```
 3  *
 4  * The function is expected
 5  * The function accepts INTE
 6  */
 7
 8  int fourthBit(int number)
 9  {
10      int binary[32];
11      int i=0;
12      while(number>0)
13      {
14          binary[i]=number%2;
15          number/=2;
16          i++;
17      }
18      if(i>=4)
19      {
20          return binary[3];
21      }
22      else
23      return 0;
24
25  }
```

	Test	Ex
✓	printf("%d", fourthBit(32))	0
✓	printf("%d", fourthBit(77))	1

Passed all tests! ✓

```
4  * The function is expected
5  * The function accepts foll
6  *  1. LONG_INTEGER n
7  *  2. LONG_INTEGER p
8  */
9
10 long pthFactor(long n, long
11 {
12   int count=0;
13   for(long i=1;i<=n;i++)
14   {
15       if(n%i==0)
16       {
17           count++;
18           if(count==p)
19           {
20               return i;
21           }
22       }
23   }
24   return 0;
25 }
```

	Test
✓	printf("%ld", pthFactor(10, 3)
✓	printf("%ld", pthFactor(10, 5)
✓	printf("%ld", pthFactor(1, 1))

Passed all tests! ✓

```c
/*
 * Complete the 'myFunc' fun
 *
 * The function is expected
 * The function accepts INTE
 */

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

```

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

Passed all tests! ✓

```c
/*
 * Complete the 'powerSum' f
 *
 * The function is expected
 * The function accepts foll
 *  1. INTEGER x
 *  2. INTEGER n
 */

int powerSum(int x, int m, i
{
    int power = 1;
    for(int i = 0; i < n; i+
    power *= m;
    if(power > x) return 0;
    if(power == x) return 1;
    return powerSum(x - powe
}
```

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

Passed all tests! ✓