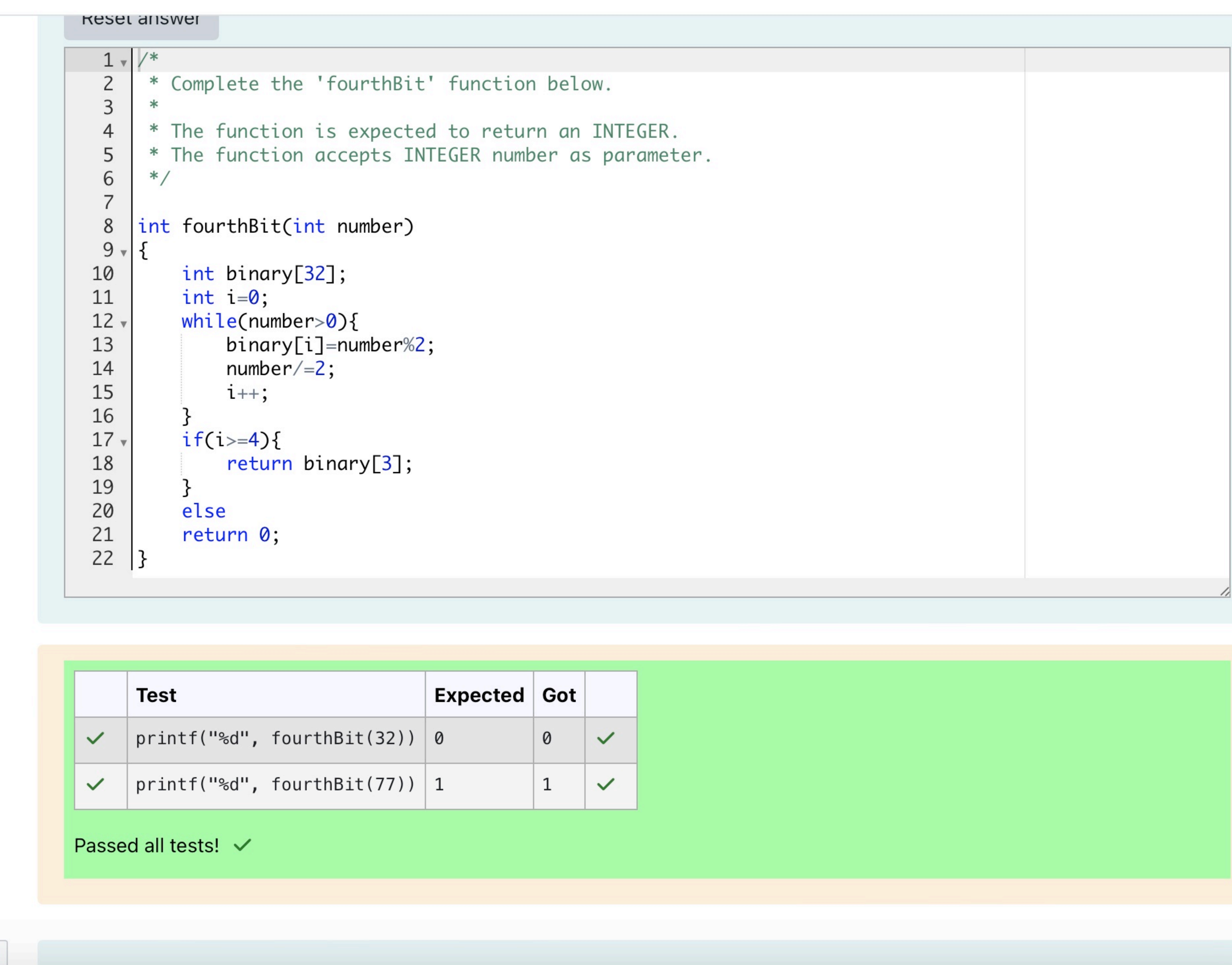
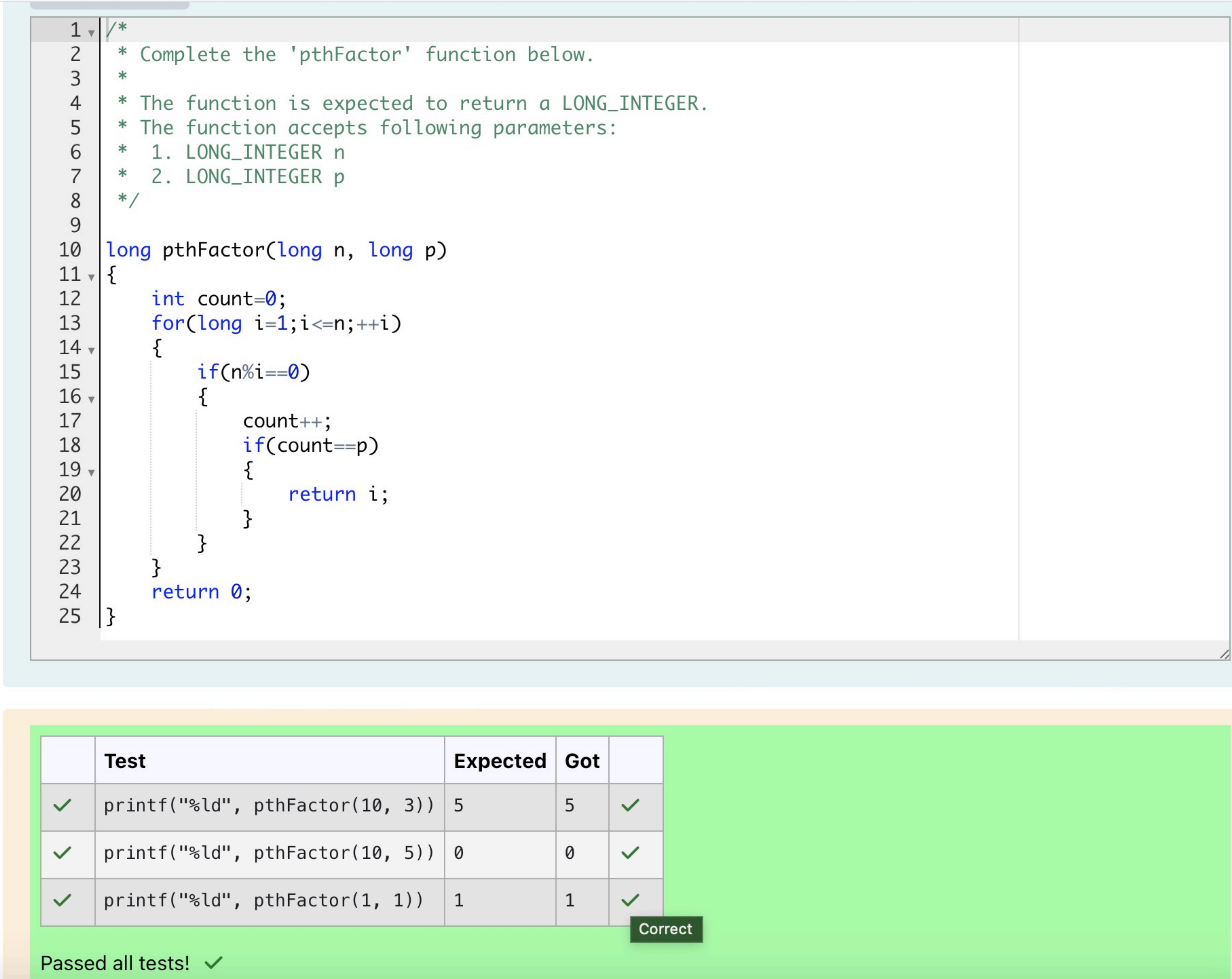


Reset answer

```c
/*
 * Complete the 'fourthBit' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER number as parameter.
 */

int fourthBit(int number)
{
    int binary[32];
    int i=0;
    while(number>0){
        binary[i]=number%2;
        number/=2;
        i++;
    }
    if(i>=4){
        return binary[3];
    }
    else
    return 0;
}
```

	Test	Expected	Got	
✓	`printf("%d", fourthBit(32))`	0	0	✓
✓	`printf("%d", fourthBit(77))`	1	1	✓

Passed all tests! ✓

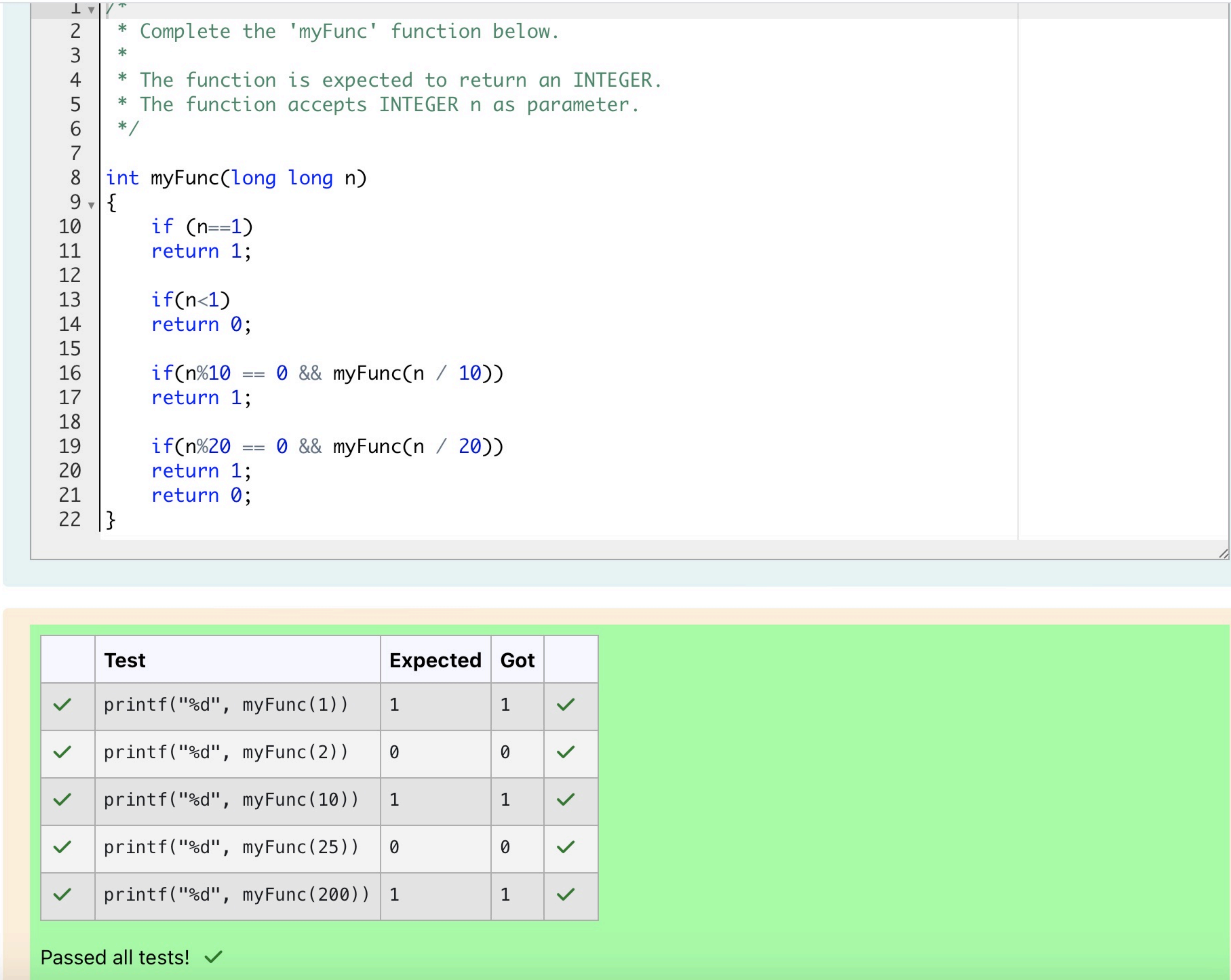


```c
/*
 * Complete the 'pthFactor' function below.
 *
 * The function is expected to return a LONG_INTEGER.
 * The function accepts following parameters:
 *  1. LONG_INTEGER n
 *  2. LONG_INTEGER p
 */

long pthFactor(long n, long p)
{
    int count=0;
    for(long i=1;i<=n;++i)
    {
        if(n%i==0)
        {
            count++;
            if(count==p)
            {
                return i;
            }
        }
    }
    return 0;
}
```

	Test	Expected	Got	
✓	`printf("%ld", pthFactor(10, 3))`	5	5	✓
✓	`printf("%ld", pthFactor(10, 5))`	0	0	✓
✓	`printf("%ld", pthFactor(1, 1))`	1	1	✓

Correct

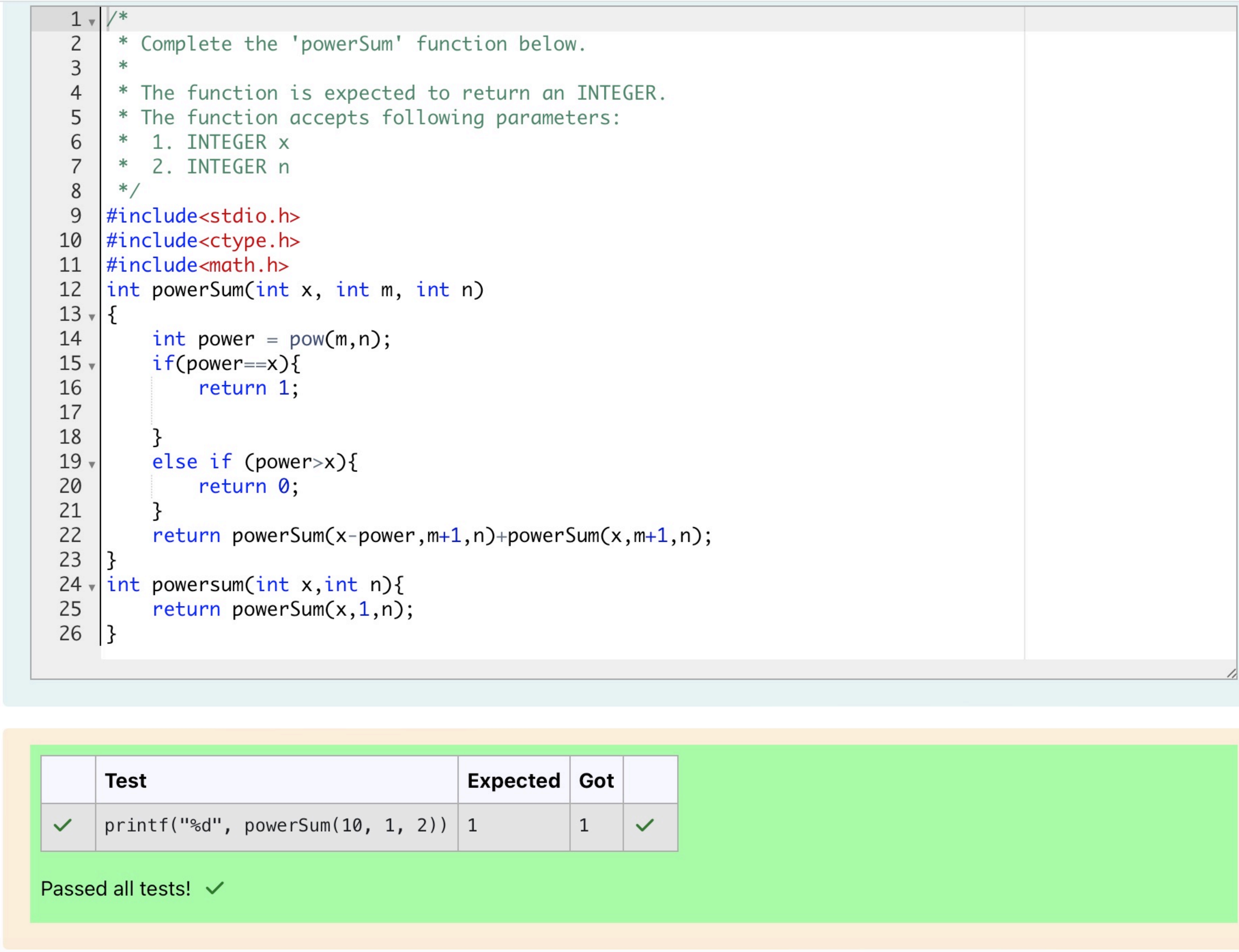
Passed all tests! ✓



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

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

    if(n<1)
    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! ✓



```c
/*
 * 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<ctype.h>
#include<math.h>
int powerSum(int x, int m, int n)
{
    int power = pow(m,n);
    if(power==x){
        return 1;

    }
    else if (power>x){
        return 0;
    }
    return powerSum(x-power,m+1,n)+powerSum(x,m+1,n);
}
int powersum(int x,int n){
    return powerSum(x,1,n);
}
```

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

Passed all tests! ✓