Birla Institute of Technology,

Off Campus Deoghar

PROGRAMMING FOR PROBLEM SOLVING LAB

NAME:	AKASH DIP
ROLL NUMBER:	BTECH/60002/20
SEMESTER:	2ND
BRANCH:	COMPUTER SCIENCE ENGINEERING

Birla Institute of Technology, Mesra

Off Campus Deoghar

PROGRAMMING FOR PROBLEM SOLVING LAB

INDEX

S. NO.	ASSIGNMENT NO.	DATE	PAGE NO.	REMARKS
1.	**ASSIGNMENT 1**	**28/04/2021**	**Turned In**	
2.	**ASSIGNMENT 2**	**05/05/2021**	**2**	

Akash Dip
BTECH/60002/20
CSE 2020
btech60002.20@bitmesra.ac.in

ASSIGNMENT NUMBER – 2

[DATE: 05/05/2021]

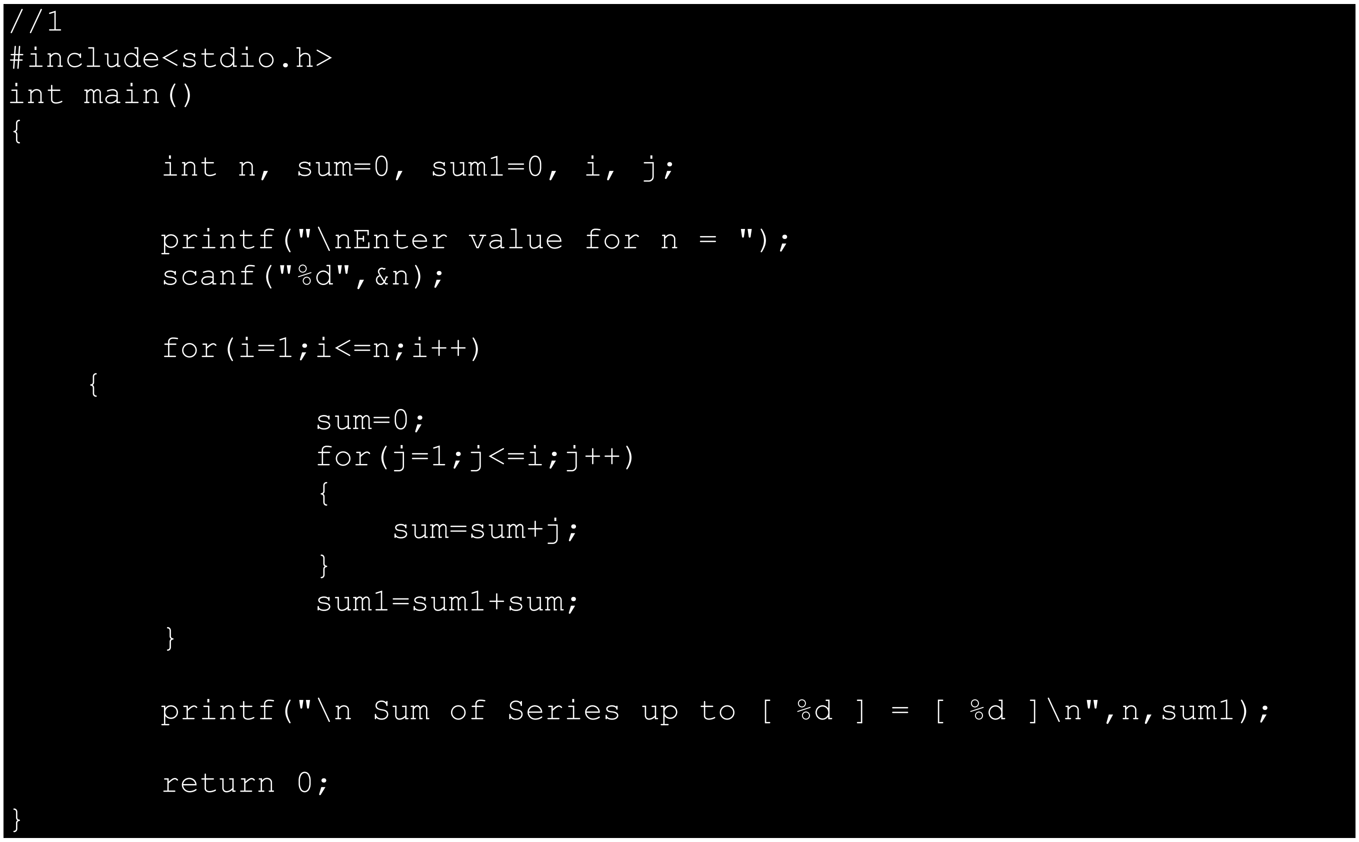
PROBLEM NO.	PROBLEM STATEMENT	PAGE NO.	REMARKS
1	Write a C program to sum the following series: S=1+(1+2) +(1+2+3) + ... +(1+2+3+.... + N)	3	
2	Write a C program to check whether the given number is an Armstrong number.	4	
3	Write a C program to print the perfect number between 1 to 1000.	5	
4	Write a C program to count the number of digits of a given number. [Number should be user input].	6	
5	Print the pattern up to 5 rows: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	7	
6	Write a C program to print the sum of the numbers between 1 to 50 which are divisible by 3 but not 4.	8	
7	Write a C program to find the GCD (Greatest Common Divisor) of two numbers. Draw the flowchart of the program also.	9	

PROGRAMMING LANGUAGE USED: **C**

Problem Number: 1

Problem Statement: Write a C program to sum the following series:
S=1+(1+2) +(1+2+3) +....... +(1+2+3+.... + N)

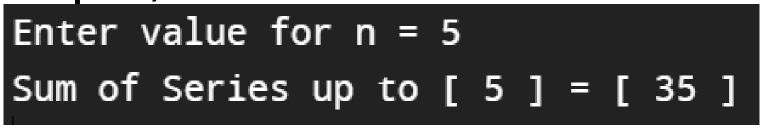
Solution:



```c
//1
#include<stdio.h>
int main()
{
        int n, sum=0, sum1=0, i, j;

        printf("\nEnter value for n = ");
        scanf("%d",&n);

        for(i=1;i<=n;i++)
    {
                sum=0;
                for(j=1;j<=i;j++)
                {
                    sum=sum+j;
                }
                sum1=sum1+sum;
        }

        printf("\n Sum of Series up to [ %d ] = [ %d ]\n",n,sum1);

        return 0;
}
```

Output Discussion:

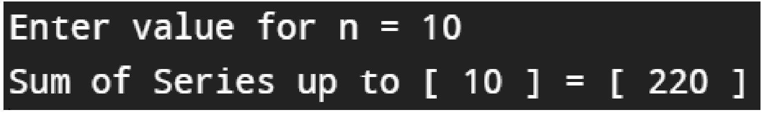
Sample I/O 1:



```
Enter value for n = 5
Sum of Series up to [ 5 ] = [ 35 ]
```

Explanation: `(1) + (1+2) + (1+2+3) + (1+2+3+4) + (1+2+3+4+5) = 35`

Sample I/O 2:



```
Enter value for n = 10
Sum of Series up to [ 10 ] = [ 220 ]
```

Explanation: `(1) + (1+2) + (1+2+3) + .... +(1+2+3+4+ .... +10) = 220`

Problem Number: 2

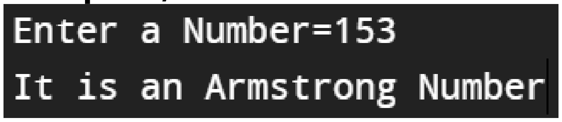
Problem Statement: Write a C program to check whether the given number is an Armstrong number.

Solution:

```c
//2
#include<stdio.h>
int main()
{
    int n,r,sum=0,temp;
    printf("Enter a Number=");
    scanf("%d",&n);
    temp=n;
    while(n>0)
    {
        r=n%10;
        sum=sum+(r*r*r);
        n=n/10;
    }
    if(temp==sum)
        printf("It is an Armstrong Number");
    else
        printf("It is not an Armstrong Number");
    return 0;
}
```

Output Discussion:

Sample I/O 1:



```
Enter a Number=153
It is an Armstrong Number
```

Explanation:

153 = (1*1*1) + (5*5*5) + (3*3*3)

where:

(1*1*1) = 1

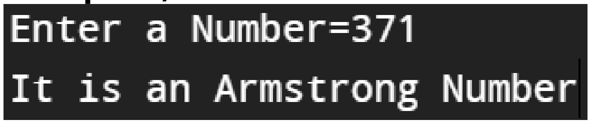
(5*5*5) = 125

(3*3*3) = 27

So: 1+125+27 = 153

And hence 153 is an Armstrong Number

Sample I/O 2:



```
Enter a Number=371
It is an Armstrong Number
```

Explanation:

371 = (3*3*3) + (7*7*7) + (1*1*1)

where:

(3*3*3) = 27

(7*7*7) = 343

(1*1*1) = 1

So: 27+343+1 = 371

And hence 371 is an Armstrong Number

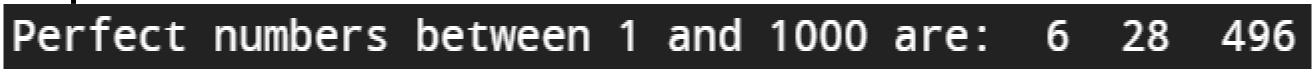
Problem Number: 3

Problem Statement: Write a C program to print the perfect number between 1 to 1000.

Solution:

```
//3
#include<stdio.h>
int main()
{
    int sum=0,p,i;
    printf("\n Perfect numbers between 1 and 1000 are: ");
    for(i= 1; i<= 1000; i++){
    p=1;
    while(p<=(i/2))
    {
        if(i % p == 0)
            sum=sum+p;
        p++;
    }
    if(sum==i)
        printf(" %d ",i);
    sum=0;
    }
    return 0;
}
```

Output Discussion:



```
Perfect numbers between 1 and 1000 are:  6  28  496
```

Explanation: As we know a number is said to be a perfect number if the sum of the factors excluding the number itself is equal to the given number and hence sum of factors of 6 i.e., 1+2+3 is equal to 6, similarly 28 = 1+2+4+7+14 and 1+2+4+8+16+31+62+124+248 = 496.

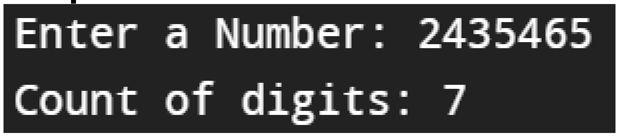
Problem Number: 4

Problem Statement: Write a C program to count the number of digits of a given number. [Number should be user input].

Solution:

```
//4b
#include <stdio.h>
int main()
{
   long n;
   int count=0;
   printf("Enter a Number: ");
   scanf("%ld",&n);
   while(n!=0)
   {
       count++;
       n=n/10;
   }

   printf("\n Count of digits: %d",count);
    return 0;
}
```

Output Discussion:



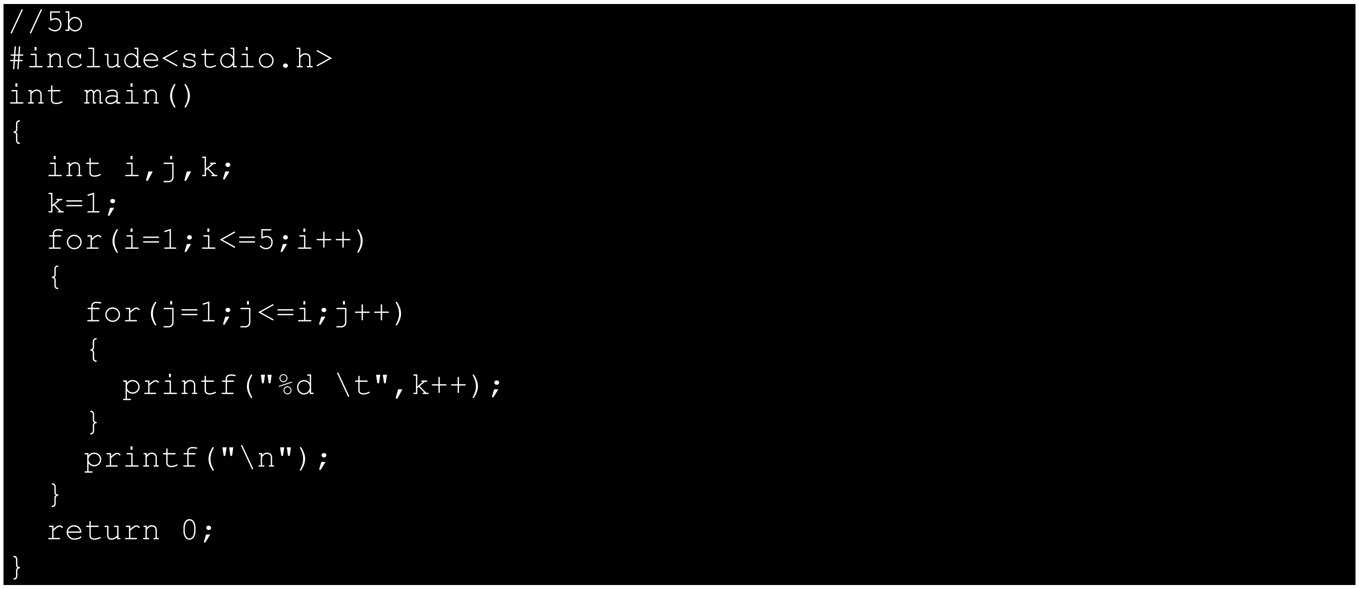
```
Enter a Number: 2435465
Count of digits: 7
```

Problem Number: 5

Problem Statement: Print the pattern up to 5 rows:

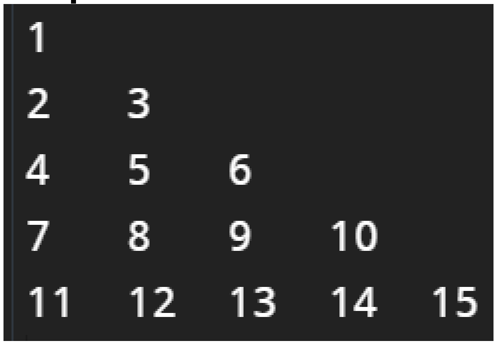
```
 1
 2    3
 4    5    6
 7    8    9    10
 11   12   13   14   15
```

Solution:



```
//5b
#include<stdio.h>
int main()
{
  int i,j,k;
  k=1;
  for(i=1;i<=5;i++)
  {
    for(j=1;j<=i;j++)
    {
      printf("%d \t",k++);
    }
    printf("\n");
  }
  return 0;
}
```

Output Discussion:



```
1
2     3
4     5     6
7     8     9     10
11    12    13    14    15
```

The program uses two iterations to achieve the above pattern and also **\t** i.e., escape sequence for aligning the pattern.

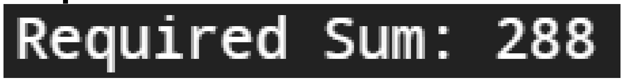
Problem Number: 6

Problem Statement: Write a C program to print the sum of the numbers between 1 to 50 which are divisible by 3 but not 4.

Solution:

```
//6
#include<stdio.h>
int main()
{
  int i,sum=0;
  for(i=1;i<50;i++)
  {
    if(i%3==0 && !(i%4==0))
    sum += i;
  }
  printf("Required Sum: %d",sum);
  return 0;
}
```

Output Discussion:



```
Required Sum: 288
```

The program sums up the numbers between 1 to 50 which are divisible by 3 but not 4 i.e., 3 + 6 + 9 + 15 + 18 + 21 + 27 + 30 + 33 + 39 + 42 + 45 = **288**

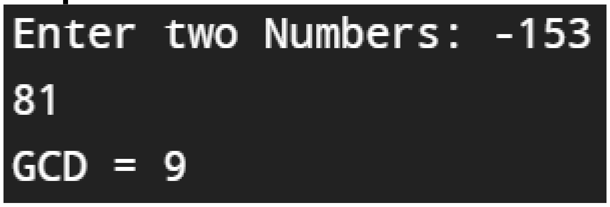
Problem Number: 7

Problem Statement: Write a C program to find the GCD (Greatest Common Divisor) of two numbers. Draw the flowchart of the program also.

Solution:

```
//7
#include <stdio.h>
int main()
{
    int n1, n2;

    printf("Enter two Numbers: ");
    scanf("%d %d",&n1,&n2);

    //if user enters -ve number then the sign is reversed
    n1 = (n1 > 0) ? n1 : -n1;
    n2 = (n2 > 0) ? n2 : -n2;

    while(n1!=n2)
    {
        if(n1 > n2)
            n1 -= n2;
        else
            n2 -= n1;
    }
    printf("GCD = %d",n1);

    return 0;
}
```

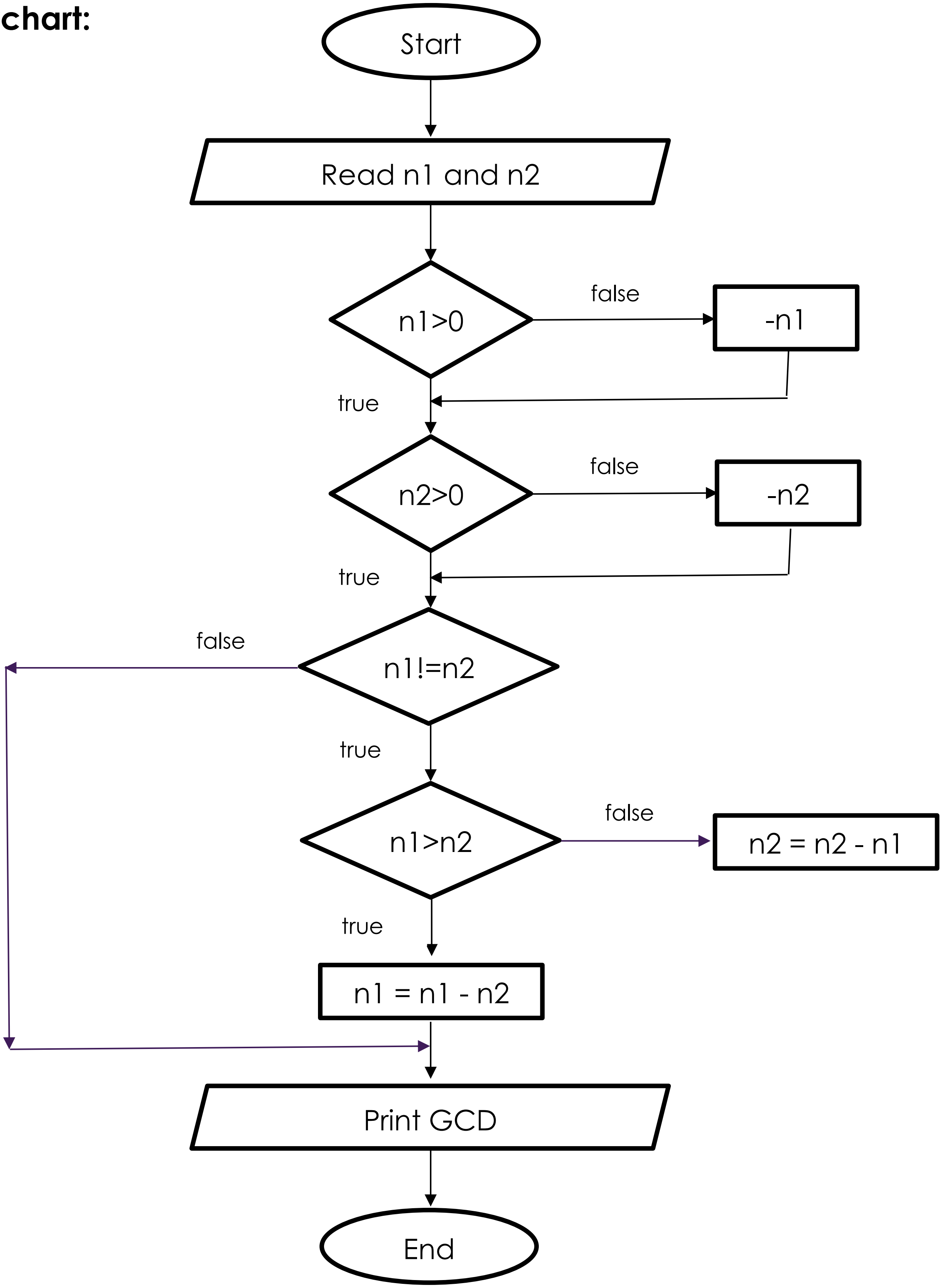
Output Discussion:



```
Enter two Numbers: -153
81
GCD = 9
```

The program first reverses the negative value (if any) to positive and then runs an iteration to find out the GCD and then finally prints the result.

Flowchart:


Start
Read n1 and n2
n1>0
false
-n1
true
n2>0
false
-n2
true
n1!=n2
false
true
n1>n2
false
n2 = n2 - n1
true
n1 = n1 - n2
Print GCD
End