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.

```c
#include <stdio.h>

void main()
{
int num1=0, num2=1,no,counter,fab;

printf("ENTER LENGTH OF SERIES (N) : ");
scanf("%d",&no);

printf("<----FIBONACCI SERIES---->");
printf("\n%d  %d",num1,num2);
for(counter = 1; counter <= no-2; counter++)
{
fab=num1 + num2;
printf("  %d",fab);
num1=num2;
num2=fab;
}
}
```

```
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).

```c
#include<stdio.h>

void main()
{
    int n;

    float temp, squareroot;

    printf("Provide the number: ");

    scanf("%d", &n);
    squareroot = n / 2;
    temp = 0;

    while(squareroot != temp){
        temp = squareroot;
        squareroot = ( n/temp + temp) / 2;
    }

    printf("The square root of '%d' is '%f'", n, squareroot);
}
```

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.

```c
#include <stdio.h>

void main()
{
int no,counter,counter1,check;
printf("INPUT THE VALUE OF N: ");
scanf("%d",&no);
printf("\nTHE PRIME NO. SERIES B/W 1 TO %d : \n",no);

for(counter = 2; counter <= no; counter++)
{
 check = 0;
 for(counter1 = counter-1; counter1 > 1 ; counter1--)
   if(counter%counter1 == 0)
   {
    check++;
    break;
   }
   if(check == 0)
   printf("%d ",counter);
}
}
```

```
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.

```c
#include <stdio.h>

void main()
{
    int num, k = 1, sum = 0;
    printf("Enter the number whose digits are to be added:");
    scanf("%d", &num);
    while (num != 0)
    {
            k = num % 10;
            sum = sum + k;
            k = num / 10;
            num = k;
    }
    printf("Sum of the digits:%d", sum);
}
```

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.

```c
#include <math.h>
#include <stdio.h>

int main() {
   int num, originalNum, remainder, n = 0;
   float result = 0.0;

   printf("Enter an integer: ");
   scanf("%d", &num);

   originalNum = num;

   // store the number of digits of num in n
   for (originalNum = num; originalNum != 0; ++n) {
      originalNum /= 10;
   }

   for (originalNum = num; originalNum != 0; originalNum /= 10) {
      remainder = originalNum % 10;

     // store the sum of the power of individual digits in result
     result += pow(remainder, n);
   }
```

```
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).