CSO - 101

Name: Maddulapalli Phani Hevambanath
Roll No: 21075050
Branch: Computer Science and Engineering
Section: BA-1
Email ID: mphani.hevambanath.cse21@itbhu.ac.in

1.) $x = -10$

2.) $x = -48$

3.) Output is 1

x=1.

Explanation: Compiler executes from right to left. Since $y == z$ is true (y=5 and z=5) its value is 1. So x is assigned the value "1".

4.) The code will give compilation error. Because in increment operator can only take variables as operand but not expressions.

5.)

Expr_1 = 8

Expr_2 = 65.000000

Expr_3 = 1.000000

6.) Corrected expression : $x * w + y * (z = 7)$

Value of the corrected expression = 35

7.)

Registers: Registers are a type of computer memory used to quickly accept, store and transfer data and instructions that are being used immediately by the CPU. The registers used by the CPU are often termed as "processor registers".

A processor register may hold an instruction, a storage address, or any data (such as bit sequence or individual characters).

Types of Registers:

(1.) MAR - Memory Address Register

The MAR is issued to fetch the instructions and data from the memory and helps to execute the instructions.

→ The Memory Address Register is mainly used for reading and writing operation of data from memory.

(2.) MDR - Memory Data Register.

The memory data register is used to store the data that will be stored or fetched from the computer memory i.e. the Random Access Memory (RAM).

→ The main use of MDR is to act as a buffer as it can store anything that can be copied from the computer memory and can be used by the processor for further operations.

* Memory Buffer Register (MBR), Program Counter (PC), etc.... are examples of some other registers.

8.) We use

* %.x to print value in hexadecimal format (letters will be printed in lower case).
* %.X to print value in hexadecimal format (letters will be printed in upper case).
* %.o to print value in octal format.

Code:

```
#include <stdio.h>
int main()
{
    float x = 123.331;
    printf("%.3f \n %.1f \n %.2f \n %.3e", x, x, x, x);
    return 0;
}
```

9.) → Formatted I/O functions allow to supply input or display output in user desired format.

→ Unformatted I/O functions are the most basic form of input and output and they do not allow to supply input or display output in user desired format.

Formatting characters in C:

%.c - character	%.o - octal representation
%.d - signed integer	%.x - hexadecimal representation
%.u - unsigned integer	%.p - pointer
%.f - float values (real values)	%.s - string

We can write a scanf function on # sign and read two numbers.

```
#include <stdio.h>
int main()
{
    int x,y;
    printf("Enter input:");
    scanf("%d#%d",&x,&y);
    printf("x=%d, y=%d", x,y);
    return 0;
}
```

If input is 22#69

then scanf reads x = 22 and y = 69

So the output will be x=22, y=69.

10.) for loop:

The syntax for the 'for' loop is

```
for (initialization statement ; test expression ; update statement)
{
    // statements inside the body of for loop
}
```

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is false, the for loop is terminated.
- However, if the test expression is evaluated to be true, statements inside the body of the for loop are executed, and the update expression is updated.
- Again the test expression is evaluated.

→ This process goes on until the test expression is false. When the test expression is false, the loop terminates.

Difference between for loop and while loop:

In for loop the number of iterations to be done is already known and is used to obtain a certain result whereas in while loop the command runs until a certain condition is reached and the statement is proved to be false.

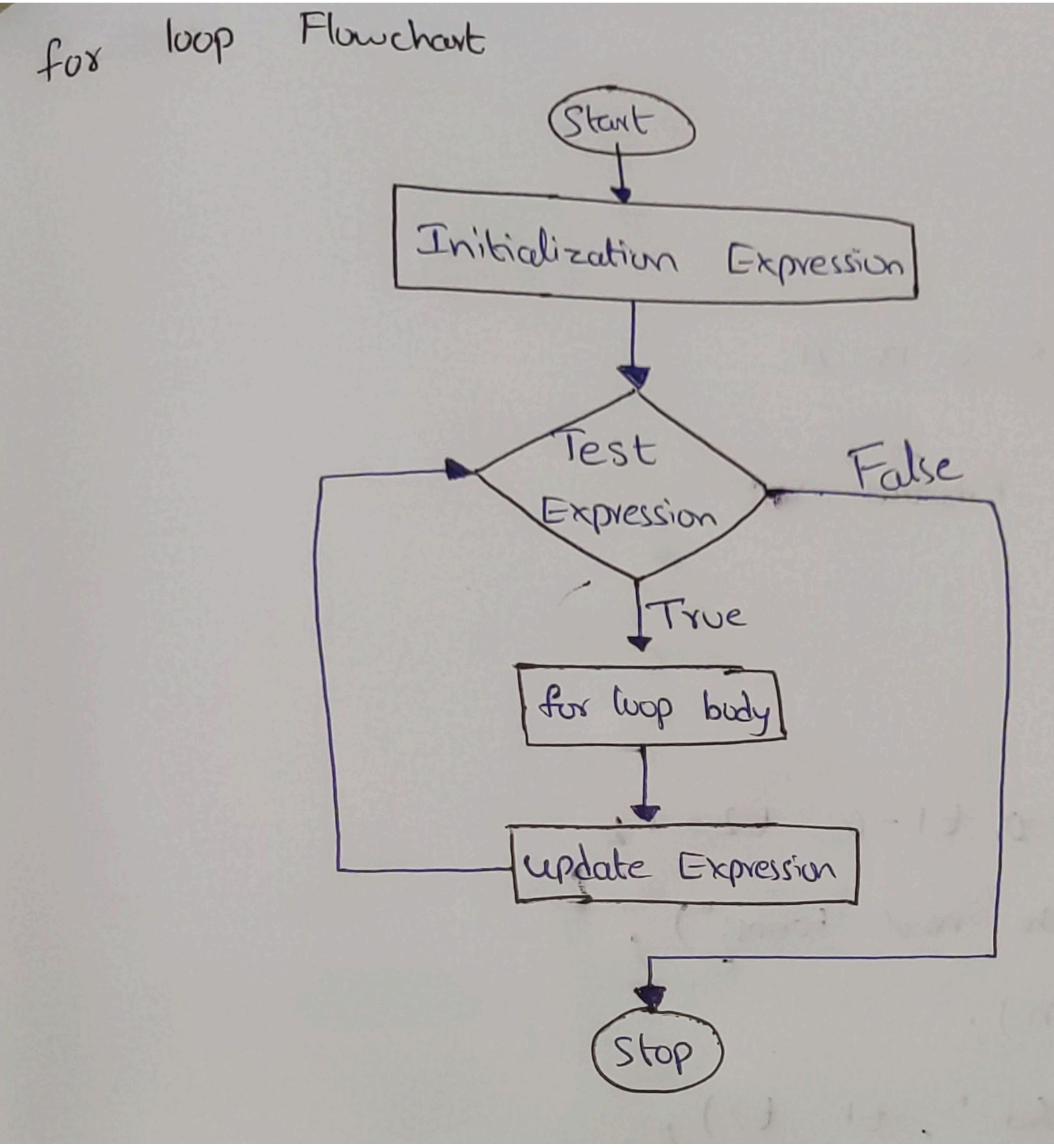
Structure of for loop:

```
for (initialization ; test ; update)
{ // body of the loop }
```

Structure of while loop:

```
while (condition)
{ statements; // body }
```

for loop Flowchart



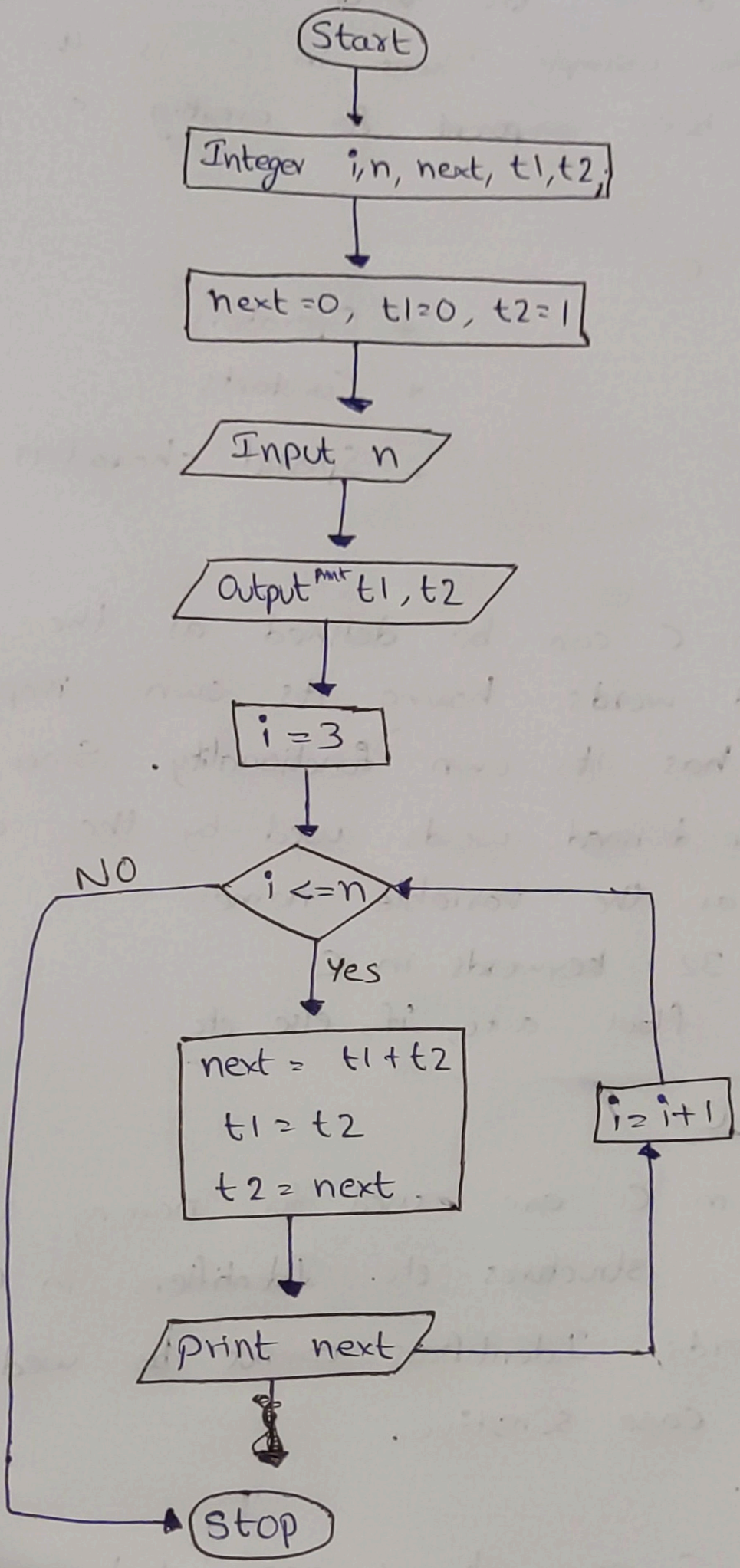
11.) C Program for Fibonacci Series.

```
#include <stdio.h>
int main()
{
    int i, n, next=0, t1=0, t2=1;
    printf("Enter the no. of terms:");
    scanf("%d", &n);
    printf("%d, %d,", t1, t2);
    for (i=3; i<=n; i++)
    {
        next = t1 + t2;
        t1 = t2;
        t2 = next;
        printf("%d, ", next);
    }
    return 0;
}
```

Input: 9

Output: 0, 1, 1, 2, 3, 5, 8, 13, 21

Flowchart:


Start
Integer i,n, next, t1,t2;
next =0, t1=0, t2=1
Input n
Output t1, t2
i =3
i <=n
NO
Yes
next = t1+t2
t1 = t2
t2 = next
print next
i= i+1
Stop

12.) We can define the token as the smallest individual element in C. For example, Tokens in C is the building blocks or the basic component for creating a program in C language.

Types of tokens in C:

* Keywords
* Identifiers
* Strings
* Operators.
* Constants
* Special characters

→ Keywords in C:

Keywords in C can be defined as the pre-defined or the reserved words having its own importance, and each keyword has its own functionality. Since keywords are the pre-defined words used by the compiler, they cannot be used as the variable names.

* There are 32 keywords in C.

Ex: int, float, auto, if, else, etc...

→ Identifiers in C:

Identifiers in C are used for naming variables, functions, arrays, structures, etc. Identifiers in C are the user-defined words. Identifiers cannot be used as keywords. Identifiers are case sensitive.

→ Strings in C:

Strings in C are always represented as an array of characters having null character '\0' at the end of the string. Strings in C are enclosed within double quotes, while characters are enclosed within single quotes.

→ Operators in C:

Operators in C is a special symbol used to perform the functions. The data items on which the operators are applied are known as operands.

Depending on no.of operands, operators are classified into unary and binary operators.

→ Constants in C:

A constant is a value assigned to the variable which will remain the same throughout the program i.e, the constant value cannot be changed.

→ Special Characters in C:

Some special characters are used in C, and they have a special meaning which cannot be used for another purpose.

Ex: Square brackets [] Comma ,
Sim Paranthesis ( ) Hash / pre processor #
Curly braces { } Asterisk *

---

13) The C preprocessor is a macro processor that is used automatically by the C compiler to transform your program before actual compilation. It is called a macro processor because it allows you to define macros, which are brief abbrevations for longer constructs.

→ All preprocessor lines begin with #

The preprocessor provides the ability for the inclusion of header files, macro expansions, conditional compilation, and line control.

In many C implementations, it is a seperate program invoked by the compiler as the first part of translation.

→ The C preprocessor modifies a source file before handling it to the compiler, allowing conditional compilation. with #ifdef, defining constants with # define, including header files with #include. and using builtin macros such as __FILE__.

→ Use of the preprocessor is advantageous since it makes programs -

- easier to develop
- easier to read
- easier to modify
- C code more transportable between different machine architectures.

14.) (a) * The #include preprocessor directive is used to paste code of given file into current file. It is used to include system-defined and user-defined header files.

* The #define directive is used to define values of or macros that are used by the preprocessor to manipulate the program source code before it's compiled.

(b)

* int main() indicates, that the main() can return integer type data.
* void main() indicates that the main() function will not return any value.

→ When our program is simple and it is not going to terminate before reaching the last line of the code, then we can we the void main().

(c) . Declarative programming is a programming paradigm that expresses the logic of a computation without describing its control flow.

. Imperative programming is a programming paradigm that uses statements that change a program's state.

15.)

(a) Nested loop means a loop statement inside another loop statement. That is why nested loops are also called as "loop inside loop".

Eg: Syntax for nested for loop;

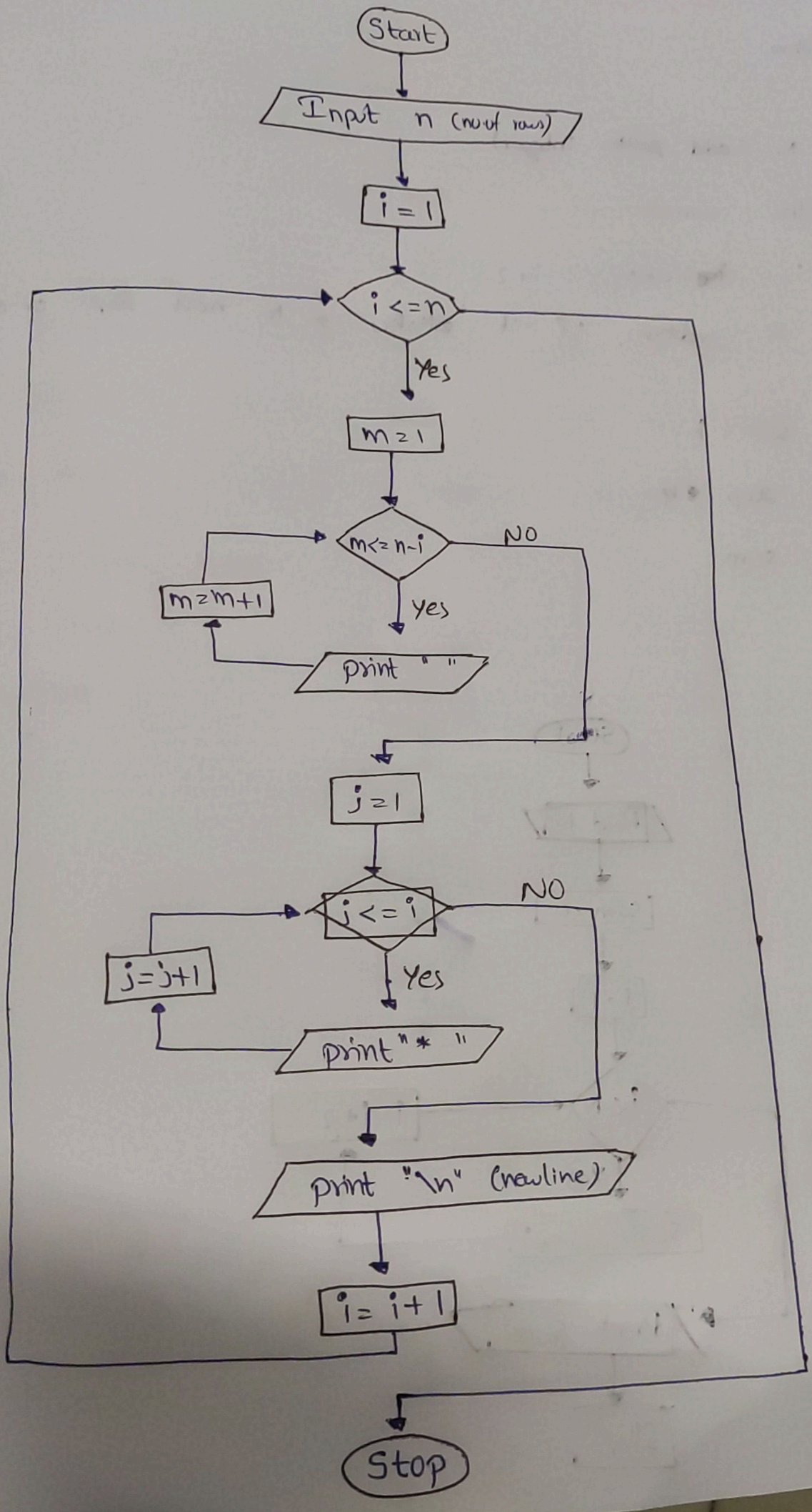
```
for (initialization; condition; update)
{
    for (initialization; condition; update)
    { //statement of inside loop }

  //statement of outer loop
}
```

(b) Algorithm

1. Start
2. Read n (no. of rows)
3. For i = 1 to n
4. For m = 1 to n-i
5. print " "
6. For j = 1 to i
7. print "* "
8. Stop

Flowchart

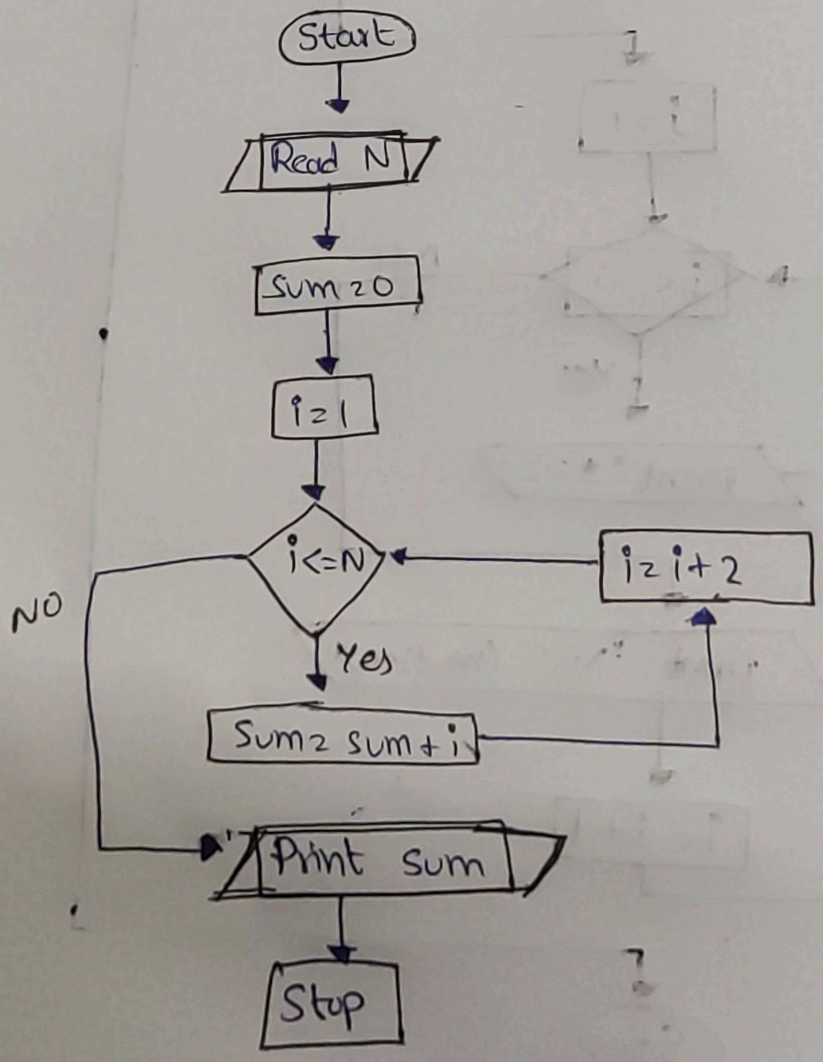

Start
Input n (no. of rows)
i = 1
i <= n
Yes
m = 1
m <= n-i
NO
yes
m = m+1
print " "
j = 1
j <= i
NO
Yes
j = j+1
print "* "
print "\n" (newline)
i = i+1
Stop

16.)

(a) Algorithm

1. Start
2. Read N (odd positive integer)
3.) Initialize sum=0
4.) For i=1, i<=N, i=i+2
5.) Check for condition, if it satisfies go to next step, or else stop
6.) Sum = sum + i
7.) Go to step 4
8.) Display sum
9.) Stop

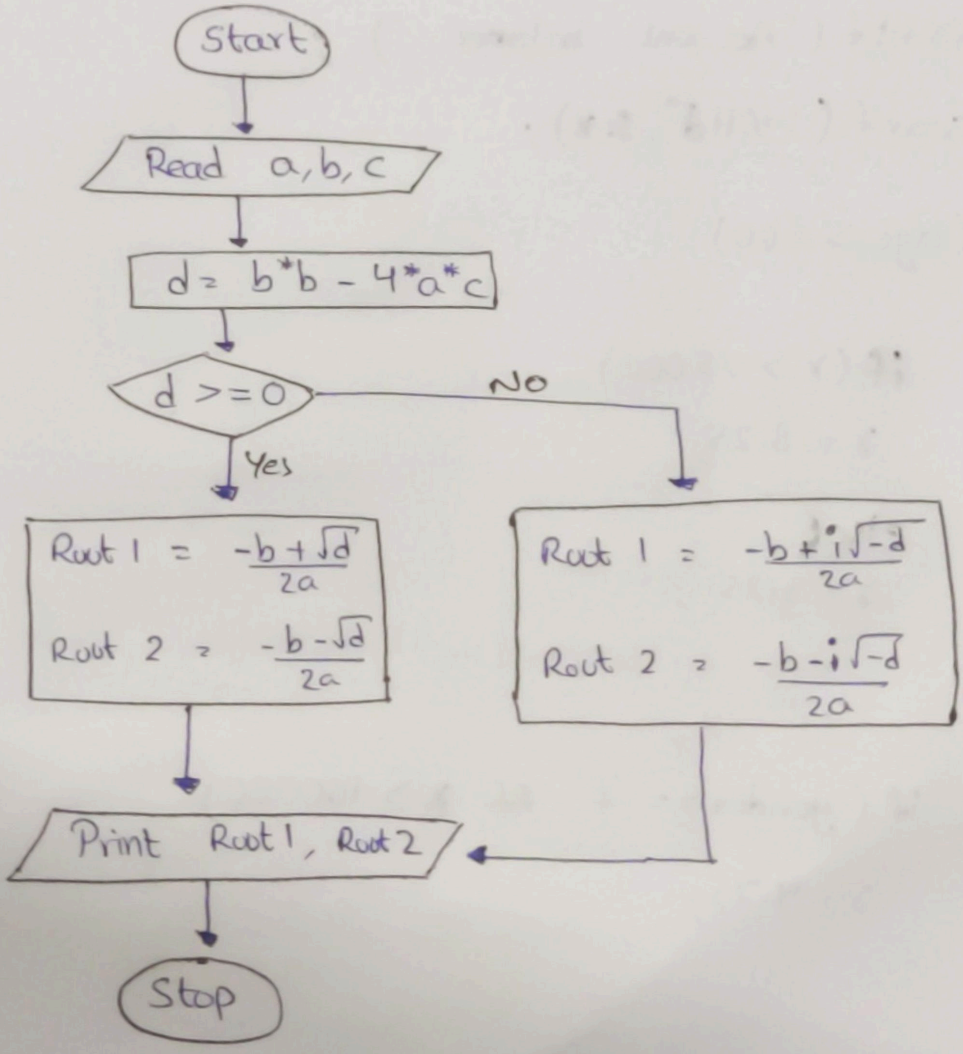
Flowchart



b.) Algorithm

1. Start
2. Read coefficients a, b, c
3. Calculate $d = b^2 - 4ac$
4) If $d < 0$ display "roots are imaginary".
5. Root 1 = $\frac{-b + i\sqrt{-d}}{2a}$ Root 2 = $\frac{-b - i\sqrt{-d}}{2a}$ where $i = \sqrt{-1}$
   end if
6. If $d = 0$ display "roots are real and equal"
7. If $d > 0$ display "roots are read and unequal"
8.) Root 1 = $\frac{-b + \sqrt{d}}{2a}$ Root 2 = $\frac{-b - \sqrt{d}}{2a}$
   end if
9. Display roots
10. Stop

Flowchart



17.)

```
# include < stdio.h>
int main ( )
{
    long long int   x, age ;
    double   r;
    char   gender;
// x is the account balance and r is rate of interest
    printf ("Gender (m or f): ");
    scanf ("%c", &gender);
    printf ("Enter your age: ");
    scanf ("%lld", &age) ;
    printf ("Account balance: ") ;
    scanf ("%lld", &x);

if (age >=60)
{
    if (x > 75000)
     r = 8.25;
    else
     r = 3.25 ;
}

else if ( gender == 'f' && x > 100000)
     r = 4.75;
```

```
else if (x < 5000)
    r = -1.25;
printf("Rate of interest = %.1f \n", r);
printf("Interest = %.1f \n", x*r/100);
printf("Amount = %.11d", x += x*r/100);
// Here -ve interest is the penalty.
return 0;
}
```

18.)

```
#include <stdio.h>
int main()
{
int n, zero=0, one=0, two=0, three=0, four=0, five=0,
    six=0, seven=0, eight=0, nine=0;

while(1)
{
    printf("Enter the number: ");
    scanf("%d", &n);
// only non-negative integers are considered valid inputs for n.
    if (n==-1) break;
    if (n==0) zero++;
while(n)
    {  if(n%10==0) zero++;
       else if(n%10==1) one++;
       else if(n%10==2) two++;
       else if(n%10==3) three++;
       else if(n%10==4) four++;
       else if(n%10==5) five++;
       else if(n%10==6) six++;
       else if(n%10==7) seven++;
       else if(n%10==8) eight++;
       else if(n%10==9) nine++;
n=n/10; }   // end inner while
```

```
    } // end outer while
    printf (" 0: %d,  1: %d , 2: %d, 3: %d , 4: %d , 5: %d , 6: %d,
            7: %d,  8: %d,  9: %d", zero, one, two, three,
            four, five ; six, seven, eight, nine) ;
    return 0;
}
```