① Explain the evolution of programing languages.

→

1950's	Creation of high-level languages
1960s	forth, Simula I, lisp, Cobol
1970s	pascal, C language
1980s	ML Smalltalk, c++
1990s	Pearl, Python, Java
2000s	Internet programing
2010	Concurrency and asynchronicity. Javascript and go language

Programing has its origin in the 19th century where the first "programmable" looms and player piano scrolls were developed. In the 1960s, assembly level programing had evolved and programing languages such as Forth, Simula I and Cobol came into existance. FORTAN was the first comprehensive high level programing language that was widely used, C language was designed in 1970s in Bell laboratories by Dennis Ritchie. It also has a sucessor c++, With the concept of object oriented programing languages like Java, peal and python came into existance.

With the introduction of internet and its level of influence on the comman people, the early 21st century saw the development of internal programing and mainly javascript

2) Explain the various steps involved in problem solving with diagram.

→ The various steps involved in problem solving are as below

- Defining the program :- Understanding the situation, making note of the requirements and the output. Knowing the exact input and output comes under the process of defining the program.
- Planing the code :- Planing the code means creating algorithms, flowcharts and pseducode.
- Coding the program :- It actually is creating writting the actual code for the program with the help of algorithm, flowchart and pseudocode made earlier.
- Testing the program :- Checks the efficiency of the program by providing it many test cases to check the reliability of the program.
- Refining the program :- Making the program realistic, appealing and more reliable according to the requirements and upgrading it.
- Documenting the program :-

(3) Describe the basic structure of C program with an example.

→ Structure of a C program is based on set of rules defined by the compiler; it has 6 sections

(i) Documentation Section:

Used for providing comments which are not executable and ignored at the time of execution. The comments are treated as single white space by compiler.

(ii) Preprocessor Section:

Also known as Preprocessor directive or header files. They are not a part of compiler but they instruct the compiler to do required preprocessing. Begin with # symbol and preprocessor is written within <>

(iii) Global Declaration Section:

Used to declare global or public variables. They are declared outside all functions. Variables declared here are accessable to all functions in the code even multiple times.

(iv) main() section

Execution starts with a opening brace({) and ends with a closing brace(}). It is basically divided into two sections.

- Declaration
- Executable

(v) Local Declaration Section:

Local variables initialized with basic data types and declared within the main() program, ~~also called~~

(vi) Execution Section:

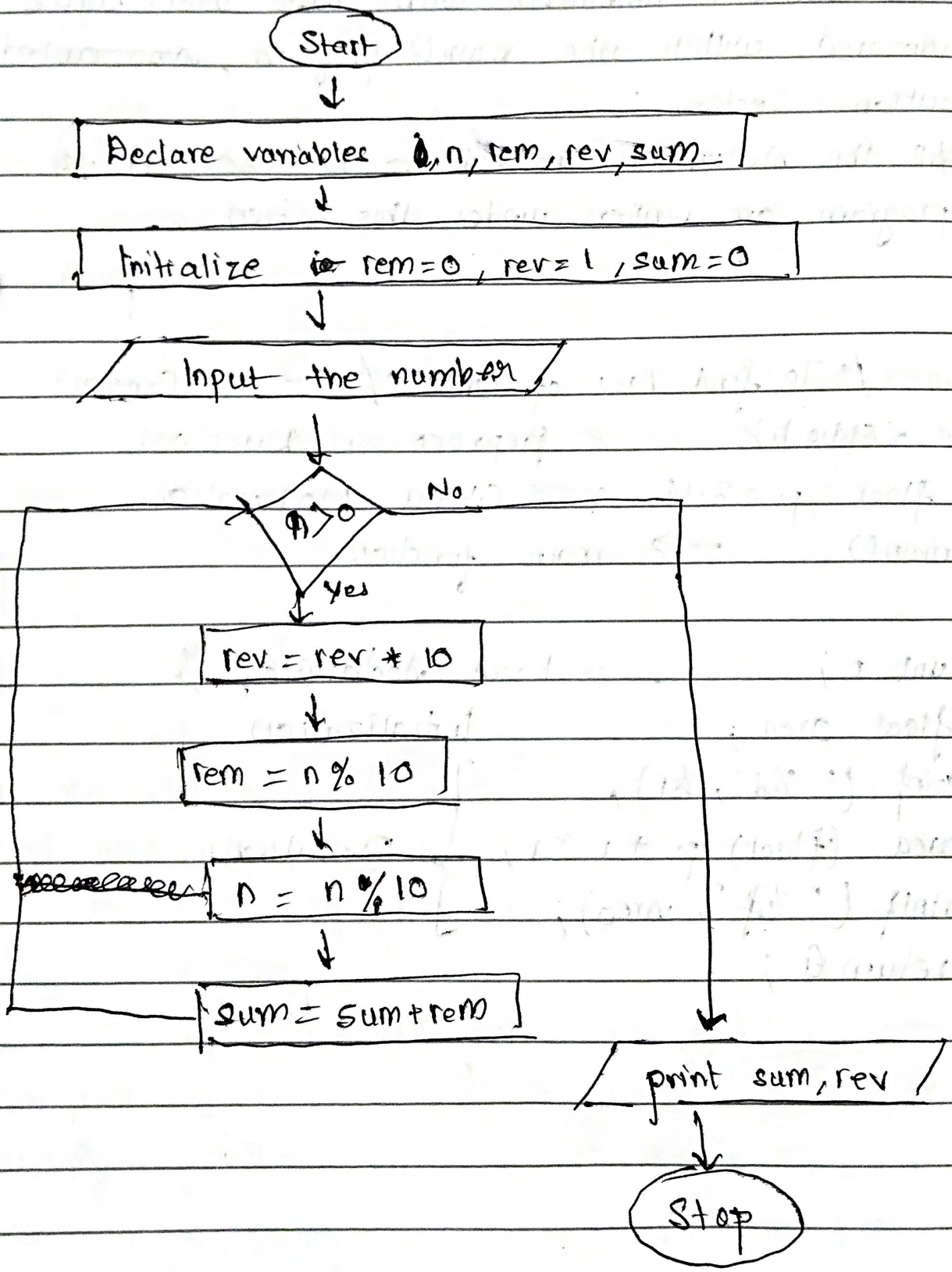
All the statements needed to be executed in the program are written under this section.

Example:

```
/* To find Area of Circle */ ---> Comment
#include <stdio.h> ---> Preprocessor Directives
const float pi = 3.14 ---> Global Declaration
int main() ---> main function
{
    int r; ---> Local declaration & Initialization
    float area;
    scanf ("%d", &r);
    area = (float) pi * r * r;
    printf ("%f", area);
    return 0;
}
```

Execution

(4) Draw the flowchart and write the algorithm and c code to find the sum and to reverse the digits of given 5 digit number.



Algorithm

- Start
- Declare variables n, rem, rev, sum
- Initialise them as sum = 0 rem = 0 rev = 1
- Take the number as input
- Use loop for following steps
  - → rev = rev * 10
  - → rem = n % 10
  - → n = n / 10
  - → sum = sum + rem
  - → Check if n > 0
- Print rev, sum
- End

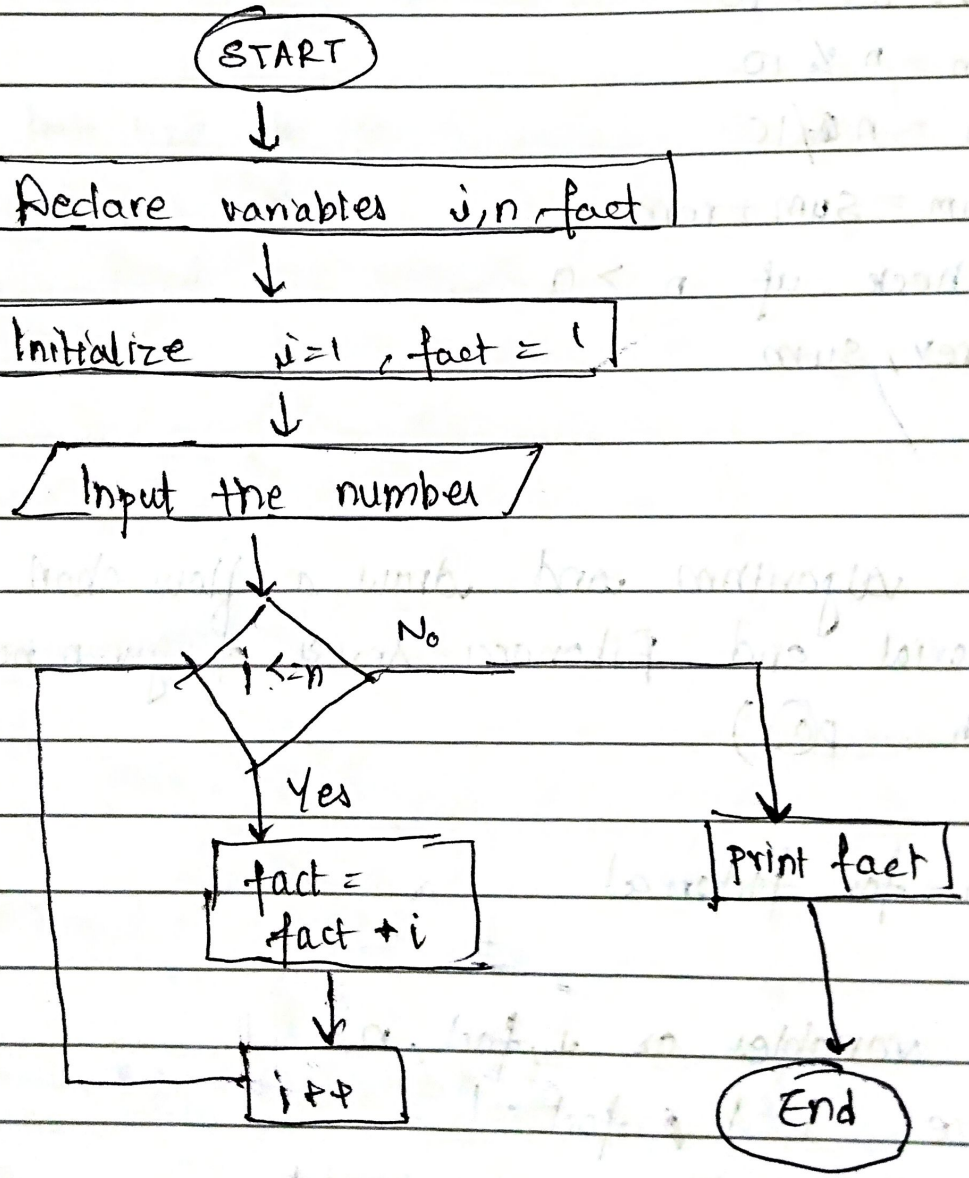
(5) Write an algorithm and draw a flow chart to find factorial and Fibonacci series of given number

→ (for Fib .... pg 4)

→ Algorithm for factorial

- Start
- Declare variables as i, fact, n;
- Initialize i = 1 , fact = 1
- Take the number n as input

- Use loop for following steps
  - → fact = fact * i
  - → increase i by 1
  - → Check whether i is less then or equal to n
- print fact
- End



(6) Analyse the varous storage classes in C with suitable examples

→ Storage Class Specifiers tells the complier about the place to store a variable, its lifetime and initial value of the variable.

According to storage classes and variable scope, the variables are categorized as

(i) Automatic variables

It is the default storage class, defined inside a function. The scope of the variable is local to the function block where variable is defined. It's content vanishes after execution.

(ii) Global Variable

It is available to all functions, defined outside the function, Keyword Declare is used to define these variables. It's content vanishes after the entire program is executed.

(iii) Static variable

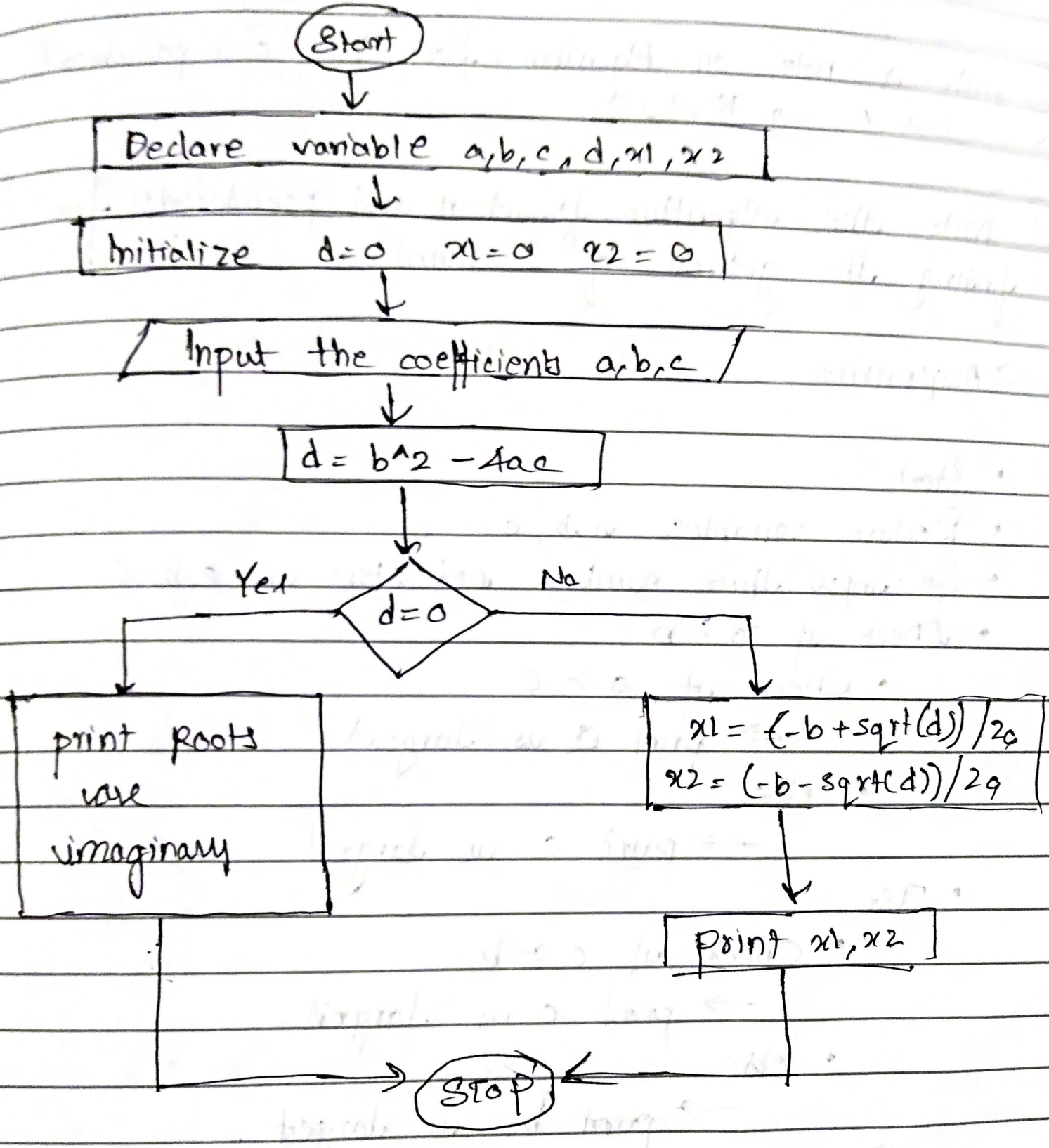
Keyword STATIC is used to define the variable. Variable declared as static variable remains the same throughout the program. It's scope depends on where it is declared

iv) Register Variables : Registers are special storage areas within a computer's central processing unit. Register variables tell the compiler that the variable is kept in the CPU, since register access is faster than memory access

(7) Sketch a flowchart for roots of a quadratic equation and write an algorithm for it.

→ ALGORITHM

- Start
- Declare variables a, b, c, d, x1, x2
- Initialize d=0, x1=0 x2=0
- Calculate d = b^2 - 4ac
- Get the coefficients a, b, c
- If d<0
  → Display "Roots are imaginary"
- else
  x1 = (-b + sqrt(d))/2a
  x2 = (-b - sqrt(d))/2a
- End


Start
Declare variable a,b,c,d,x1,x2
Initialize d=0 x1=0 x2=0
Input the coefficients a,b,c
d = b^2 -4ac
d=0
Yes
No
print roots are imaginary
x1 = (-b+sqrt(d))/2a
x2 = (-b-sqrt(d))/2a
Print x1,x2
Stop

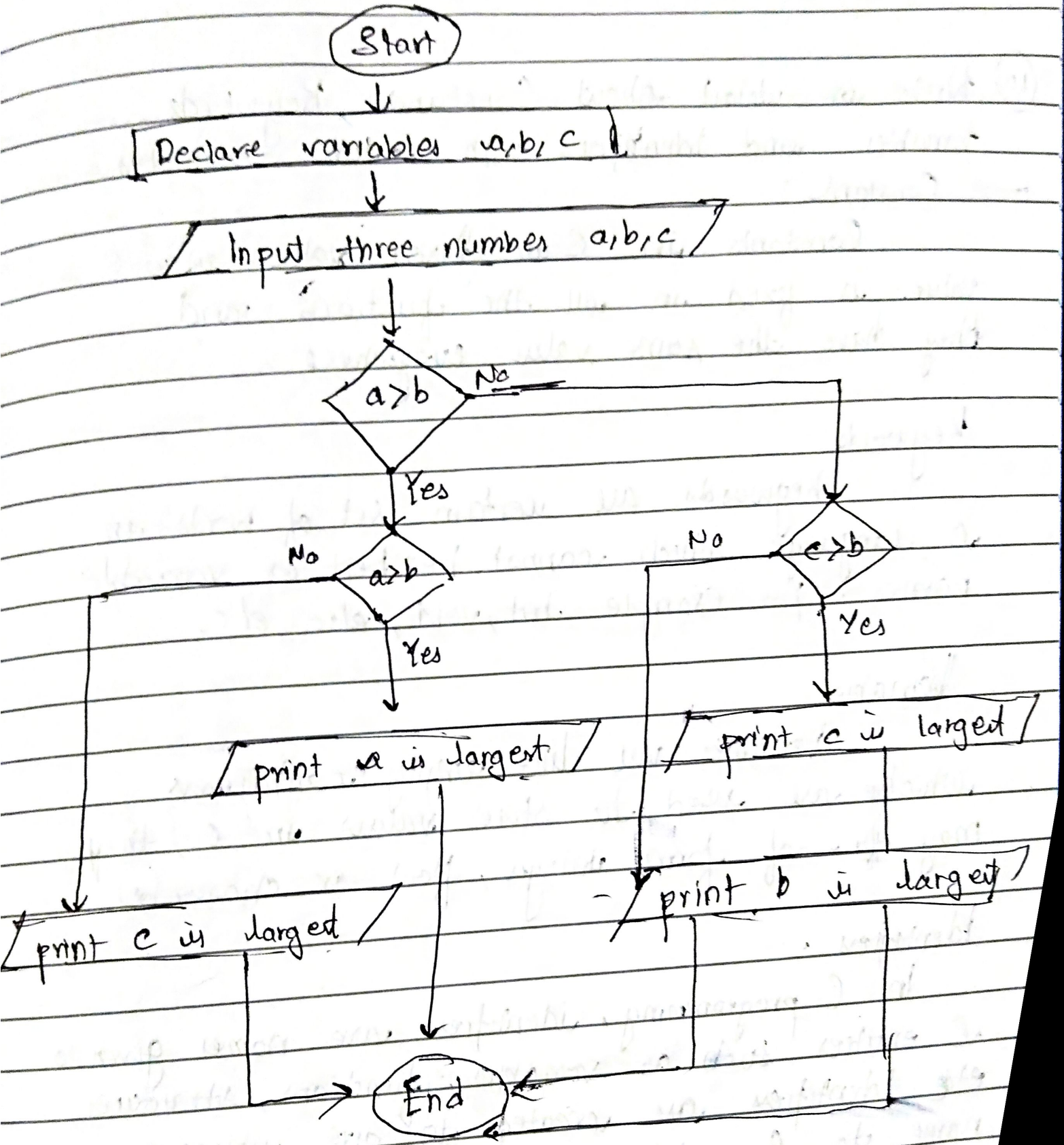
(8) Write a note on Algorithm, flow chart and pseudocode

→ Part A q 10, 11, 12

(9) Write the algorithm, flowchart and pseudocode for finding the greatest of 3 numbers.

→ ALGORITHM

- Start
- Declare variables a, b, c
- Get input three numbers and store in a, b, c
- Check if a > b
  - Check if a > c
    - → print a is largest
  - else
    - → print c is largest
- Else
  - Check if c > b
    - → print c is largest
  - else
    - → print b is largest.
- End


Start
Declare variables a, b, c
Input three numbers a, b, c
a>b
No
Yes
a>b
No
Yes
print a is largest
print c is largest
c>b
No
Yes
print c is largest
print b is largest
End

(10) Write in detail about Constants, Keywords, Variables and Identifiers with suitable examples.

→ Constants :

Constants in C are fixed values whose value is fixed in all the functions and they have the same value everywhere.

Keywords :

Keywords are certain set of words in C language which cannot be kept as variable names : for example int, void, else etc.

Variables

Variables are like empty containers which are used to store values in C ; they may be of form integer, float or character.

Identifiers :

In C programming, identifiers are names given to C entities such as variables, functions, structures etc. Identifiers are created to give unique name to C entities to identify it during the execution of program.

(12) Write BITWISE AND and OR

→
```
#include <stdio.h>
int main()
{
    int x, a, o, y;
    scanf("%d%d", &x, &y);
    a = x && y;
    o = x || y;
    printf(" Bitwise AND : %d
           Bitwise OR : %d", a, o);
    return 0;
}
```

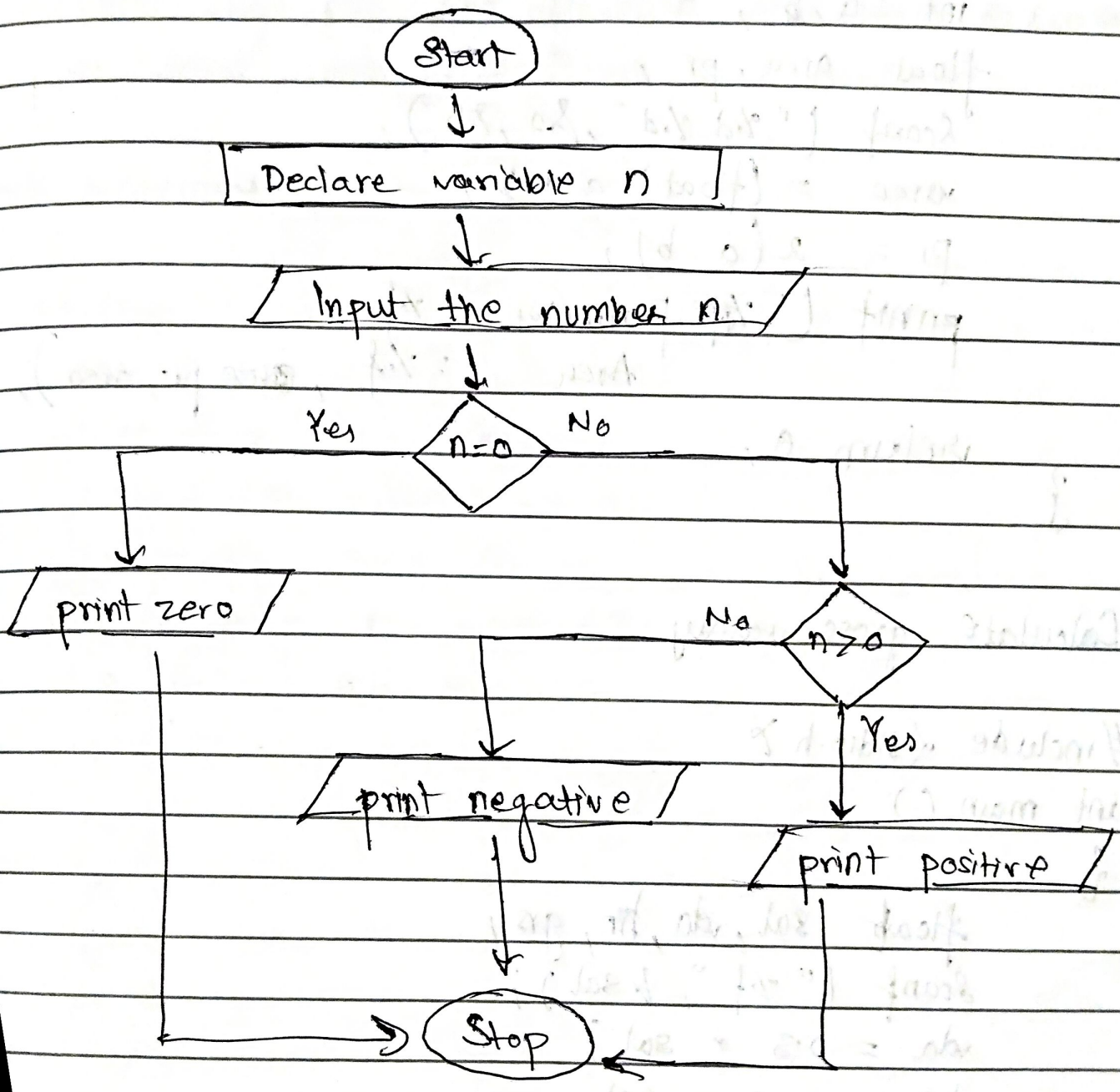
(13) C program to calculate sum of digits of a number

→
```
#include <stdio.h>
int main()
{
    int n, sum = 0;
    scanf("%d", &n);
    while (n > 0)
    {
        sum = sum + (n % 10);
        n = n/10;
    }
```

```
    printf("Sum : %d", sum);
    return 0;
}
```

(14) Symbols for flowchart

→ q 12 – A



(15) C program to find area and perimeter

→

```
#include <stdio.h>
int main()
{
    int a,b;
    float area, pr;
    scanf("%d %d", &a, &b);
    area = (float) a * b;
    pr = 2(a+b);
    printf("Perimeter : %f
            Area : %f", pr, area);

    return 0;
}
```

(16) Calculate gross salary

→

```
#include <stdio.h>
int main()
{
    float sal, da, hr, grs;
    scanf("%f", &sal);
    da = 0.3 * sal;
    hr = 0.25 * sal;
```

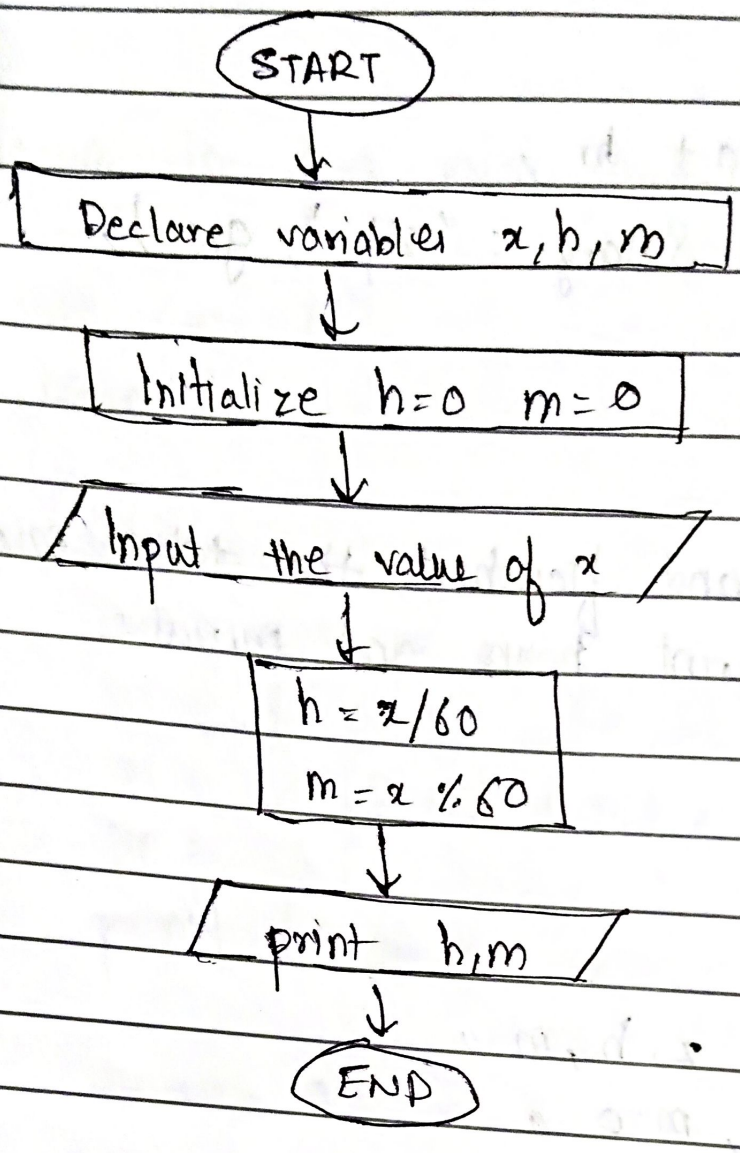
```
    grs = sal + da + hr ;
    printf (" Gross Salary : "%f ", grs );
    return 0 ;
}
```

(17) Write algo, psd. and flowchart that take minutes as input and print hours and minutes

→ ALGORITHM

- Strat
- Declare variables x, h, m ;
- Initialize h=0 , m=0 ;
- Input the number as x
- Calculate h = x / 60 ;
- Calculate m = x % 60 ;
- print h and m
- End

FLOWCHART



PSEUDOCODE..

1. READ variabe x
2. h = x/60
3. m = x % 60
4. WRITE h, m

(18) C program to find vagg mark & percentage

→
```
#include <stdio.h>
int main ()
{
    int a,b,c,d,e,f, sum=0 ;
    float avg ;
    Scanf (" %d %d %d %d %d %d ", &a,&b,&c, &d,&e &f);
    Sum = a+b+c+d+e+f ;
    avg = (float) sum/6 ,
    printf (" Aggregate Mark : %d
                Percentage  : %f , sum, avg );
    return 0 ;
}
```

(19) Explain the scope, lifetime of variables in C with examples

→ q 3.8

(20) C program to convert the temperature from farenheit to celcius

→
```
#include <stdio.h>
int main()
{
    int f, c;
    scanf("%d", &f);
    c = (5/9) * (f + 32);
    printf("%d", c);
    return 0;
}
```

(21) Explain bitwise operators in C with suitable examples.

→ In the Arithmetic - logic unit of the system, mathematical operations like addition, substraction, multiplication and division are done in bit-level. To perform bit-level operations in C programming, bitwise operators are used.

Bitwise operators in C language are

&	bitwise AND
|	bitwise OR
~	bitwise NOT
^	XOR
<<	left shift
>>	right shift

Truth table

a	b	a&b	a|b	a^b	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Eg A << 2 ; the operands value is moved left by 2 bits.

A >> 2 ; the operands value is moved right by 2 bits.

(22) Write algorithm to find largest among three numbers

- Start
- Declare variables a, b, c
- Get input 3 numbers and store in a, b, c
- Check if a > b
  - Check if a > c
    → Print a is largest
  - else
    → Print c is largest
- Else
  - Check if b > c
    → Print b is largest
  - else
    → print c is largest
- End

(25) Explain in details about operators with an example

→ Operator is defined as a symbol (or) special character that instruct the compiler to perform certain mathematical (or) logical operations.

- Increament & Decrement Operators
  Unary operators that add or subtract one from the operand. Increament is written as ++ and decremant as --
  They are of 2 kind -
  → Prefix increament / decrement (++i / --i)
  → Postfix increment / decrement (i++ / i--)

- Comma Operator:
  Special operator which seperate the declaration of multiple variables. It has lowest precendence and is evaluated at last. It returns the value of rightmost operand when multiple are used

- Arrow Operator (→)

It is used to access the structure members when we use pointer variable to access it. When pointer to a structure is used then arrow operator is used.

- Assignment Operators.

It assigns the result of the expression or the ~~variab~~ value to a variable.

variable (=) expression

↓

Assignment operator

Some shorthand operators are

	Short hand Assignment operators
a = a+1	a+=1
a = a-1	a-=1
a = a * 1	a*=2
a = a / b	a/=b
a = a % b	a %=b
c = c*(a+b)	c*=(a+b)
b = b/(a+b)	b/=(a+b)

- Bitwise operators
  (q 21 B)

- Sizeof Operators
  This operator returns the size of its operand in bytes. Used to calculate the size of data type or variables and returns the size in integer format.
  It's syntax looks more like a function but it is considered as an operator in C programing

```
int i;
sizeof(i) = 4
```