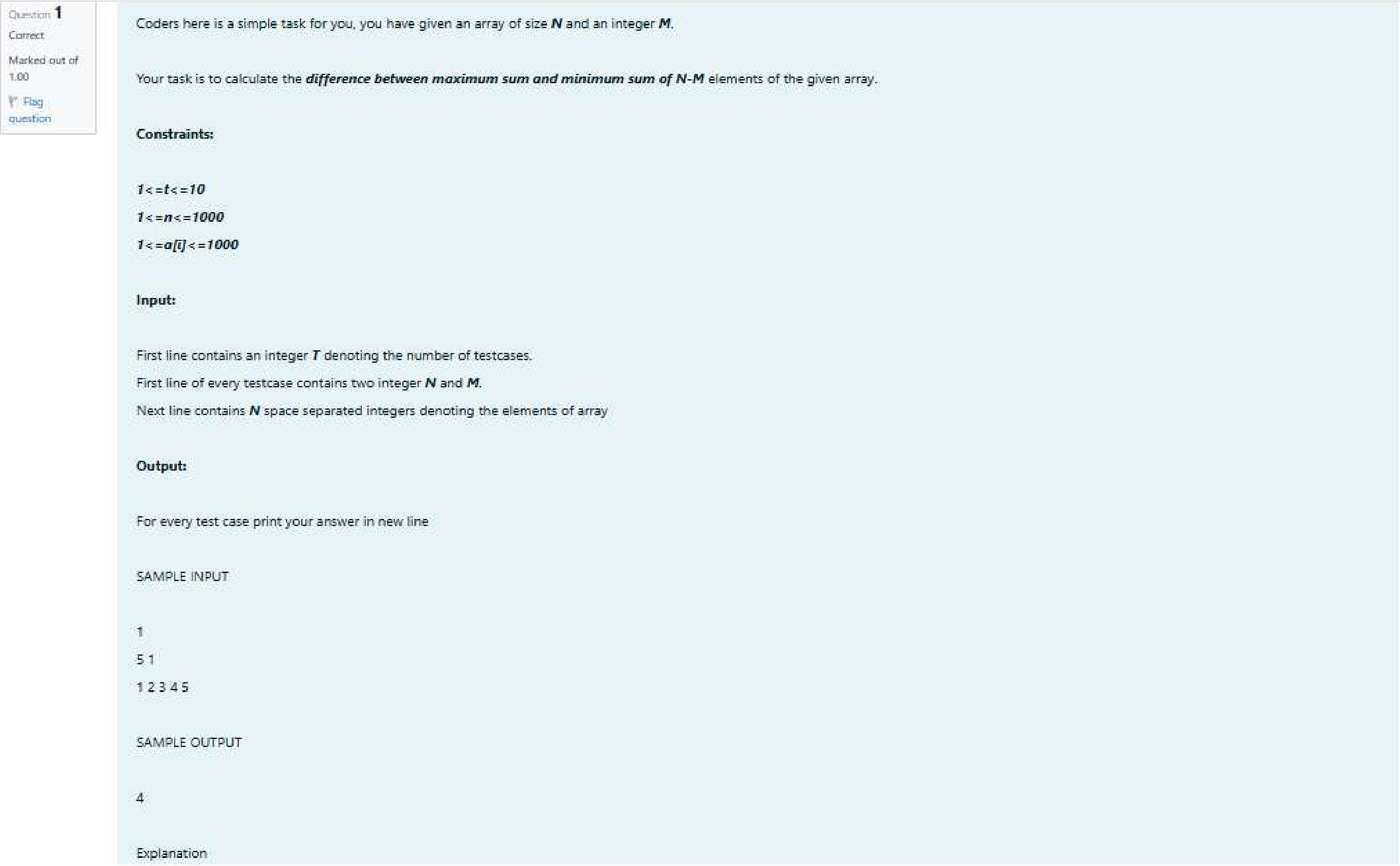


Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    char str1[10],str2[10],t;
    int i=0,j=0;
    int c1=0,c2=0;
    scanf("%s",str1);
    scanf("%s",str2);
    while(str1[i]!='\0'){
        c1++;
        i++;
    }
    while(str2[j]!='\0'){
        c2++;
        j++;
    }
    printf("%d %d\n",c1,c2);
    printf("%s%s\n",str1,str2);
    t=str1[0];
    str1[0]=str2[0];
    str2[0]=t;
    printf("%s %s",str1,str2);
    return 0;
    }


```

	Input	Expected	Got	
✓	abcd ef	4 2 abcdef ebcd af	4 2 abcdef ebcd af	✓

Passed all tests! ✓



Coders here is a simple task for you, you have given an array of size ***N*** and an integer ***M***.

Your task is to calculate the ***difference between maximum sum and minimum sum of N-M*** elements of the given array.

Constraints:

1<=t<=10

1<=n<=1000

1<=a[i]<=1000

Input:

First line contains an integer ***T*** denoting the number of testcases.

First line of every testcase contains two integer ***N*** and ***M***.

Next line contains ***N*** space separated integers denoting the elements of array

Output:

For every test case print your answer in new line

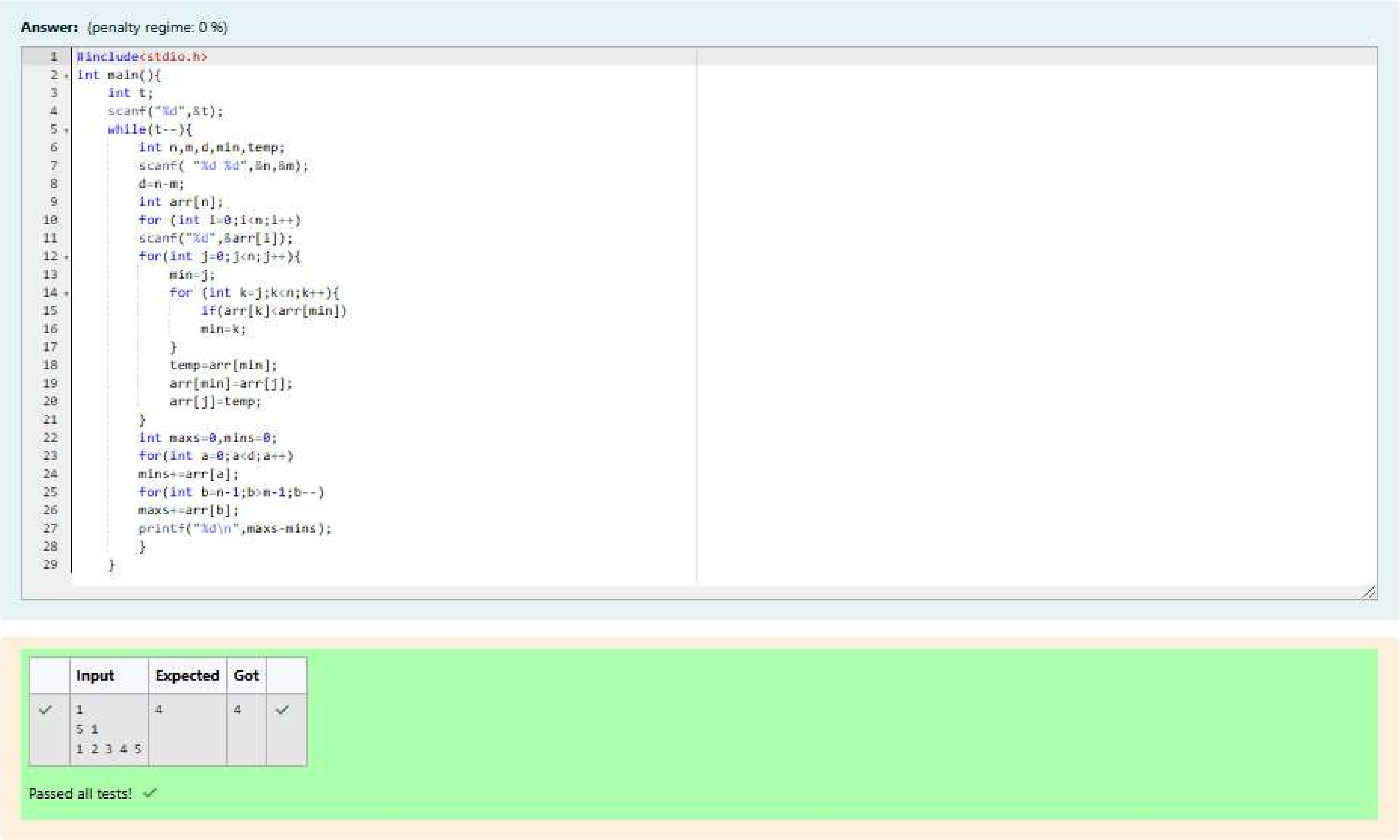
SAMPLE INPUT

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

SAMPLE OUTPUT

```
4
```

Explanation

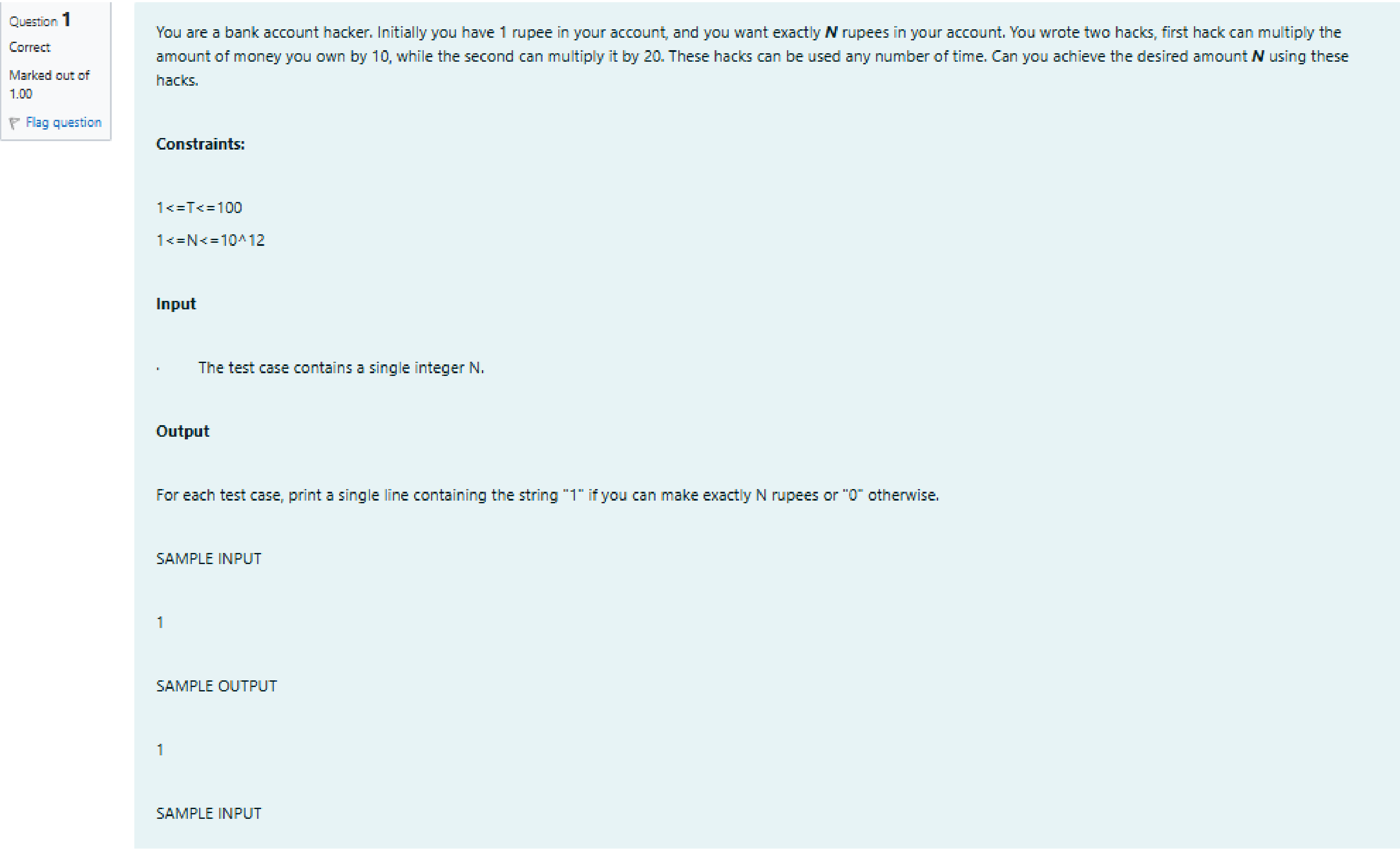


Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int t;
    scanf("%d",&t);
    while(t--){
        int n,m,d,min,temp;
        scanf( "%d %d",&n,&m);
        d=n-m;
        int arr[n];
        for (int i=0;i<n;i++)
        scanf("%d",&arr[i]);
        for(int j=0;j<n;j++){
            min=j;
            for (int k=j;k<n;k++){
                if(arr[k]<arr[min])
                min=k;
            }
            temp=arr[min];
            arr[min]=arr[j];
            arr[j]=temp;
        }
        int maxs=0,mins=0;
        for(int a=0;a<d;a++)
        mins+=arr[a];
        for(int b=n-1;b>m-1;b--)
        maxs+=arr[b];
        printf("%d\n",maxs-mins);
        }
    }
```

	Input	Expected	Got	
✓	1 5 1 1 2 3 4 5	4	4	✓

Passed all tests! ✓



Question **1**

Correct

Marked out of 1.00

You are a bank account hacker. Initially you have 1 rupee in your account, and you want exactly ***N*** rupees in your account. You wrote two hacks, first hack can multiply the amount of money you own by 10, while the second can multiply it by 20. These hacks can be used any number of time. Can you achieve the desired amount ***N*** using these hacks.

Constraints:

1<=T<=100

1<=N<=10^12

Input

- The test case contains a single integer N.

Output

For each test case, print a single line containing the string "1" if you can make exactly N rupees or "0" otherwise.

SAMPLE INPUT

1

SAMPLE OUTPUT

1

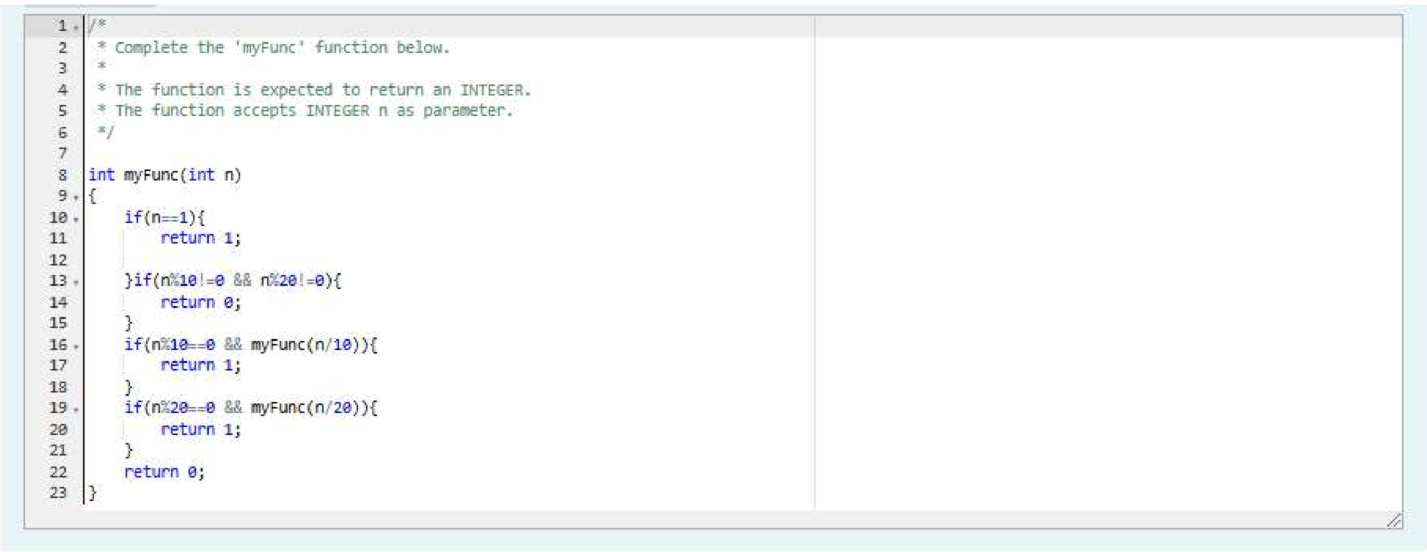
SAMPLE INPUT

```
2
```

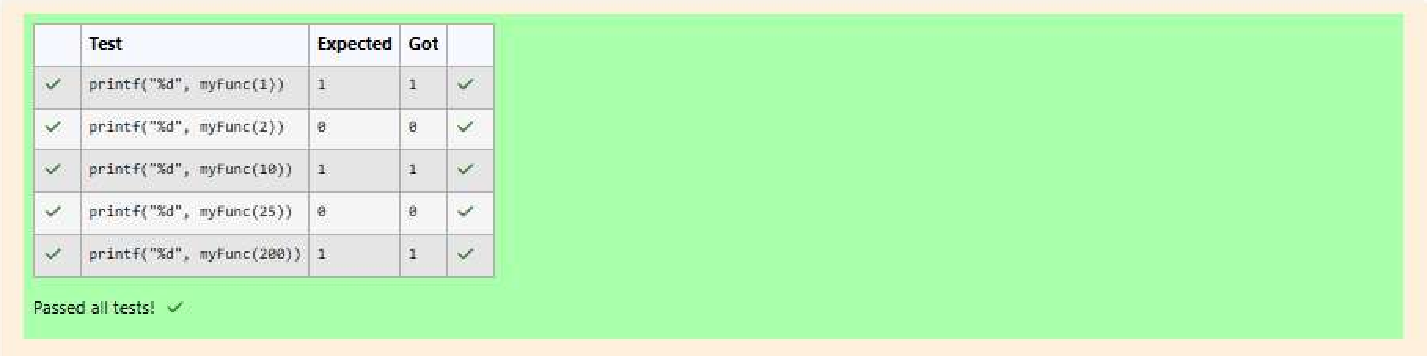
SAMPLE OUTPUT

```
0
```

Answer: (penalty regime: 0 %)

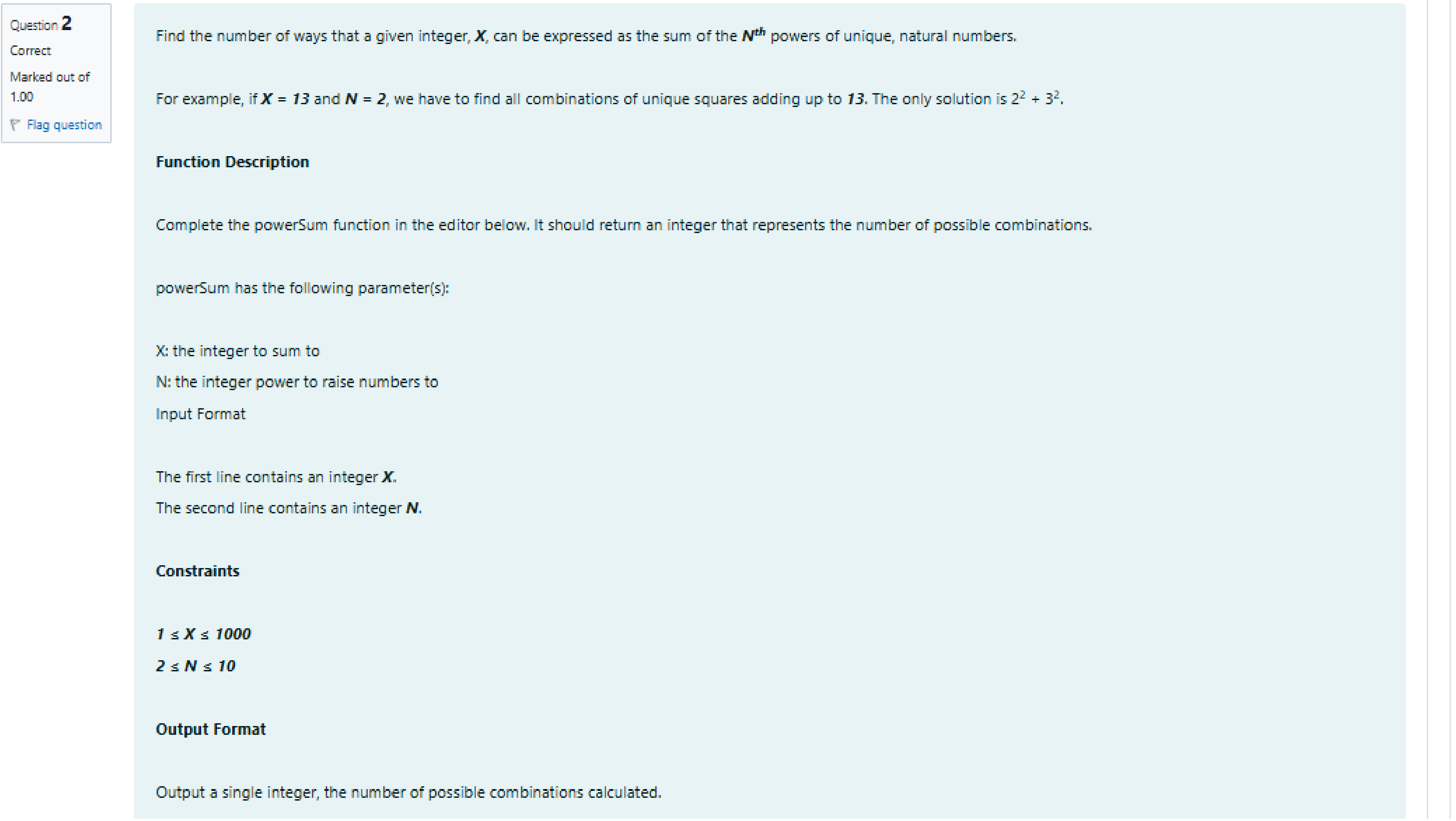


```
/*
 * Complete the 'myFunc' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER n as parameter.
 */

int myFunc(int n)
{
    if(n==1){
        return 1;

    }if(n%10!=0 && n%20!=0){
        return 0;
    }
    if(n%10==0 && myFunc(n/10)){
        return 1;
    }
    if(n%20==0 && myFunc(n/20)){
        return 1;
    }
    return 0;
}
```



	Test	Expected	Got	
✓	printf("%d", myFunc(1))	1	1	✓
✓	printf("%d", myFunc(2))	0	0	✓
✓	printf("%d", myFunc(10))	1	1	✓
✓	printf("%d", myFunc(25))	0	0	✓
✓	printf("%d", myFunc(200))	1	1	✓

Passed all tests! ✓



Question **2**

Correct

Marked out of 1.00

Flag question

Find the number of ways that a given integer, X, can be expressed as the sum of the N^{th} powers of unique, natural numbers.

For example, if $X = 13$ and $N = 2$, we have to find all combinations of unique squares adding up to 13. The only solution is $2^2 + 3^2$.

Function Description

Complete the powerSum function in the editor below. It should return an integer that represents the number of possible combinations.

powerSum has the following parameter(s):

X: the integer to sum to

N: the integer power to raise numbers to

Input Format

The first line contains an integer X.

The second line contains an integer N.

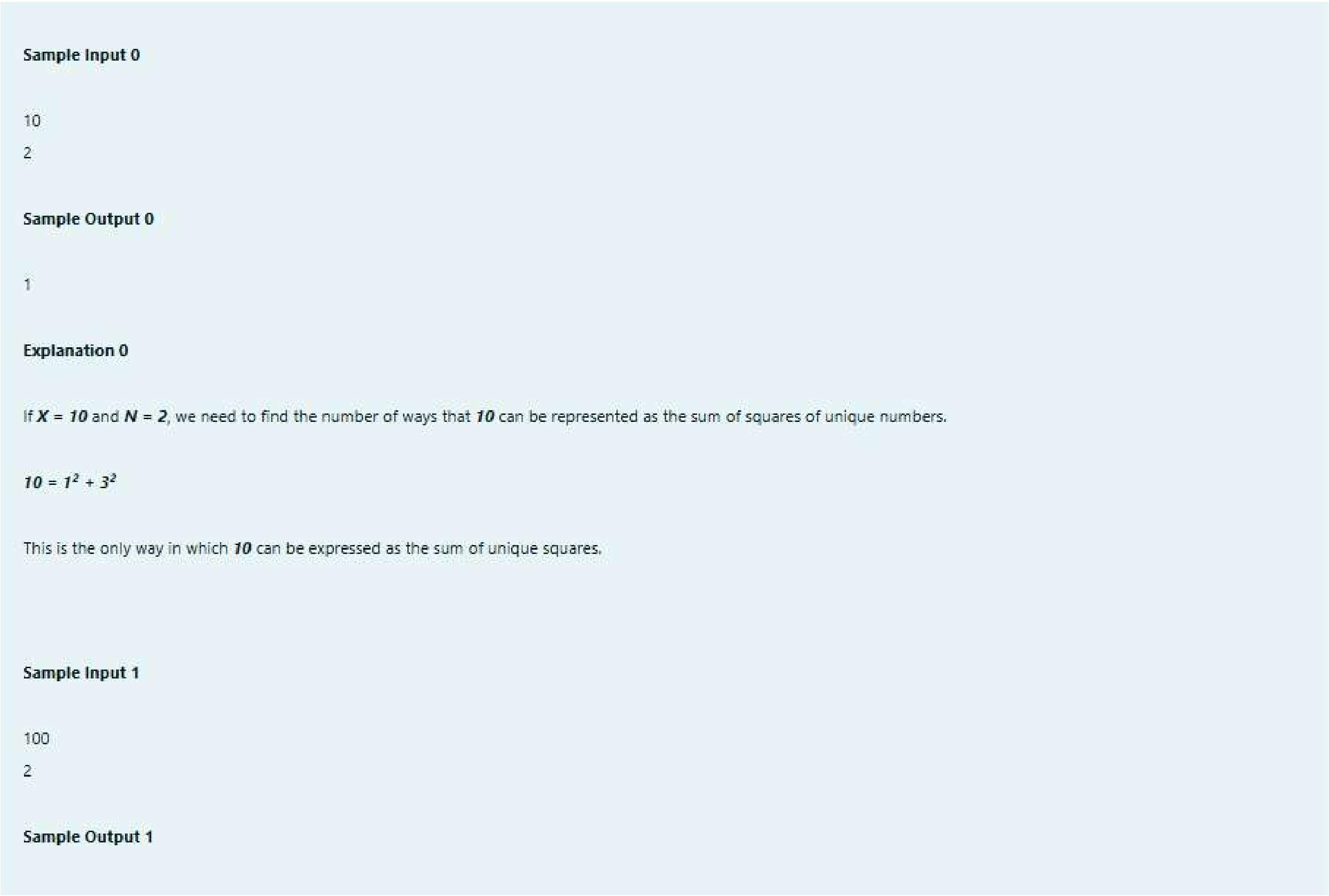
Constraints

$1 \le X \le 1000$

$2 \le N \le 10$

Output Format

Output a single integer, the number of possible combinations calculated.



Sample Input 0

```
10
2
```

Sample Output 0

```
1
```

Explanation 0

If ***X*** **=** ***10*** and ***N*** **=** ***2***, we need to find the number of ways that ***10*** can be represented as the sum of squares of unique numbers.

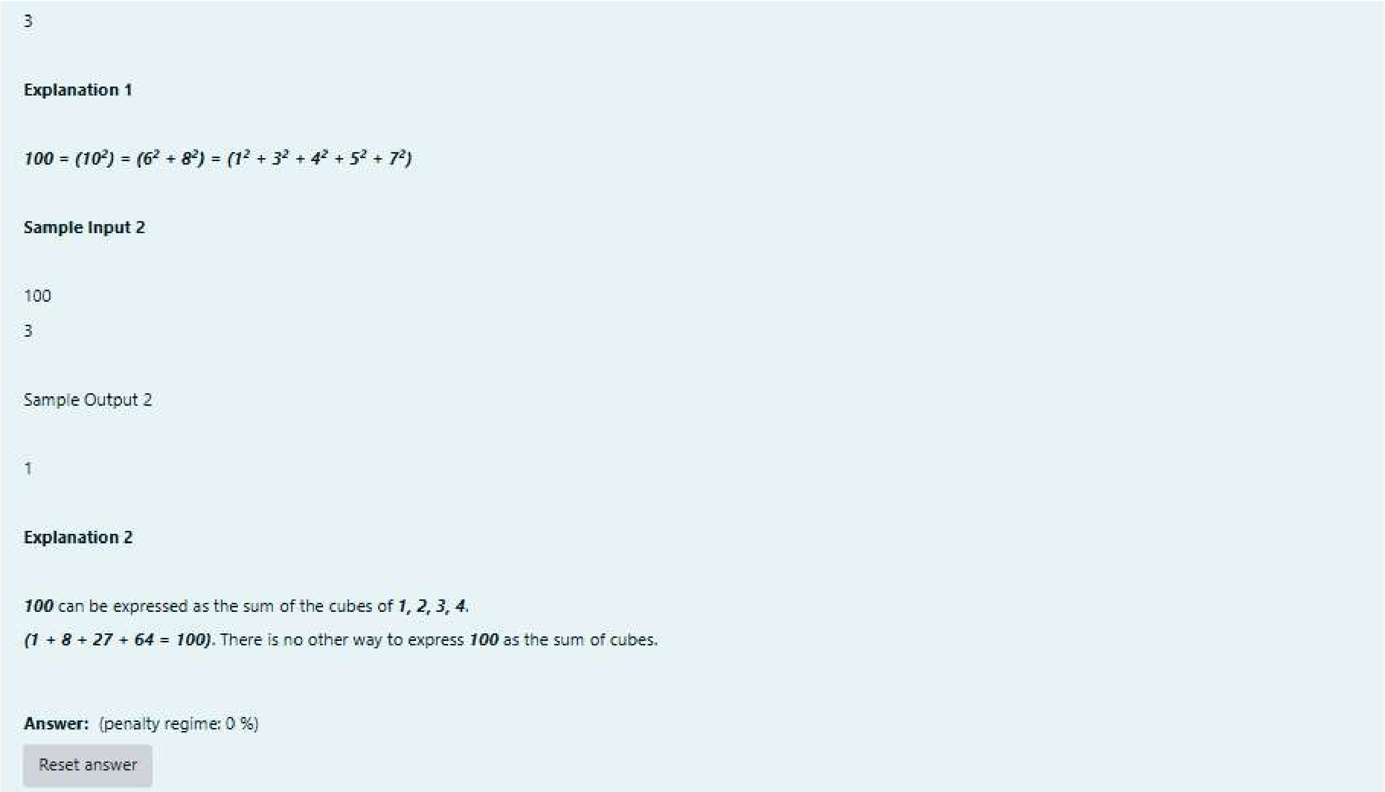
$10 = 1^2 + 3^2$

This is the only way in which ***10*** can be expressed as the sum of unique squares.

Sample Input 1

```
100
2
```

Sample Output 1



3

Explanation 1

$100 = (10^2) = (6^2 + 8^2) = (1^2 + 3^2 + 4^2 + 5^2 + 7^2)$

Sample Input 2

100
3

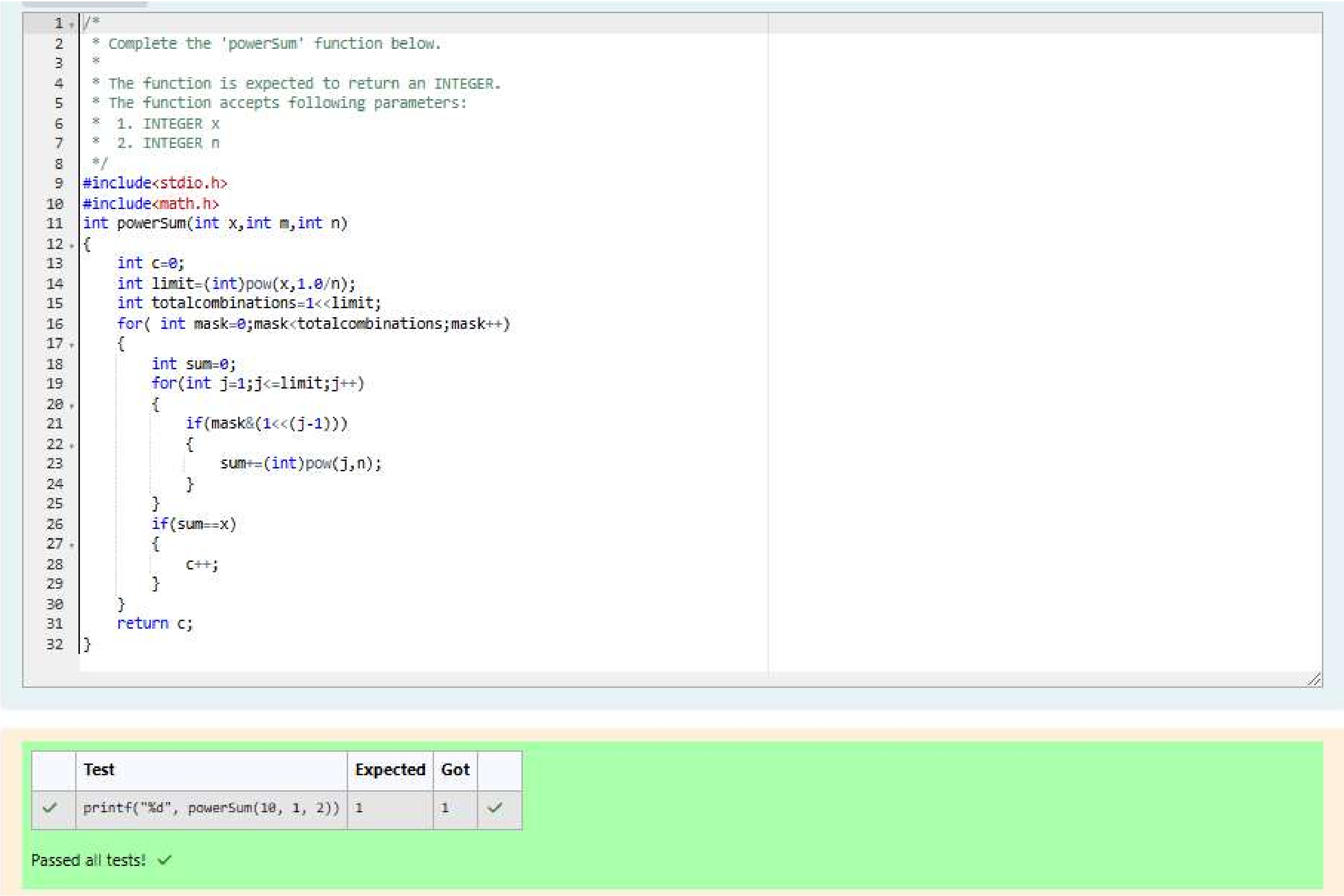
Sample Output 2

1

Explanation 2

100 can be expressed as the sum of the cubes of ***1, 2, 3, 4***.
(1 + 8 + 27 + 64 = 100). There is no other way to express ***100*** as the sum of cubes.

Answer: (penalty regime: 0 %)



```
/*
 * Complete the 'powerSum' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER x
 *  2. INTEGER n
 */
#include<stdio.h>
#include<math.h>
int powerSum(int x,int m,int n)
{
    int c=0;
    int limit=(int)pow(x,1.0/n);
    int totalcombinations=1<<limit;
    for( int mask=0;mask<totalcombinations;mask++)
    {
        int sum=0;
        for(int j=1;j<=limit;j++)
        {
            if(mask&(1<<(j-1)))
            {
                sum+=(int)pow(j,n);
            }
        }
        if(sum==x)
        {
            c++;
        }
    }
    return c;
}
```

	Test	Expected	Got	
✓	printf("%d", powerSum(10, 1, 2))	1	1	✓

Passed all tests! ✓