SAP ID - 60004200182
Name - Ayush Jain
Div - J1

CP Lab Experiments

Q.1) **Aim:** To calculate simple interest taking principal, rate of interest and number of years as input from user. Write algorithm and draw flowchart for the same.

Theory: 1) Float is a datatype which is used to represent the floating point numbers.

2) The printf() function is used to display output and the scanf() function is used to take input from users.

3) Simple interest can be calculate as:

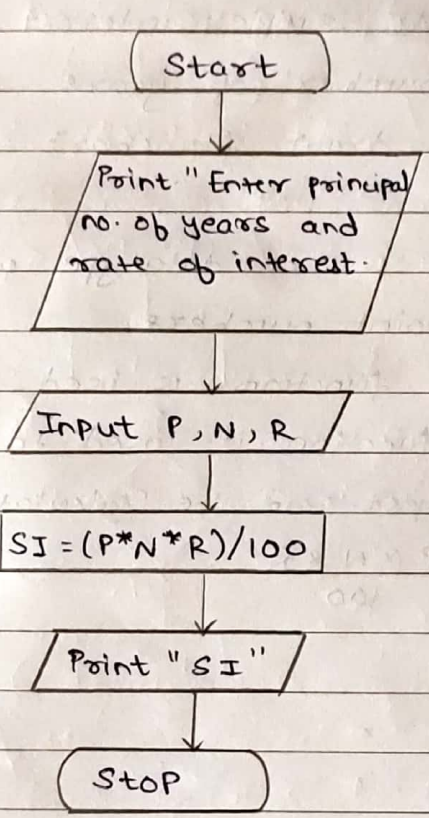
$$SI = \frac{P \times N \times R}{100}$$

Algorithm:

P: Principal amount
N: Time in years
R: Rate of interest
SI: Simple Interest.

Step 1 : start
Step 2 : Print "Enter principal, no. of years and rate of interest"
Step 3 : Input P, N, R
Step 4 : SI = (P*N*R)/100
Step 5 : Display SI
Step 6 : Stop

Flowchart :



Program:

```
#include <stdio.h>
int main()
{
float P,N,R,SI;
printf("Enter Principal,no.of years,rate of interest\n");
scanf("%f %f %f",&P,&R,&N);
SI=(P*N*R)/100;
printf("Simple Interest = %f",SI);
return 0;
}
```

Output:

```
Enter Principal,rate of interest and no. of years
1300
5.6
2.5
Simple Interest = 182.000000
```

Q. 2 Aim : Write a program to find greatest of three numbers using conditional operators. Write algorithm and draw flowchart for the same.

Theory :

1) The conditional operator is also known as a ternary operator. The conditional statements are the decision making statements which depends upon the output of the expression. It is represented by two symbols i.e. '?' and ':'.

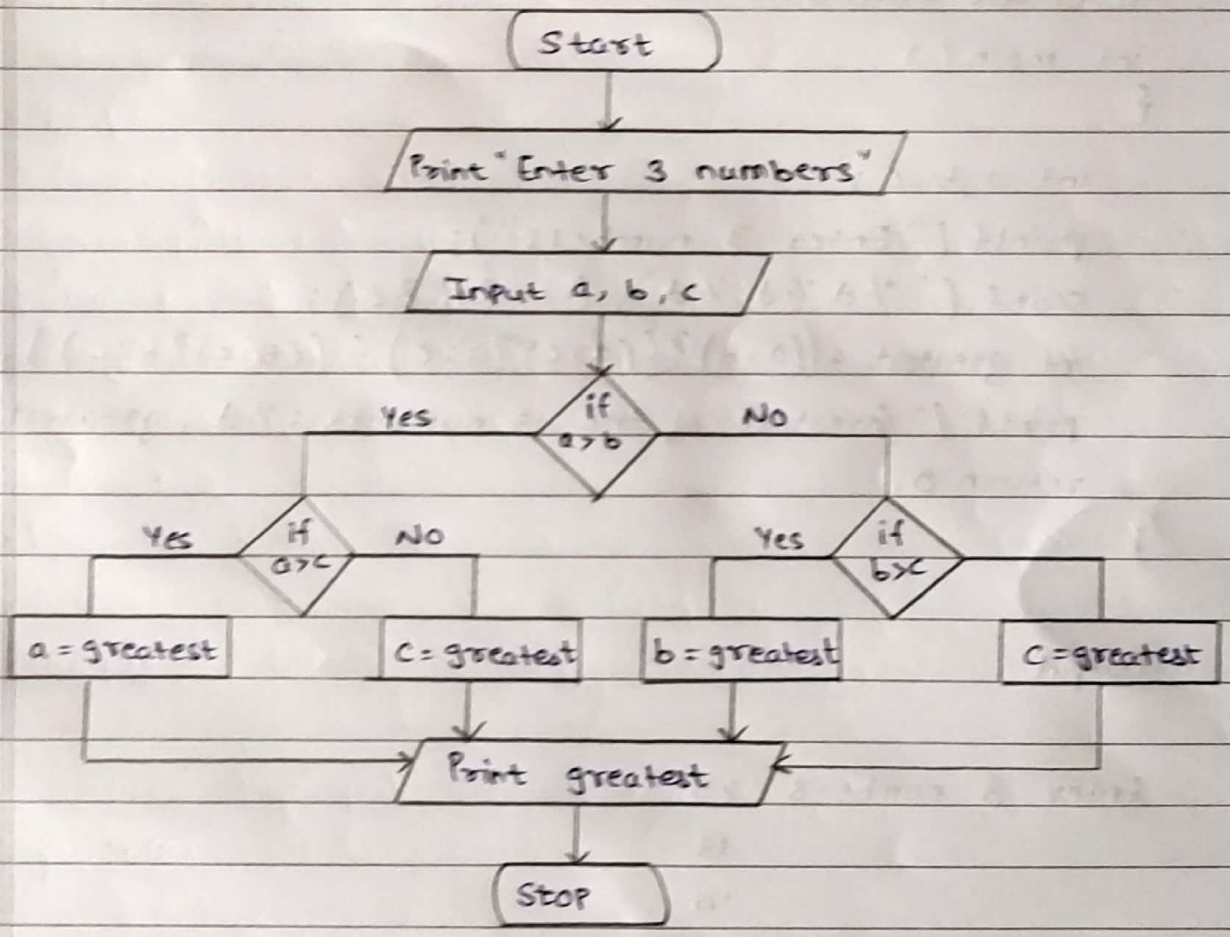
2) As conditional operator works on three operands, so it is also known as the ternary operator.

Syntax : Expression 1 ? expression 2 : expression 3 ;

Algorithm

Step 1 : Start
Step 2 : Display "Enter 3 numbers"
Step 3 : Input a, b, c
Step 4 : If a > b and a > c then a = greatest
else b > c then b = greatest
else c = greatest
Step 5 : Print greatest
Step 6 : Stop

Flowchart:



Program:

```
#include <stdio.h>
int main()
{
    int a,b,c;
    printf("Enter 3 numbers");
    scanf("%d %d %d",&a,&b,&c);
    int greatest =((a>b)?((a>c)?a:c) : ((b>c)?b:c));
    printf("Greatest of the 3 numbers : %d , greatest);
    return 0;
}
```

Output

```
Enter 3 numbers : 25
                  13
                  14
Greatest of the 3 numbers : 25
```

Aim:

Q. 3) Write a program to check if the year entered is leap year or not. Write algorithm and draw flowchart for the same.

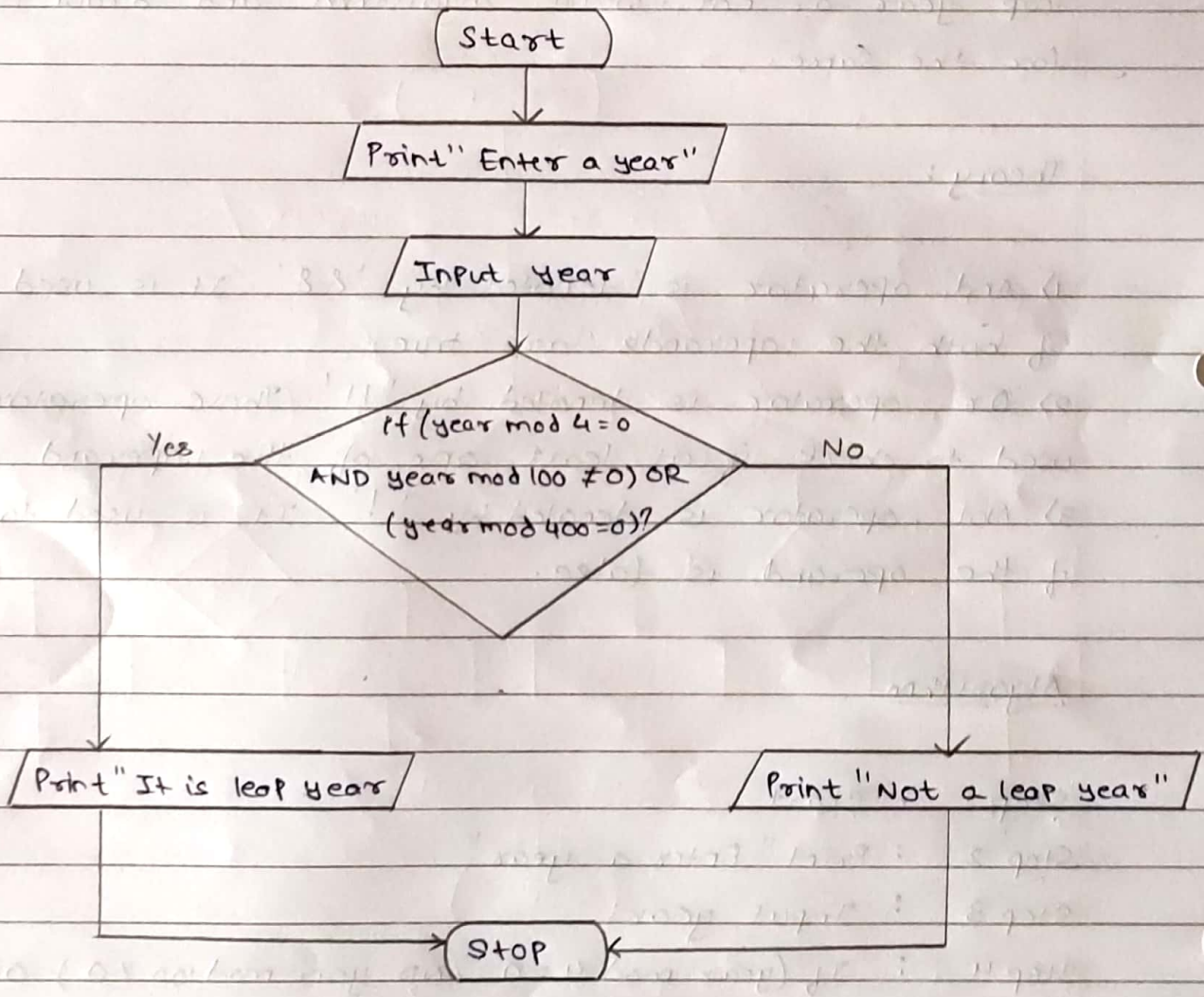
Theory:

1) And operator is denoted by '&&'. It is used to check if both the operands are true.
2) Or operator is denoted by '||'. These operators are used to check if at least one of the operand is true.
3) Not operator is denoted by '!'. It is used to check if the operand is false.

Algorithm:

Step 1 : Start
Step 2 : Print "Enter a year".
Step 3 : Input year.
Step 4 : If (year mod 4 = 0 AND year mod 100 ≠ 0) OR
(year mod 400 = 0)
then Print "It is a leap year"
else "Not a leap year".
Step 5 : Stop

Flowchart



Program

```
#include <stdio.h>
int main()
{
   int year;
   printf("Enter a year");
   scanf("%d", &year);
   if((year%4==0 && year%100!=0) || (year%400==0))
      {
        Printf("It is a leap year\n");
      }
   else
      {
        printf("It is not a leap year");
      }
    return 0;
}
```

Output:

```
Enter a year: 2024
It is a leap year.
```

Q.4 Aim: Write a program to calculate roots of a quadratic equation.

Theory:

1) Double is a keyword for the double data type. It represents floating point numbers with better precision.
2) The sqrt() function in C programming language returns the square root value.

Program:

```
# include <stdio.h>
# include <math.h>
int main ()
{
    double r1, r2;
    int a, b, c;
    Printf (" Enter coefficient a,b,c of quadratic equation
            ax^2+bx+c ");
    scanf (" %d %d %d", &a, &b, &c);
    int d = (b*b) - (4*a*c);
    if (d>0)
{
    r1 = -(double)b/(2*a) + sqrt(d)/(double)a*2.0);
    r2 = -(double)b/(2.0*a) - sqrt(d)/(double)a*2.0);
    Printf(" Roots are real and unequal and they are \n %.f
            \n %.f ", r1, r2);
}
```

```
else if(d==0)
{
   r1 = -(double)b /(2.0*a);
   printf ("Roots are real and equal and it is %f", r);
}
else
{
   printf ("Roots are imaginary \n");
}
return 0;
}
```

Output :

Enter coefficients a, b, c of quadratic equation ax^2+bx+c :

1
-4
3

Roots are real and unequal and they are

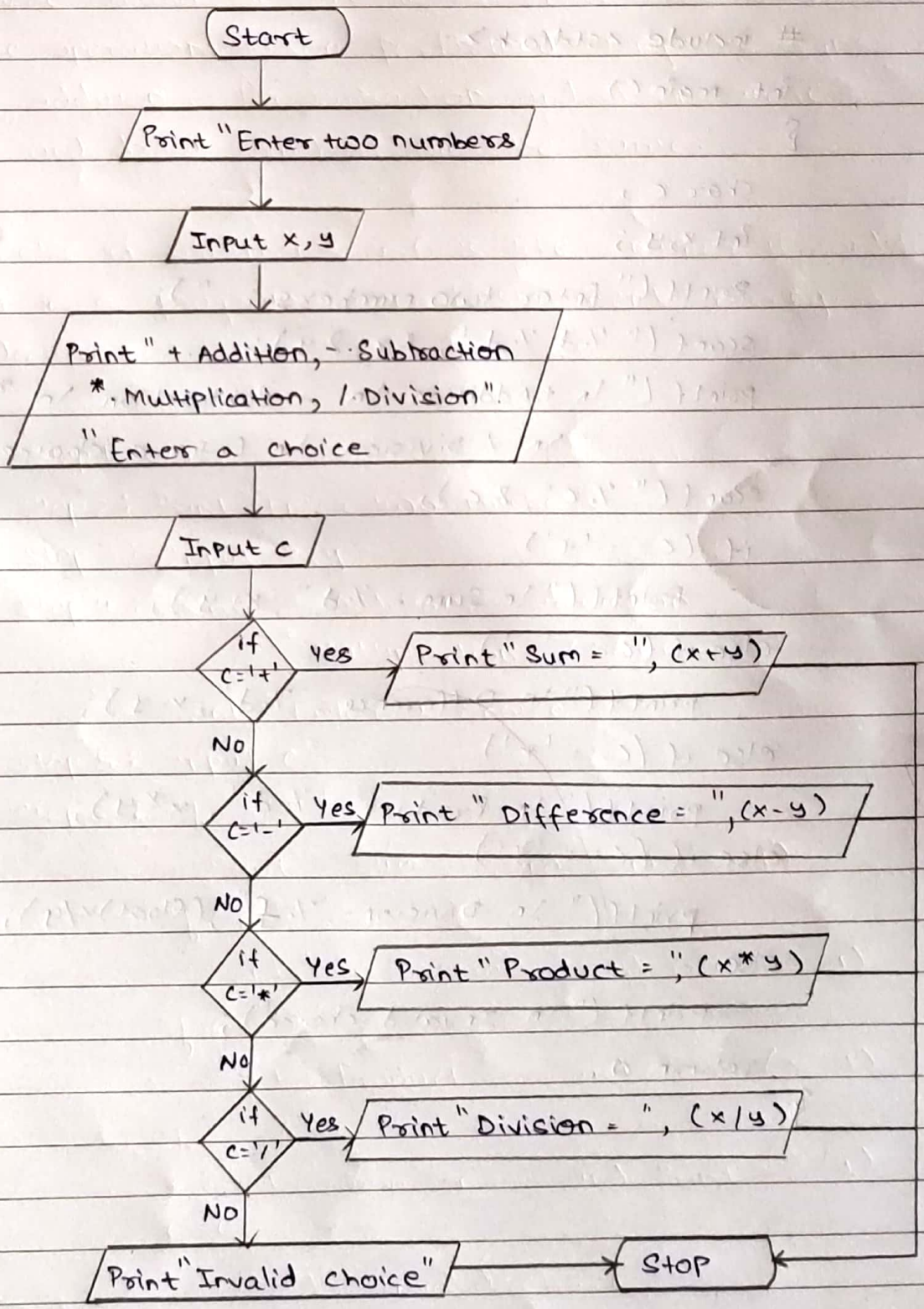
3.00
1.00

Q.5 Aim: Write a menu driven program to perform add/subtract multiply/divide based on the users choice. The user will indicate the operation to be performed using the signs i.e. + for addition, - for subtraction and so on. Write algorithm and draw flowchart for the same.

Theory: i) else-if statements in C is like another if condition. It is used in a program when if statement having multiple decisions.

Algorithm: Step 1 : Start
Step 2 : Print " Enter two numbers "
Step 3 : Input x, y
Step 4 : Print " + Addition, - Subtraction, * Multiplication, / Division "
Print " Enter a choice"
Step 5 : Input c
Step 6 : If c = '+'
Print " Sum = " and print (x+y)
If c = '-'
Print " Difference = " and Print (x-y)
If c = '*'
Print " Product = " and Print (x*y)
If c = '/'
Print " Division : " and Print (x/y)
else
Print " Invalid choice".
Step 7 : Stop

Flowchart :



Program:

```
# include <stdio.h>
int main()
{
   char c;
   int x,y;
   Printf(" Enter two numbers : ");
   scanf(" %d %d ", &x, &y);
   printf(" \n +.Addition \n -.Subtraction \n *. Multiplication
            \n /. Division \n Enter a choice");
   scanf(" %c", &c);
   if (c==='+')
      Printf(" \n Sum = %d", x+y);
   else if (c==='-')
      printf(" \n Difference = %d", x-y);
   else if (c==='*')
      printf(" \n Product = %d", x*y);
   else if (c==='/')
      printf(" \n Division= %f", (float)x/y);
   else
      printf(" \n Invalid choice");
   return 0;
}
```

Output :

```
Enter two numbers : 5  4
+ . Addition
- . Subtraction
* . ~~Divi~~Product
/ . Division
Enter a choice : /
Division = 1.250000
```

Q. 6 Aim : Write a program to find the sum of seven terms using a for loop for the following series

$$\frac{1}{1!} + \frac{2}{2!} + \frac{3}{3!} + \dots \frac{7}{7!}$$

Theory :

1) A for loop is a repetition control structure which allows us to write a loop that is executed a specific number of times. The loop enables us to perform n number of steps together in one line.

Program :

```
#include <stdio.h>
int main()
{
  float fact=1, sum=0;
  for (float i=1; i<=7; i++)
    {
      fact*=i;
      sum=sum+(i/fact);
    }
    printf("Sum of first 7 terms of series : %f ", sum);
}
```

Output :

Sum of first 7 terms of series : 2.718055

Q. 7) Aim : Write a program to read a number, reverse the number and display the sum of digits of number.

Theory : i) while loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed multiple times depending upon a given boolean condition.

Program :

```
# include <stdio.h>
int main ()
{
   int n, rev=0, sum=0, a;
   Printf ("Enter a number: ");
   scanf ("%d", &n);
   while (n>0) {
        a = n%10;
      rev = rev*10 + a;
      sum = sum + a;
      n = n/10;
    }
Printf ("\n Reverse of the number is %d", rev);
Printf ("\n Sum of digits is %d", sum);
}
```

Output :

```
Enter a number : 1234
Reverse of the number is 4321
Sum of digits is 10
```

Q. 8) Aim: Write a program using for loop to display the following asking the user for the number of lines. Write algorithm and flowchart for the same.

```
    A
  A B A
A B C B A
  ⋮
```

Theory: Pattern programs are nothing but patterns consisting of numbers, alphabets or symbols in a particular form. These kind of pattern programs can be solved easily using for loop condition.

Algorithm:

Step 1: Start
Step 2: Print "Enter number of lines"
Step 3: Input n
Step 4: i = 1
Step 5: If i > n then go to step 15
Step 6: j = 1
Step 7: If j < n - i then print " "
j++, go to step 7
Else go to step 8.
Step 8: k = 1
Step 9: If k <= i then print (k + 64) type casted to char
k++, go to step 9.
Step 10: l = i - 1

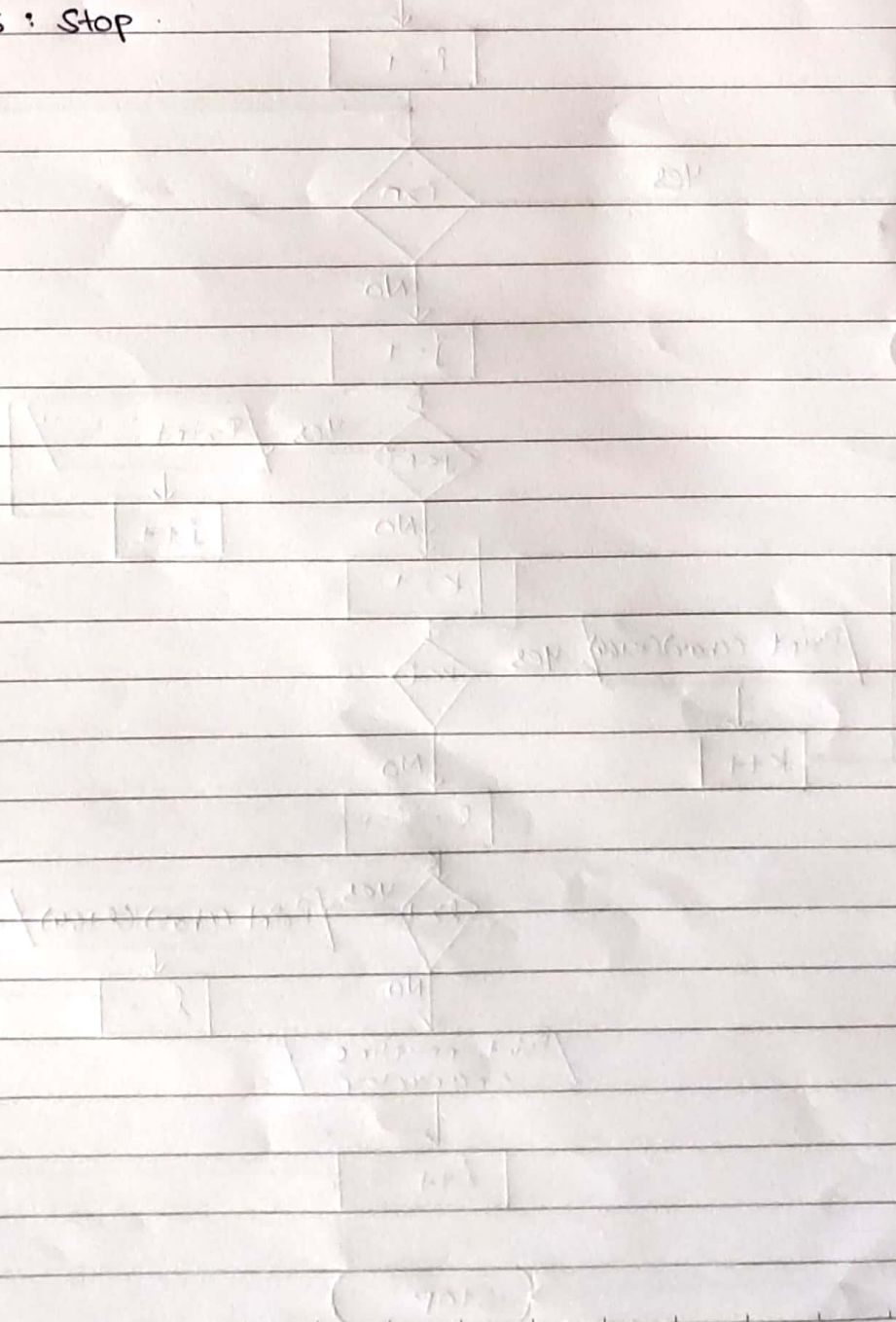
Step 11 : If $l > 0$ then print $(l+64)$ type casted to char
$l--$, go to step 11
else go to step 12

Step 12 : $i++$

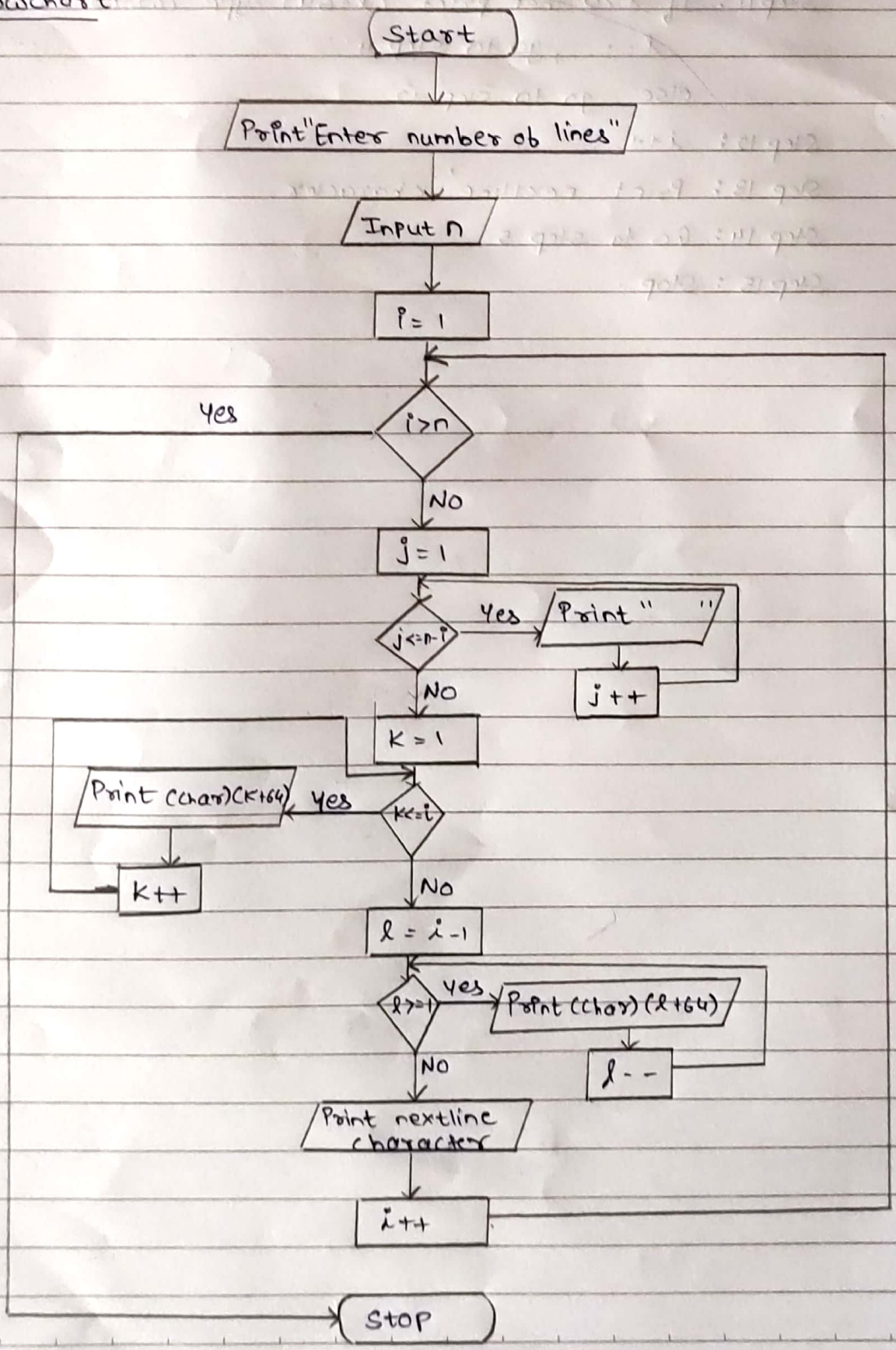
Step 13 : Print nextline character.

Step 14 : Go to step 5

Step 15 : Stop



Flowchart:



Program:

```
#include <stdio.h>
int main()
{
  int n;
  Printf("Enter number of lines : ");
  Scanf("%d", &n);
  for(int i=1; i<=n; i++)
  {
    for(int j=1; j<=n-i; j++)
    { printf(" "); }
    for(int k=1; k<=i; k++)
    { printf("%c", k+64); }
    for(int l=i-1; l>=1; l--)
    { printf("%c", l+64); }
    printf("\n");
  }
return 0;
}
```

Output:

```
Enter number of lines : 4
      A
    A B A
  A B C B A
A B C D C B A
```

Q. 9) Aim: Write a program to check if the entered number is Armstrong or not. Write Algorithm and draw flowchart.

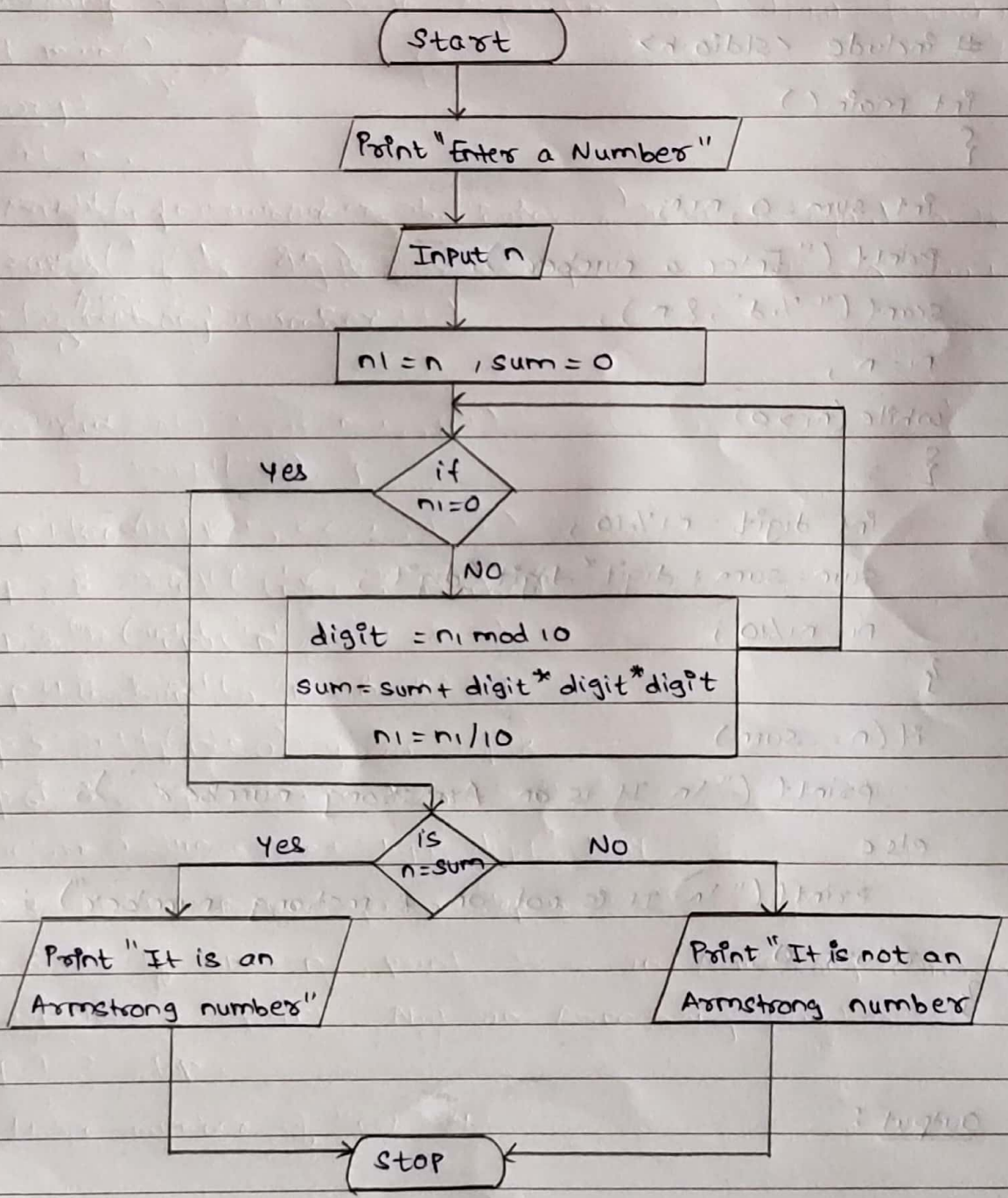
Theory:

1) Armstrong number is a number that is equal to the sum of cubes of its digits. For example: 153, 371, 370, 407 are the armstrong number.

Algorithm:

Step 1 : start
Step 2 : Print "Enter a number"
Step 3 : Input n
Step 4 : Sum = 0, n1 = n
Step 5 : If n1 = 0, go to step 8
Step 6 : digit = n1 mod 10
Sum = Sum + digit* digit* digit
Step 7 : n1 = n1/10, go to step 5
Step 8 : If Sum = n, print "It is an Armstrong number"
else print "Not an Armstrong number".
Step 9 : Stop.

Flowchart:



Program:

```
#include <stdio.h>
int main()
{
   int sum=0, n1,n;
   printf ("Enter a number");
   scanf ("%d", &n);
   n1=n;
   while (n1>0)
   {
      int digit = n1%10;
      sum = sum + digit*digit*digit;
      n1 = n1/10;
   }
   if (n==sum)
      printf ("\n It is an Armstrong number");
   else
      printf ("\n It is not an Armstrong number");
   return 0;
}
```

Output:

```
Enter a number : 153
It is an Armstrong number.
```

Q. 10) **Aim:** Write a Program to display first n elements of Fibonacci series (using 'for')

Theory: The Fibonacci sequence is a series of numbers where a number is the addition of the last two numbers. The Fibonacci sequence : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ....

Program:

```
#include <stdio.h>
int main()
{
  int a=0, b=1, c, n;
  printf("Enter number of terms: ");
  scanf("%d", &n);
  printf("\n%d %d ,a,b);
  for(int i=3; i<=n; i++)
  {
    c=a+b;
    printf("%d ,c);
    a=b
    b=c;
  }
  return 0;
}
```

Output:

Enter number of terms: 8

0, 1, 1, 2, 3, 5, 8, 13