

Birla Institute of Technology & Science, Pilani, Hyderabad Campus

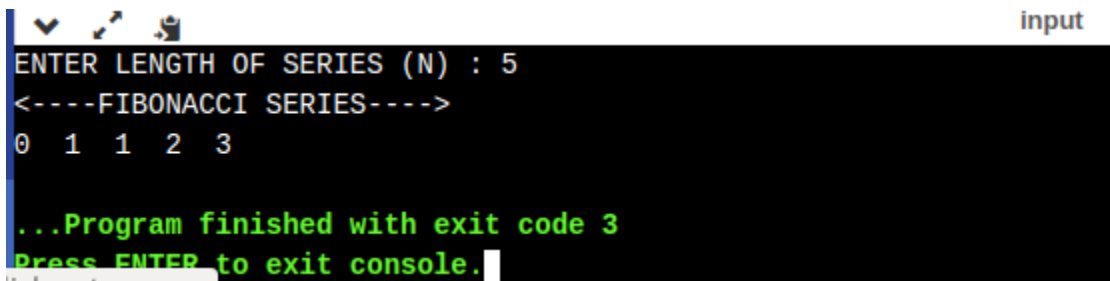
Second Semester 2020-2021

Computer Programming [CS F111]

Lab 4

Q1. Print the Fibonacci series for a given n.

```
1  #include <stdio.h>
2
3  void main()
4  {
5      int num1=0, num2=1,no,counter,fab;
6
7      printf("ENTER LENGTH OF SERIES (N) : ");
8      scanf("%d",&no);
9
10     printf("<----FIBONACCI SERIES---->");
11     printf("\n%d  %d",num1,num2);
12     for(counter = 1; counter <= no-2; counter++)
13     {
14         fab=num1 + num2;
15         printf("  %d",fab);
16         num1=num2;
17         num2=fab;
18     }
19 }
20
```



input

```
ENTER LENGTH OF SERIES (N) : 5
<----FIBONACCI SERIES---->
0  1  1  2  3

...Program finished with exit code 3
Press ENTER to exit console.
```

Q2. Calculate the square root of a number n without using predefined functions (library functions).

```

1  #include<stdio.h>
2
3  void main()
4  {
5      int n;
6
7      float temp, squareroot;
8
9      printf("Provide the number: ");
10
11     scanf("%d", &n);
12     squareroot = n / 2;
13     temp = 0;
14
15     while(squareroot != temp){
16         temp = squareroot;
17         squareroot = ( n/temp + temp) / 2;
18     }
19
20     printf("The square root of '%d' is '%f'", n, squareroot);
21 }
22

```

input

```

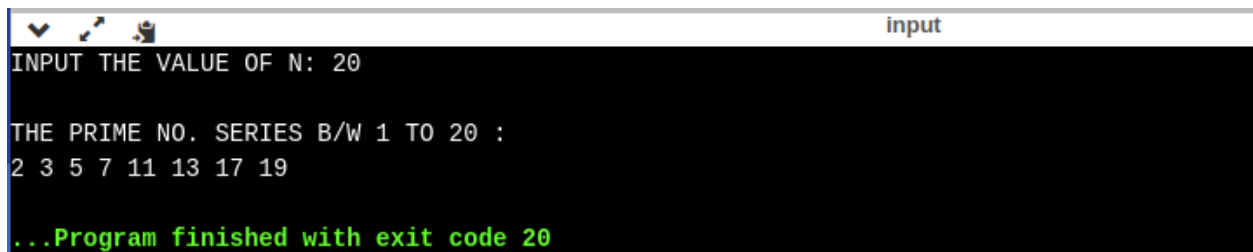
Provide the number: 36
The square root of '36' is '6.000000'

...Program finished with exit code 37
Press ENTER to exit console.

```

Q3. Print all prime numbers between 0 and n. Take n as input.

```
1  #include <stdio.h>
2
3  void main()
4  {
5      int no, counter, counter1, check;
6      printf("INPUT THE VALUE OF N: ");
7      scanf("%d", &no);
8      printf("\nTHE PRIME NO. SERIES B/W 1 TO %d : \n", no);
9
10     for(counter = 2; counter <= no; counter++)
11     {
12         check = 0;
13         for(counter1 = counter-1; counter1 > 1; counter1--)
14             if(counter%counter1 == 0)
15             {
16                 check++;
17                 break;
18             }
19         if(check == 0)
20             printf("%d ", counter);
21     }
22 }
```



```
input
INPUT THE VALUE OF N: 20

THE PRIME NO. SERIES B/W 1 TO 20 :
2 3 5 7 11 13 17 19

...Program finished with exit code 20
```

Q4. Write a program to calculate the sum of the digits of an integer.

```

1  #include <stdio.h>
2
3  void main()
4  {
5      int num, k = 1, sum = 0;
6      printf("Enter the number whose digits are to be added:");
7      scanf("%d", &num);
8      while (num != 0)
9      {
10         k = num % 10;
11         sum = sum + k;
12         k = num / 10;
13         num = k;
14     }
15     printf("Sum of the digits:%d", sum);
16 }
17
18
19

```

input

Enter the number whose digits are to be added:5461
Sum of the digits:16

...Program finished with exit code 20
Press ENTER to exit console.

Q5. Check whether the given number is Armstrong or not.

```

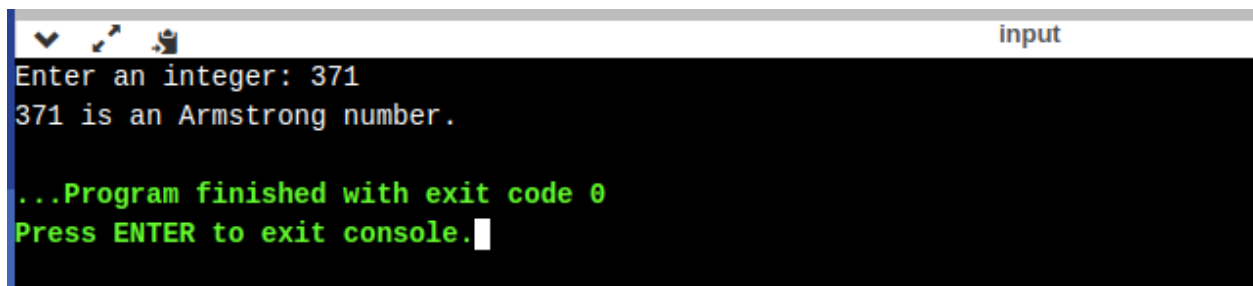
1  #include <math.h>
2  #include <stdio.h>
3
4  int main() {
5      int num, originalNum, remainder, n = 0;
6      float result = 0.0;
7
8      printf("Enter an integer: ");
9      scanf("%d", &num);
10
11     originalNum = num;
12
13     // store the number of digits of num in n
14     for (originalNum = num; originalNum != 0; ++n) {
15         originalNum /= 10;
16     }
17
18     for (originalNum = num; originalNum != 0; originalNum /= 10) {
19         remainder = originalNum % 10;
20
21         // store the sum of the power of individual digits in result
22         result += pow(remainder, n);
23     }
24

```

```

24
25 // if num is equal to result, the number is an Armstrong number
26 if ((int)result == num)
27     printf("%d is an Armstrong number.", num);
28 else
29     printf("%d is not an Armstrong number.", num);
30 return 0;
31 }
32

```



```

input
Enter an integer: 371
371 is an Armstrong number.

...Program finished with exit code 0
Press ENTER to exit console.

```

Tasks:

Answer any one of the following questions:

(1) Write a C program to find the reverse number of a given positive number.

Hint: a similar while loop as in practice problem 4 (in lab sheet 4) will work.

(2) Write a C program to print all four-digit Armstrong numbers,

Hint: Unlike in practice problem 5 (lab sheet 4), now no need to read num from the user; instead maintain a for-loop `for(num=1000; num<=9999; num++)`. Inside this for-loop, check num is an Armstrong number or not. If yes, print the number.

(3) Print the smallest two consecutive primes whose difference is at least 10;

Hint: This task requires two variables to store two consecutive primes: *curr* (holds a prime number) and *prev* (holds the before prime to curr). Initially, take *prev*=2 and *curr*=3. Run a for-loop as in practice problem 3 (lab sheet 4), where the *counter* starts from 4, but there is no upper limit on the *counter* as in problem 3. The stopping condition of the loop depends on the values of *curr* and *prev*. As in the inner for-loop in problem 3 (lab sheet 4), verify whether the *counter* is a prime. If yes, then update the values of *curr* and *prev* appropriately (as in the Fibonacci program problem 1).