Answer: (penalty regime: 0 %)

Reset answer

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

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! ✓

Answer: (penalty regime: 0 %)

Reset answer

```c
/*
 * Complete the 'pthFactor' function belo
 *
 * The function is expected to return a L
 * The function accepts following paramet
 *  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	G
✓	`printf("%ld", pthFactor(10, 3))`	5	5
✓	`printf("%ld", pthFactor(10, 5))`	0	0
✓	`printf("%ld", pthFactor(1, 1))`	1	1

Passed all tests! ✓

Answer: (penalty regime: 0 %)

Reset answer

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

int myFunc(int n)
{
    while(n>1){
        if(n%20==0){
            n/=20;}
        else if(n%10==0){
            n/=10;
        }
        else return 0 ;

    }
    return 1;
}
```

	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! ✓

Answer: (penalty regime: 0 %)

Reset answer

```c
/*
 * Complete the 'powerSum' function below
 *
 * The function is expected to return an
 * The function accepts following paramet
 *  1. INTEGER x
 *  2. INTEGER n
 */
#include<stdio.h>
#include<math.h>
#include<ctype.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)+powerS

    }
    int powersum(int x,int n){
        return powerSum(x,1,n);}
```

st	Expected	Got	
ntf("%d", powerSum(10, 1, 2))	1	1	✓

Passed all tests! ✓