

```
1  /*
2   * Complete the 'fourthBit' function below.
3   *
4   * The function is expected to return an INTEGER.
5   * The function accepts INTEGER number as parameter.
6   */
7
8  int fourthBit(int number)
9  {
10     int binary[32];
11     int i=0;
12     while(number>0){
13         binary[i]=number%2;
14         number/=2;
15         i++;
16     }
17     if(i>=4){
18         return binary[3];
19     }
20     else
21     return 0;
22 }
```

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

Passed all tests! ✓

```

1  /*
2   * Complete the 'pthFactor' function below.
3   *
4   * The function is expected to return a LONG_INTEGER.
5   * The function accepts following parameters:
6   * 1. LONG_INTEGER n
7   * 2. LONG_INTEGER p
8   */
9
10 long pthFactor(long n, long p)
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	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! ✓

```

1  /*
2  * Complete the 'myFunc' function below.
3  *
4  * The function is expected to return an INTEGER.
5  * The function accepts INTEGER n as parameter.
6  */
7
8  int myFunc(long long n)
9  {
10     if (n==1)
11         return 1;
12
13     if(n<1)
14         return 0;
15
16     if(n%10 == 0 && myFunc(n / 10))
17         return 1;
18
19     if(n%20 == 0 && myFunc(n / 20))
20         return 1;
21     return 0;
22 }

```

	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! ✓

```

1  /*
2   * Complete the 'powerSum' function below.
3   *
4   * The function is expected to return an INTEGER.
5   * The function accepts following parameters:
6   * 1. INTEGER x
7   * 2. INTEGER n
8   */
9  #include<stdio.h>
10 #include<ctype.h>
11 #include<math.h>
12 int powerSum(int x, int m, int n)
13 {
14     int power = pow(m,n);
15     if(power==x){
16         return 1;
17     }
18     else if (power>x){
19         return 0;
20     }
21     return powerSum(x-power,m+1,n)+powerSum(x,m+1,n);
22 }
23
24 int powersum(int x,int n){
25     return powerSum(x,1,n);
26 }

```

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

Passed all tests! ✓