SARDAR VALLABHBHAI PATEL INSTITUTE OF TECHNOLOGY, VASAD -388306

(ACCREDITATED BY NATIONAL BOARD OF ACCREDITATION, AICTE)

PROGRAMMING FOR PROBLEM SOLVING

SUBJECT CODE:- 3110003

F.Y.B.E (SEM-I / II)

ASSIGNMENT BOOK

NAME:- Parmar Nisarg S.

BRANCH:- Mechanical SEMESTER:- 1st

ENROLMENT NO:- 190410119055 BATCH:- B

I.D.NO:- 19BEMEF045 YEAR:- 2019 - 2020

Sardar Vallabhbhai Patel Institute of Technology, Vasad
B.E. Semester – 1
Subject: Programming for Problem Solving (3110003)
Assignment – 1 (Unit – 1)

1. Discuss basic component of computer system with the help of block diagram.
2. Differentiate between system software and application software.
3. How many levels of programming languages are available for computing task within computer?
4. What is language translator? Explain different type of translator.
5. What is the need to draw flowchart/algorithm?
6. Write down a flowchart to identify greatest number within three numbers.
7. Write the process to compile and running a program.
8. Define tokens which are used in C programming.
9. Define data types and identifier using c program.

Assignment 1 - (Unit 1)

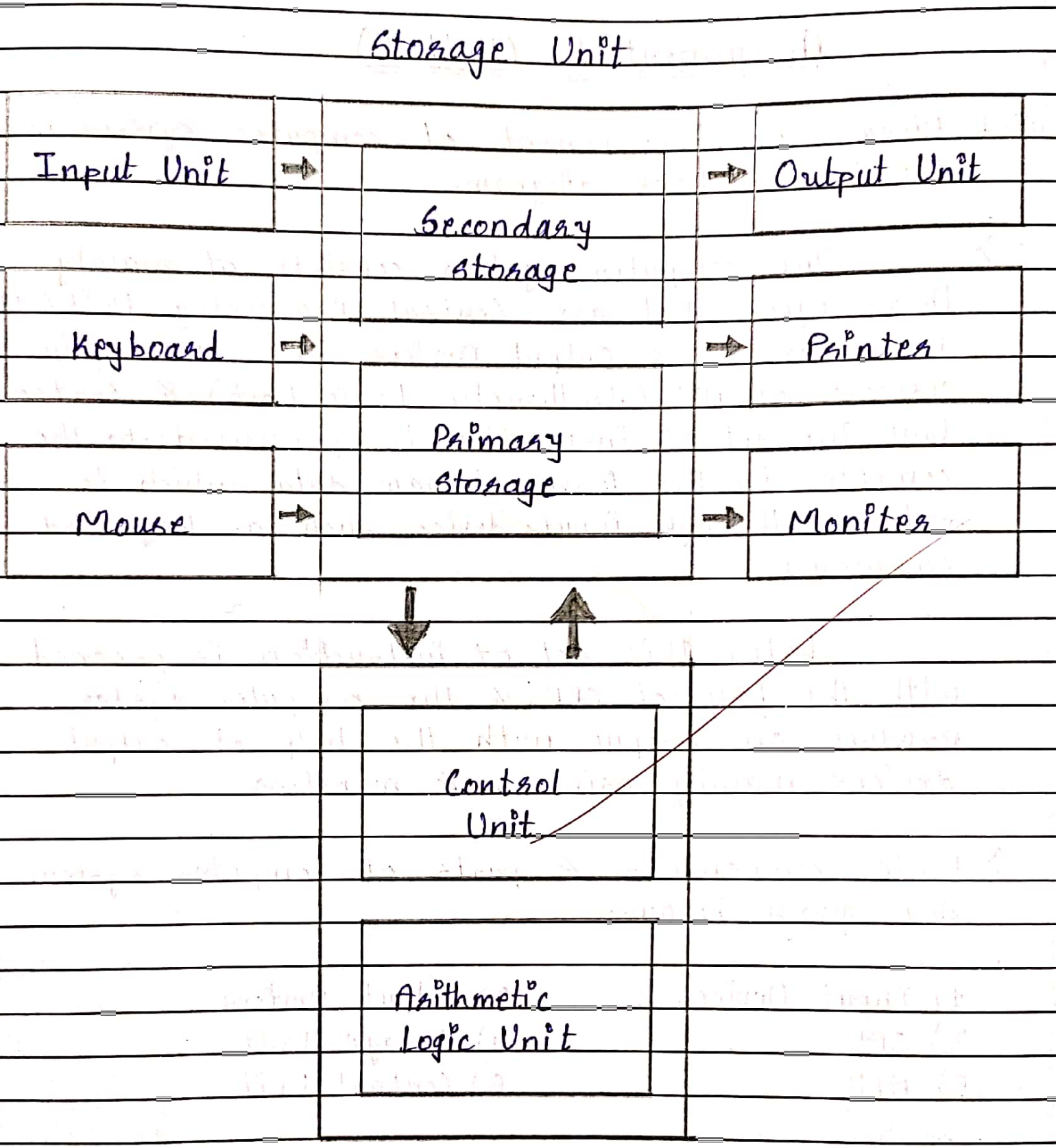
Q.1 Discuss basic component of computer system with the help of block diagram.

⇒ The computer system consists of mainly three types that are Central Processing Unit (CPU), Input Devices, & Output Devices, The CPU again consist of ALU (Arithmetic Logic Unit) & Control Unit. The set of instruction is presented to the computer in the form of raw data which is entered through input device such as keyboard or mouse.

Later this set of instruction is proceed with the help of CPU, & the computer system produce an output with the help of output devices mainly printers & monitors.

⇒ Basic components & parts of computer system are given below.

1) Input Devices
2) Output Devices
3) CPU
4) Storage Unit
5) ALU
6) Control Unit



Block Diagram of Computer's Components.

Q. 2. Differentiate between system software & application software.

	System software	Application software
1.	System software is used for operating computer hardware.	Application software is used by user to perform specific task.
2.	It is installed on the computer when operating system is installed.	It is installed according to user requirement.
3.	In general, the user doesn't interact with system software because it works in the background.	In general, the user interacts with application software.
4.	System software can run independently. It provides platform for running app. software.	It can't run independetly. They can't run without the presence of system software.
5.	e.g. compiler, assembler, debugger, driver, etc.	e.g. word processor, web browser, etc.

Q. 3. How many levels of programming language are available for computing task within compute?

Programming languages can be broadly classified into three categories.
1) Machine languages 2) Assembly languages
3) High level languages.

1) Machine language :- It is native language of computer, the language closests to the hardware itself. Each unique computer has a unique machine language. A machine language program is made up of a series of binary patterns which represents simple operations that can be accomplished by the computer. (e.g. add two numbers, move data to a memory location.) Machine language can be run directly. Programming in machine language requires memorization of the binary codes & can be difficult for human programmers.

2) Assembly language :- They represent an effort to make programming easier for the human. The machine language instructions are replaced with simple pneumonic abbreviations. (e.g. ADD, MOV.) Thus assembly languages are unique to a specific computer. An assembly languages program requires translation to machin

language. This translation is accomplish by a computer are written for each unique machine language.

3) High level language :- High level languages, like C, C++, JAVA, etc. are more English like & therefore, make it easier for programmers to "think" in the programming language. High level language also require translation to machine language before execution. This translation is accomplished by either a compiler or an interpreter.

- Compiler translate the entire source code program before execution (e.g. C++, JAVA)
- Interpreter translate source code programs one line at a time. (e.g. python)

Q.4. Explain what is language translator? Explain different type of translator?

A program written in a high level language is called as source code. To convert the source code into machine code, translators are needed.

Role of translator are....

1) Translating the high level language program into an equivalent machine language program.

2) Providing diagonastic messages wherever the programmer violets specification of the high-level language program.

The different type of translator are...

1) Compiler used to convert high-level language to low-level language. It translates the entire program & also reports the errors in source program encountered during the translator.

2) Interpreter is also used to convert high-level language to low-level language. It translates line by line & reports the error once it encountered during the translation process.

It directly executes the operation specified in the source program when the input is given by user. It gives better error diagnotics than a compiler.

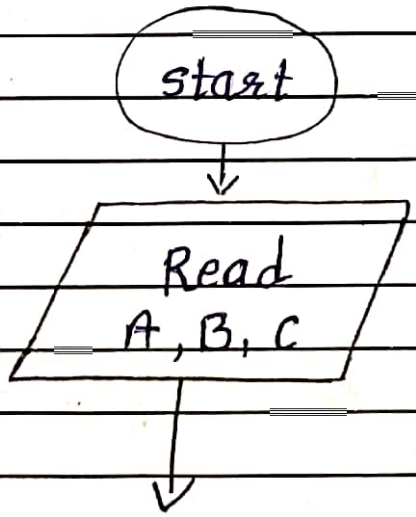
Q.5 What is the need to draw flowchart / algorithm.

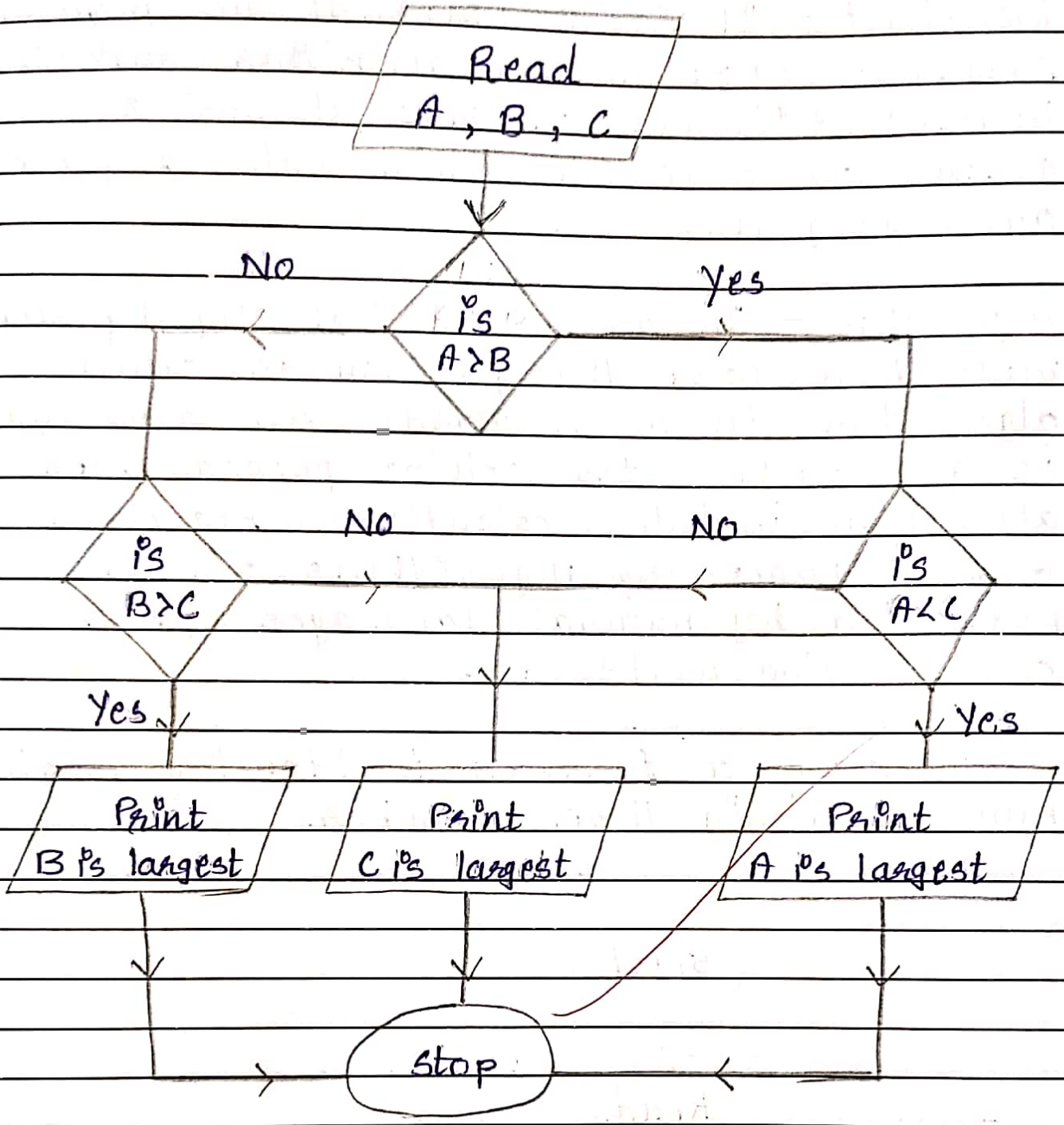
An algorithm is a step-by-step analysis of the process while a flowchart explains the steps of a program in a graphical way.

Flowchart - It is a graphical or pictorial representation of an algorithm with the help of different symbols, shapes & arrows in order to demonstrate a process or a program.

Algorithm - To write a logical step-by-step method to solve the problem is called algorithm, in other words, an algorithm is a procedure for solving programs. An algorithm includes calculations, reasoning & data processing. Algorithms can be presented by natural languages, pseudo code & flowcharts, etc.

Q. 6. Write down a flowchart to identify greatest number within three number.





Q. 7. Write the process to compile & running a program.

Let us assume that the source program

has been created in a file named program1.c
Now, the program is ready for compilation.
The compilation command to achive this task under lubuntu is

gcc. filename.c

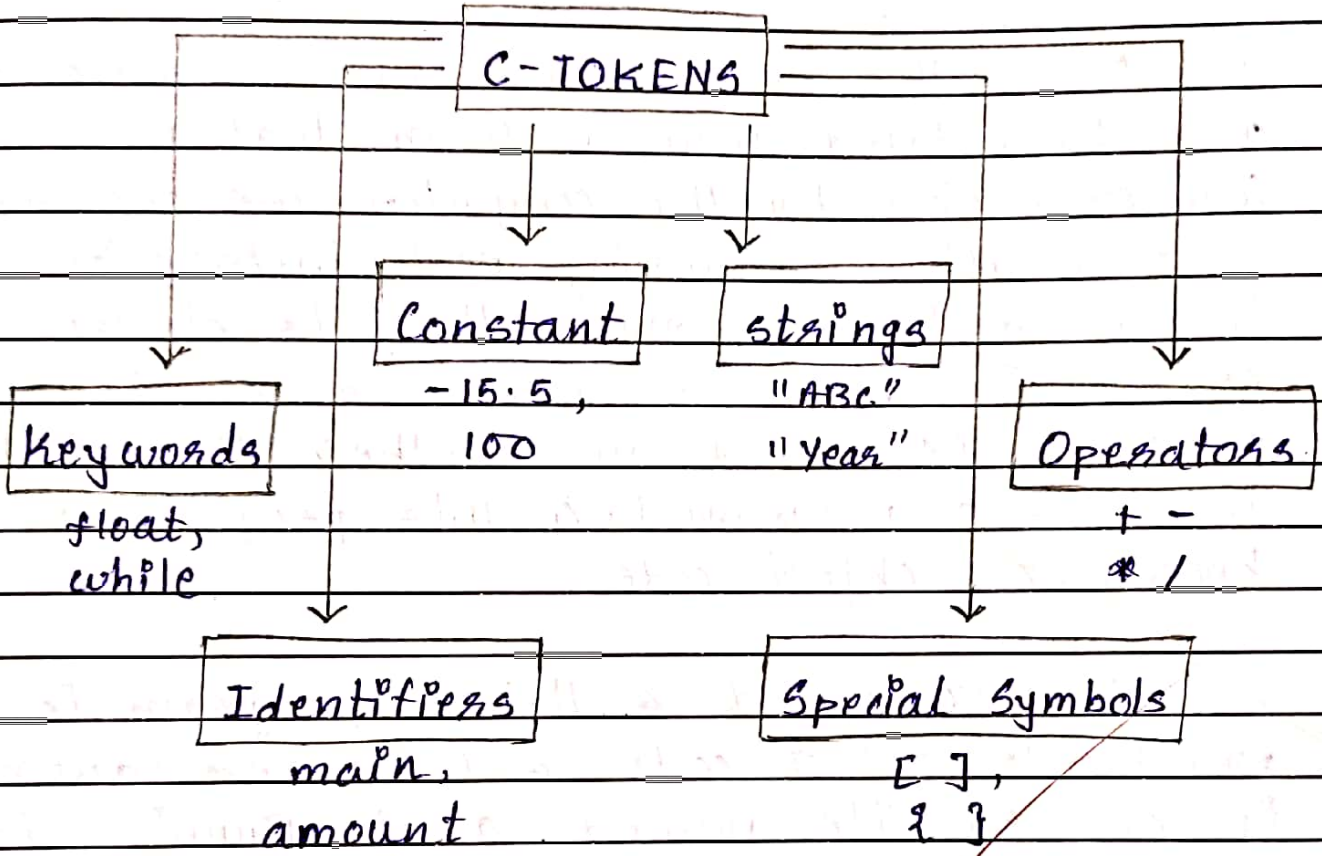
Now, the source program instructions are translated into a form that is suitable for execution by the computer. The translation is done after examining each instruction for its corrections. If every thing is alright, the compilation proceeds silently & the translated program is stored on another file with the name program1.o. This program is known as object code.

The compiled & linked program is called executable object code & is stored automatically in another file named a.out. Running is a simple task, The command is ./a.out

would load the executable object code into the computer memory & execute the instructions. During all this process if there were no errors then program will be executed successfully.

Q.8. Define tokens which are used in C-programming.

Each & every smallest individul units in C-programming are known as C-tokens C-tokens are the basic building blocks in C-language which are constructed together.



Q.9. Define data types & indentifiers using C-program.

C-program provides various data types to make it easly for a programmer to select a suitable data type as per the requirement of an applications.

Following are three data types :-
1) Primitive data type
2) Derived data type
3) User-defined data type

There are five primary fundamental data types :-
1) int for integer data
2) char for character
3) float for floating points number
4) double for double precision floating point
5) void

It is a string of alphanumeric characters that begins with an alphabetic characters or an underscore character that are used to represent various programming elements such as variable, functions, arrays, structure unions & so on. An identifier is a user defined word.

Sardar Vallabhbhai Patel Institute of Technology, Vasad
B.E. Semester – 1
Subject: Programming for Problem Solving (3110003)
Assignment – 2 (Unit – 2)

1. Write down basic structure of C program.
2. Draw a flow chart and write algorithm for the following:
   a. To find the maximum number out of three given numbers.
   b. To perform the summation of 10 elements read from the user.
3. Explain logical and relational operators used in C. Explain using proper examples.
4. Explain the ternary operator with appropriate example.
5. Explain C tokens? Briefly explain each token with C code.
6. What is the need of initialization of a variable and assignment statement used in C program?
7. How can you read the data from keyboard and display it on the screen?
8. Explain the bit-wise operators.
9. What is use of comma ',' operator?

Assignment - 2 (Unit 2)

Q.1. Write down basic structure of C-program.

C-program is divided into 7 parts. This 7 parts are called basic structure of C-program.

1) Document section.
2) Preprocessor / Link section.
3) Definition section.
4) Global declaration section.
5) Function declaration section.
6) Main function.
7) User-defined function section.

* Program to understand basic structure of C.

```
// Name of program ------------> Document section

#include <stdio.h>   }
#include <conio.h>   } ------------> Preprocessor Directives

#defined max 100 ------------> Definition section

Void add();  } ------------> Global declaration section
int x=100;   }

main()
{
}

add()
{
}
```

```
int main ()  ——————> Function section
{
  int a=100;  ——————> Variable declaration
  printf("Hello");}  ——————> Body of main
  return 0;  }          function
}
  Void add ()
{
  printf("Hello");  ——————> Function Definition
}
```

Q. 2. Draw a flowchart & write algorithm for the following.

(a) To find maximum number out of 3 given number.

Algorithm :-

1> Start
2> Read 3 number : num 1, 2, 3
3> if num 1 > num 2 then go to step 5.
4> if num 2 > num 3 then,
        print num 2 is largest.
    else
        print num 3 is largest.
5> if num 1 > num 3 then,

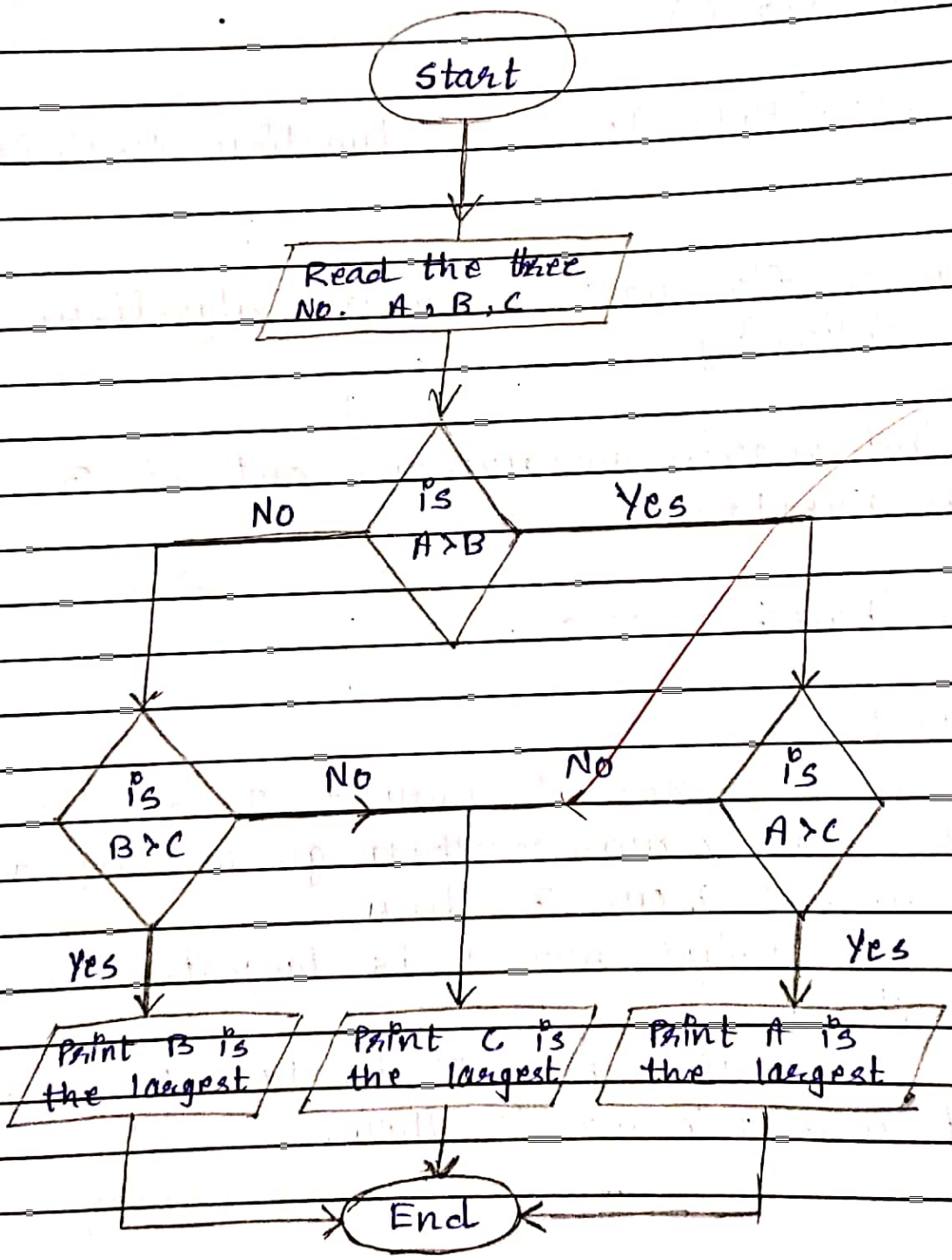
print num 1 is largest.

else

print num 2 is largest.

6) End.

Flowchart :-

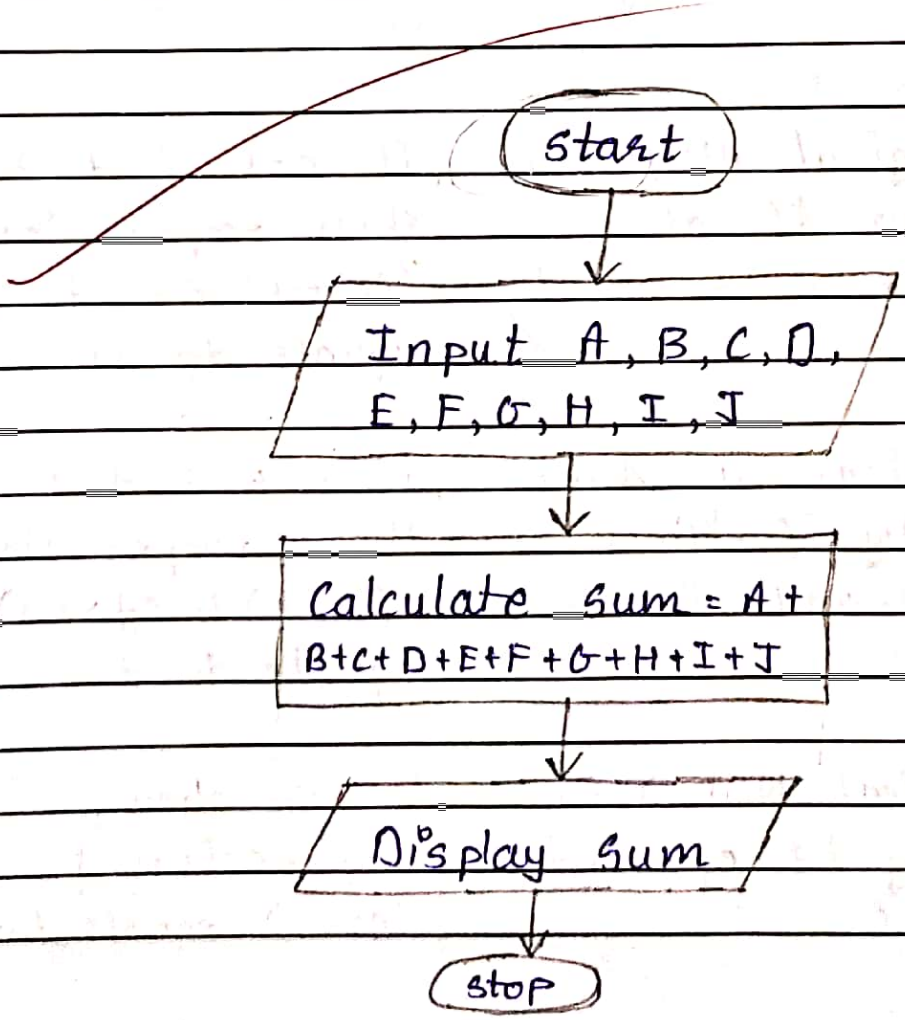


(b) To perform the summation of 10 elements read from the user.

Algorithm :-

1) Start
2) Accept 10 element from user A, B, C, D, E, F, G, H, I, J
3) Add 10 elements.
4) Display sum of 10 element.
5) stop.

Flowchart :-



Q.3. Explain logical & relational operators used in C. Explain using proper example.

Logical Operators :-

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making.

Operator	Meaning	Example
&&	Logical AND. True only if all operands are true.	if c=5 & d=2 then expression ((c==5) && (d>5)) equals to 0.
||	Logical OR. True only if either one operands is true.	if c=5 & d=2 then expression ((c==5) || (d>5)) equals to 1.
!	Logical NOT. True only if operand is 0.	if c=5 then, expression !(c==5) equals to 0.

Relational Operators :-

It checks the relationship b/w two operands. It relation is true, it returns 1 ; if false , returns 0.

Operator	Meaning	Example
==	Equal to	5==3 is evaluted to 0.
>	Greater than	5>3 is evaluted to 1.
<	Less than	5<3 is evaluted to 0.
!=	Not equal to	5!=3 is evaluted to 1.
>=	Greater than or equal to	5>=3 is evaluted to 1.
<=	Less tan or equal to	5<=3 is evaluted to 0.

Q. 4. Explain ternary operator with appropriate example.

Ternary operator is used to execute code based on the result of a binary condition.

It takes in a binary condition as input, which makes it similar to an 'if-else' control flow block. It also, however, returns a value, behaving similar to a function.

Syntax of Ternary Operators :-

condition ? Value if true : Value if false.

The statement evaluates to statement 1 if the condition is true & statement 2 otherwise.

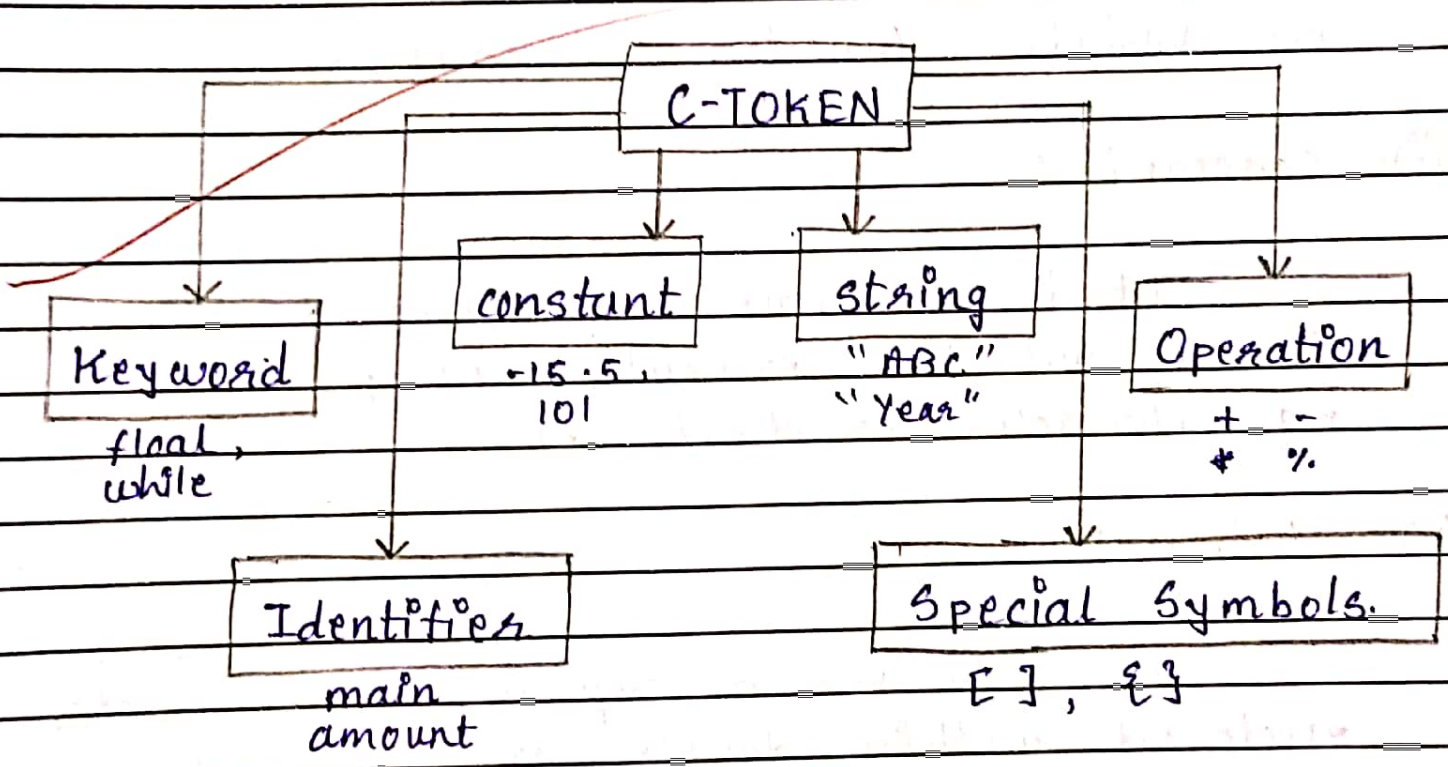
example :-

```
int a=10, b=20, c;
c = (a<b) ? a, b;
printf ("%d", c);
```

output :- 10

Q.5. Explain C token ? Briefly explain each token.

In C program, each word & punctuation is refferred to as token. C token are the smallest builind block or smallest unit of a C programm. The compile breaks a program into smallest possible units & proceeds to various stages of compilation, which is called token.



(i) Keywords :-

Keywords are pre-defined words.

In a C- compiler each keyword is meant to perform a specific function in a C program it can't be used as variable name.
e.g. float, const, int, etc.

(ii) Indentifier :-

Each program element in a C program are given a name caued indentifier.
e.g. variable, function, arrays, etc.

(iii) Constant :-

It is a value that doesn't change throughout the execution of program.
e.g. character constant, etc.

(iv) string :-

It contains a sequence of character enclosed within double quotes (" ").
e.g. "ABC", "Year", etc.

(v) Operators :-

it before it's used. It a variable that isn't initialized doesn't have a defined value.

Q. 7. Explain Bit-wise operators.

A bitwise operator is an operator used to perform bitwise operations on bit patterns or binary numerals that involves the manipulation of individual bits.

Decimal values are converted into binary values. which are the sequence of bits & bit wise operators work on these bits.

Bitwise operator in C-language are...

1) & Bitwise AND
2) | Bitwise OR
3) ~ Bitwise NOT
4) ^ XOR
5) << left shift
6) >> Right shift

Q. 8. What is the use of comma " , " operator ?

It is used to link the related expressions together. A comma-linked list of expressions are evaluated left to right & the value of right most expression is value of the combined expression.

In loops :-

```
for (n=1, m=10, n<=m; n++, m++)
while (c=getchar(), c!='10')
```

Q.9. How can you read the data from keyboard & display on the screen?

We can give value to the variable through keyboard by scanf function. It's a general input function available in C & is very similar in concept to printf function. It works much like an input statement in basic.

Scanf's general formate is :-

```
scanf("control string", &variable 1, &variable 2);
```

Control string contain the format of data being received.

scanf ("%d", &no.1);

when the statement is encountered by the computer the execution stop & waits for the value of variable number to be typed in.

X X X

Sardar Vallabhbhai Patel Institute of Technology, Vasad
B.E. Semester – 1
Subject: Programming for Problem Solving (3110003)
Assignment – 3 (Unit – 3)

1. Explain "if …else…if" ladder of C with neat diagram and a brief program code.
2. Compare for - loop and while - loop with the help of program to solve addition of 10 odd numbers (even numbers are ignored in computation).
3. Explain the syntax of "switch…case" statement. Define rules for "switch" statement. Write a program using "switch…case" statement.
4. Differentiate in between "if …else if .... else " ladder with „switch…case".
5. Write a C program to read 10 numbers from user and find sum, maximum and average of them.
6. Explain the various conditional statements. (such as if, if – else, if – else if – else etc.)
7. Differentiate between "while" & "do-while" looping construct.

OR

8. Differentiate between "entry controlled" and "exit controlled" looping constructs.

9. Explain "break" & "continue" statement used in C program. Support your answer with a C code for each.

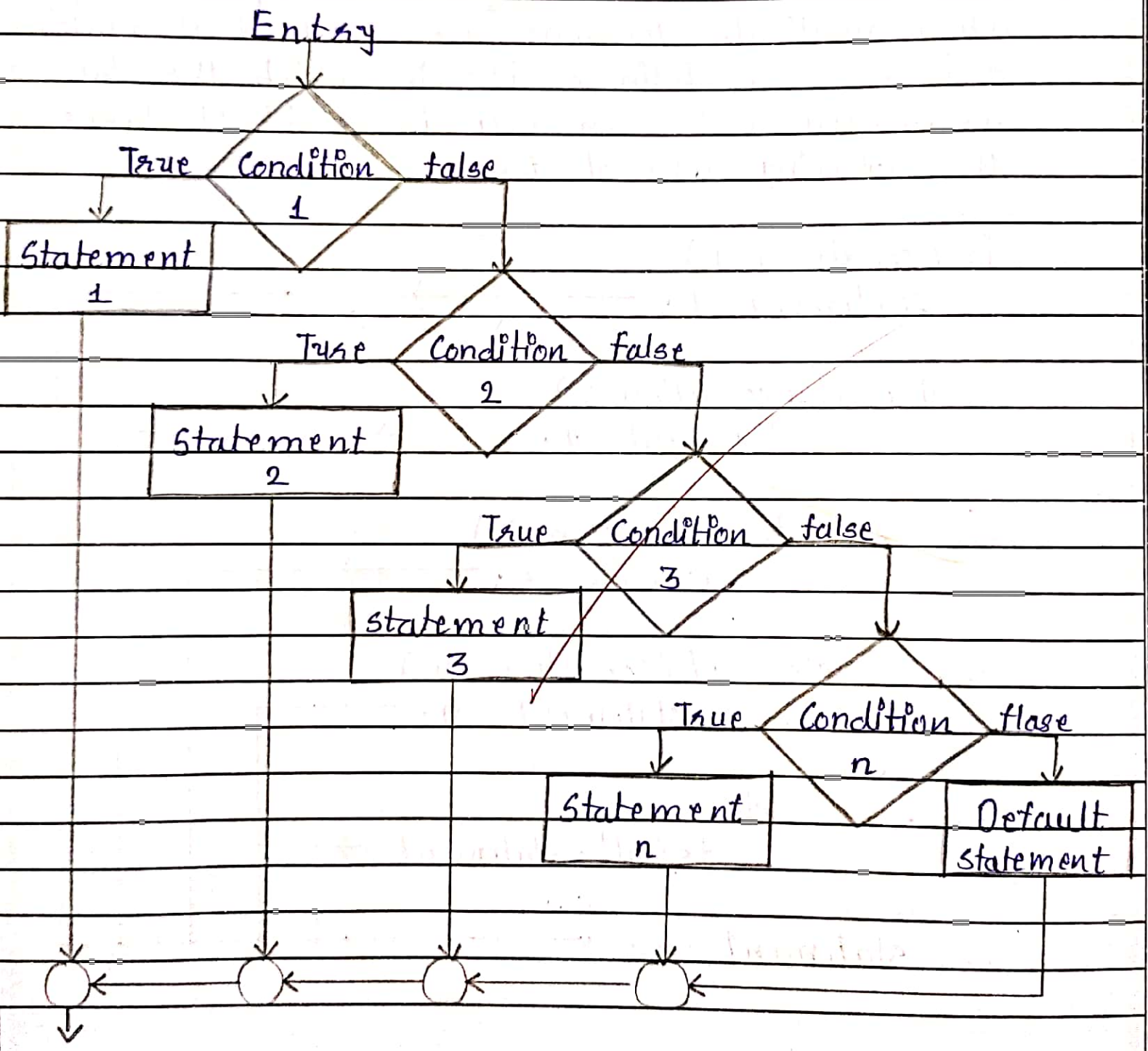
Assignment - 3 (Unit - 3)

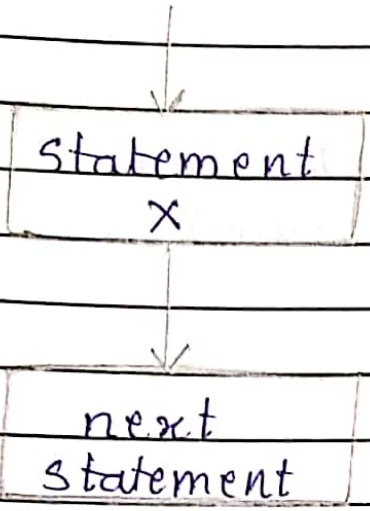
Q. 1. Explain "if... else... if" ladder of C with neat diagram and a brief program code.

There is a way of putting ifs together when multipath decisions are involved. A multipath decision is a chain of ifs in which the statement associated with each else is an if. It takes the following general form.

```
if (condition 1)
    statement - 1;

else if (condition 2)
        statement - 2;

    else if (condition 3)
            statement - 3;

        else if (condition n)
                statement - n;

            else
                default - statement;

statement - x;
```

This construct is known as the else if ladder. The conditions are evaluated from the top, downwards. When all the n conditions became false, then the final else containing the default - statement will be executed.





Example

Q. 2. Compare for two and while-loop with the help of program to solve addition of 10 odd numbers.

⇒ For loop

```
#include<stdio.h>
void main()
{
   int i, n, sum;
   printf("Enter the value = ");
   scanf("%d", &n);
   printf("\n\n");
   for(i=1, i<=10; i++)
   {
     if(i%2 != 0)
     {
       printf("%d", i);
       sum = sum + i;
     }
```

```
}
printf (" The sum of shown 10 odd
        numbers is = %d ", sum)
}
```

⟹ while loop

```
#include <stdio.h>
void main()
{
  int i, n, sum;
  printf (" Enter the value = ");
  scanf (" %d", &n);
  printf ("\n\n");
  i = 1;
  while (i <= 10)
  {
    if (i % 2 != 0)
    {
      printf (" %d", i);
      sum = sum + i;
      i++;
    }
  }
  printf (" The sum of shown 10 odd
          numbers is = %d", sum);
}
```

output

Enter the value = 20
1, 3, 5, 7, 9, 11, 13, 15, 17, 19
The sum of shown 10 odd number is = 100

Q. 3. Explain the syntax of "switch... case" statement. Define rules for "switch" statement. Write a program using "switch... case" statement.

```
Switch (expression)
{
  case value 1:-
              block -1
              break;
  case value 2:-
              block - 2
              break;
  default :
              default - block
              break;
}
```

⇒ statement - X

- The expression is an integer expression or characters.

- value-1, value-2,... are constats or constant expressions and are known ase case labtes. labels.
- Each of these values should be unique whithin a switch statement.
- block-1, block-2,... are statement lists and may contain zero or more statements case labels end with a colon (:).

⟹ Rules for switch statement.

- The switch expression must be an integral type.
- case lables must be constants or constant expressions.
- case labte: labels must end with colon.
- The breals statement tranfes the control of the switch statement.
- The break statement is optional.
- The default label is optional. If present, it will be executed when the expression doesn't find a matching case label.
- There can be at most one default label.
- It is permitted to nest switch statements. The default may be placed anywhere but usually placed at the end.

```
#include<stdio.h>
void main()
{
   char ch;
   int a,b,c,d;
   printf("Enter any two value = ");
   scanf("%d %d", &a, &b);
   printf(" + is for addition \n");
   printf(" - is for substraction \n");
   printf(" Now, Enter your choice = ");
   ch = getch();
   switch(ch)
   {
     case '+':
     {
       c = a+b;
       {
         printf("sum is %d", c);
       }
     }
     case '-':
     {
       d = a-b;
       {
         printf("sub is %d", d);
       }
     }
```

```
        default :
        {
          printf("Entered choice is invalid.");
        }
    }
}
```

Output

```
Enter any two numbers = 5 3
+ is for addition.
- is for substraction.
Now, Enter your choice = +
sum is 8
```

Q.4. Differentiate in between "if... else" ladder and "switch... case" statement.

	"if.... else" ladder	switch.... case statement
1.	In else... if ladder, the control goes through the every else if statement whithin it finds true value.	In switch... case statement the value of the switch the control jumps to the corresponding case.

2.	There is no use of break statement in else...if ladder	The break statement is mainly used in switch statement.
3.	It is more flexible than switch statement.	It is less flexible in compare else...if ladder.
4.	The code needs to be processed in the order determined by the user.	Each case is independent of the previous one.
5.	It works on the basis of true-false basis.	It works on the basis of equality operator.

Q.5. Write a C-program to read 10 numbers from user and find sum and average of them.

```
#include<stdio.h>
void main()
{
  float ave;
  int n, i, max, sum=0;
  max=0;
```

```
for (i=1, i<=10, i++)
{
  printf("Enter ten no. = ");
  scanf("%d", &n);
  if (max < n)
  {
    max = n;
  }
  printf("maximum = %d", max);
  sum = sum + n;
  ave = sum / 10;
}
printf("The sum of given numbers
        = %d", sum);
printf("The average of given numbers
        = %f", ave);
}
```

Output

```
Enter ten no. = 1 2 3 4 5 6 7 8 9 10
maximum = 10
The sum of given number = 55
The average of given number = 5.5
```

Q.6 Explain the various conditional statement:

⇒ If statement

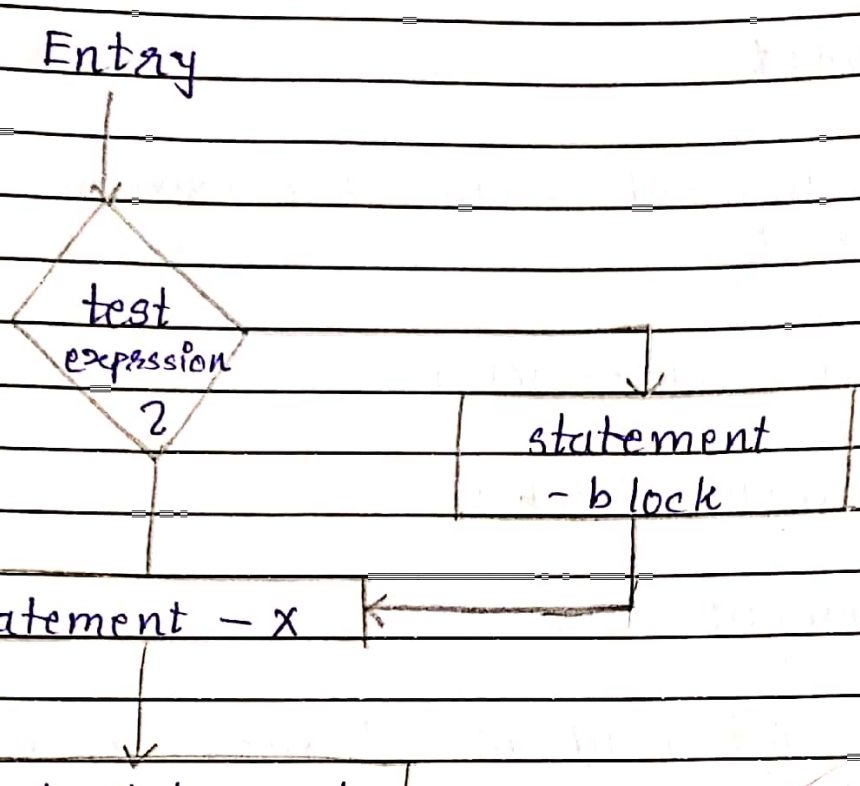
The general form of a simple if statement is

```
if (test expression)
{
    statement-block;
}
    statement-x;
```

The 'statement-block' may be a single statement or a group of statements. If the test expression is true, the statement-block will be executed; otherwise the statement-block will be skipped and the execution will jump to the statement-x. When the condition is true both the statement-block and the statement-x are executed in sequence.

ex.

```
.....
if (code == 1)
    boy = boy + 1;

if (code == 2)
    girl = girl + 1;
.....
```



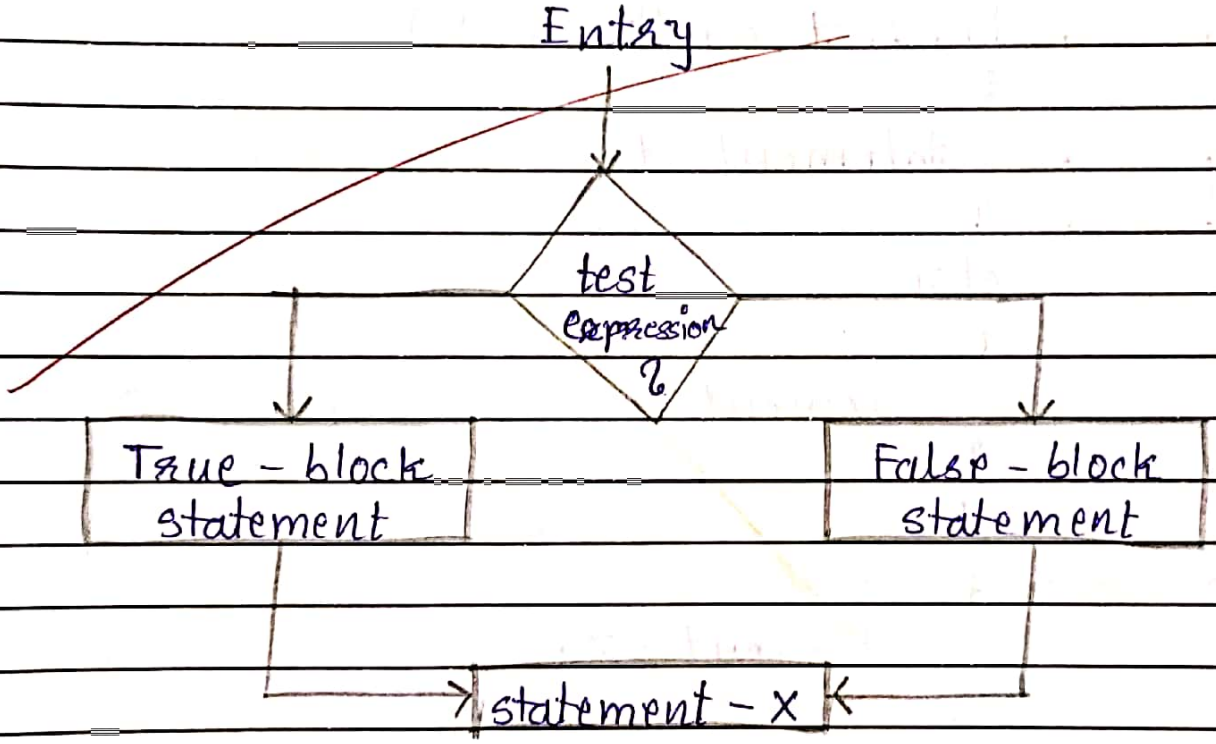
⇒ if.... else statement

The if.... else statement is an extension of the simple if statement. The general form is,

```
if (test expression)
{
   True - block statement
}
else
{
   False - block statement
}
```

statement - x

If the test expression is true, then the true-block statement, immediately following the if statements are executed; otherwise, the false-block statement are executed. In either case, either true-block or false-block will be executed, not both.



e.g.

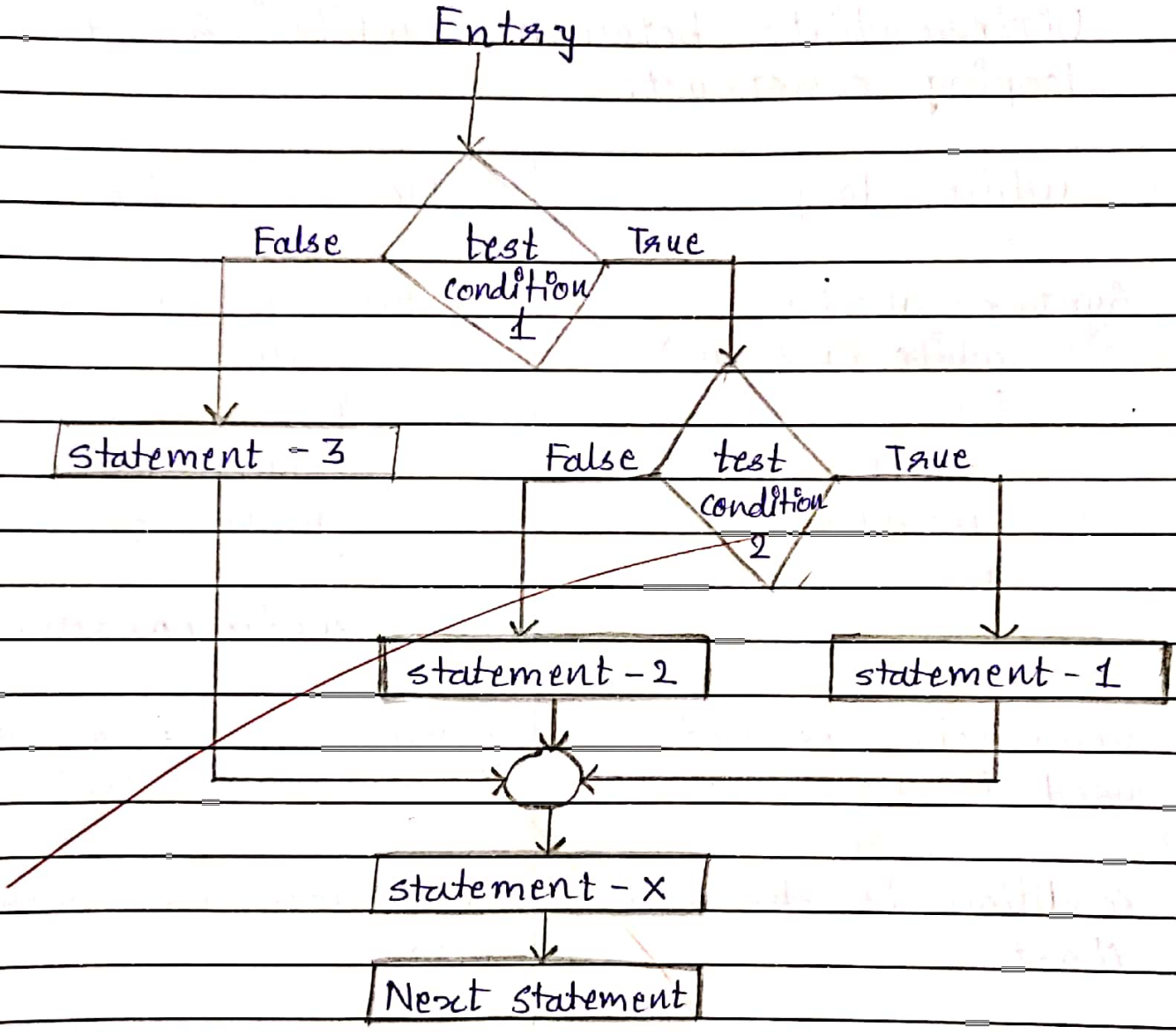
```
.....
if (code == 1)
    boy = boy + 1;
else (code == 2)
    girl = girl + 1;
.....
```

⟹ Nesting of if...else statement.

When a series of decisions are involved, we may have to use more then one if...else statement in nested form as shown below :-

```
if (test condition -1)
  {
  if (test condition - 2)
    {
      statement - 1;
    }
    else
    {
      statement - 2;
    }
    else
    {
      statement - 3;
    }
}
statement - X;
```

The logic of execution is illustrated in following figure.



If the condition-1 is false, the statement-3 will be executed, otherwise it continues to perform the second test. If the condition-2 is true, the statement-1 will be evaluated, otherwise the statement-2 will be evaluated and then the control is transfered to the statement-x.

Q.7. Differentiate between "while" & "do-while" looping construct.

	while loop	Do while loop

1.	Syntax ; n=1 ;
	while (n<=10)
	{
	statement ;
	n=n+1 ;
	}
		Syntax ; n=1 ;
		do
		{
		statement ;
		n=n+1 ;
		}
		while (n<=10) ;

2.	Semi colon is not used in while loop.	Semi colon is used in do-while loop.
3.	condition is checked first.	condition is checked later.
4.	Since condition is checked first, statement may or may not get executed.	Since condition is chacked later, the body statements will execute at least once.
5.	The main feature of the while loop is, it's an entry controlled loop.	The main feature of the do-while loop is it is an exit controlled loop.

Q.9. Explain "break" & "continue" statement used in C program.

⇒ The break statement at the end of the each block signals the end of a particular case and causes an exist from the switch statement. The break statement transfers the control out of the switch statement. The break statement is optional. A loop can be accomplished by using the break statement. When a break statement is countered inside a loop, the loop is immediately exited and the program continues with the statement immediately following the loop. When the loops are nested, the break would only exit from the loop containing it. That is, the break will exist only a single loop.

e.g:
```
#include <stdio.h>
void main ()
{
  int num=0;
  while (num <= 50)
  {
    printf ("Value of variable num is=%d", num);
    if (num == 2)
    {
      break;
```

```
    }
    num++;
  }
  printf("finished");
}
```

Output

```
Value of variable num is = 0
Value of variable num is = 1
Value of variable num is = 2
Finished
```

⇒ Like the break, C supports another similar statement called the continue statement, However, unlike the break which cause the loop to be terminated the continue, as the name implies, cause the loop to be continued with the next iteration after skipping any statement in between. The use of this in while and do-while loops, continue causes the control to go directly to the test-condition and then to continue the iteration process.

e.g.

```
#include <stdio.h>
void main()
{
  int j;
  for (j=0, j<=8, j++)
  {
    if (j==4)
    {
      continue;
    }
    printf("%d", j);
  }
}
```

Output

0 1 2 3 5 6 7 8

Sardar Vallabhbhai Patel Institute of Technology, Vasad

B.E. Semester – 1

Subject: Programming for Problem Solving (3110003)

Assignment – 4 (Unit – 4)

Topic: Arrays and Strings

1. What is array? Give example using C code and list down advantages of array.

OR

2. Explain Derived data types. Support your answer with appropriate C code.
3. Write a brief account on dimensions of array with appropriate examples. (Hint 1-Dimension, 2-Dimension, Multi-dimension)
4. Explain how string is defined in C. List the various inbuilt string functions.
5. What is string? In how many ways can you accept data in a string? Also mention the significance of null character.
6. Explain getch(), getchar(), gets(), puts().
7. Explain following string manipulation functions. strcmp( ), strlen( ), strcat( ), strstr( ), strcpy().
8. Write your own C code for above mentioned string in-built functions.
9. Write a C program to read 10 numbers from user and store them in an array. Display Sum, Minimum and Average of the numbers
10. Write a program to check whether a given string is palindrome or not.
11. Write a C program to multiply two N X N Matrix.

OR

12. Write a program in c for multiplication of two matrix
13. Write a program to accept a string and count the numbers of vowels present in a string.
14. Write a program to reverse the input string.
15. Write a program to concatenate two strings without using built in function.
16. Write a program for transposing a matrix.
17. Write a program to compute row-wise sum of 2-D Array and then store row-wise sum into 1-D array.

Assignment - 4 - (Unit - 4)

Q.1 What is array? Give example using C code and list down advantages of array.

Array is used to process such large amount of data. We need a powerful data like array type, that would facilitate efficient storing, accesing & manipulation of data items.

An array is a fixed-size sequenced collection of elements of the same data type. It is simply a grouping of like type data. It is used to represent a list of number or a list of names.

Array helps us in various field :-

- List of temperatures record every hour in a day or a month or a year.
- List of employees in an organization.
- List of products and their cost sold by a store.

- Test scores of a class of students.
- Table of daily rainful data.

ex:
```
#include <stdio.h>
int main()
{
    char name[50];
    printf("Enter your name");
    gets("%s", name);
    puts(name);
    printf("string = %s \n", name);
}
```

Output

```
Enter your name Parmar Nisarg.
string Parmar Nisarg.
```

Q.3: Write a brief account on dimentions of array with appopriate example.

We can use arrays to represent not only simple list of values but also tables of data in two, three or more dimentions.

a) One-Dimensional Array.

A list of items can be given one variable name using only on subscipt & such a variable is called a single subscripted variable or a one-dimension array.

The general form of array declaration is :-

type variable-name [size]

The subscript variable x; refers to the i^{th} element of x, in C single subscripted variable xi can be expressed as :-

x[1], x[2], x[3], ..., x[n]

The subscript begin with number 0.

e.g. We want to represent a set of five no.

```
int a[5] = { 20, 30, 50, 70, 90 };
```

a[5]: array name; size of array

b) Two - Dimensional Array.

C allows us to define such tables of items by using two-dimensional array.

Two-dimensional array are declared as:-

type array_name [row_size][column_size];

Two-dimensional arrays are stored in memory. As with the single-dimensional arrays, each dimension of the array is indexed from zero to it's maximum size minus one; the first index selects the row & the second index selects the column within that row.

e.x:
```
#include <stdio.h>
void main()
{
 int a[3][3], i, j, sum;
 Printf("Enter the value:");
 for(i=0; i<3, i++)
 {
  for(j=0; j<3, j++)
  {
   Scanf("%d", &[i][j]);
```

```
            sum = sum + a[i][j];
        }
    }
    printf("Sum is %d", sum);
}
```

Output

```
Enter the value = 10
2 5 10 11 13 3 7 18 9 20
sum is 98
```

c) Multi-Dimensional Array.

C allows arrays of three or more dimensions. The exact limit is determined by the compiler. The general form of a multi-dimensional array is

```
type array-name[s1][s2]....[sn];
```

e.g: Si is the size of the i^{th} dimension.

```
int survey[3][5][2];
float table[5][4][6][3];
```

Q.4. Explain how string is defined in C. List the various inbuilt string functions.

C does not support strings as a data type. It allows us to represent string as character arrays. Therefore a string variable is any valid C variable name and is always declared as an array of characters.

The general form of declaration of a string variable is :-

```
char string_name[size];
```

The size determines the number of characters in the string_name.

```
char city[30]; char name[50];
```

when the compiler assigns a character string to a character array, it automatically supplies a null character array. C also permits us to initialize a churaten array without specifying the number of element.

- List of inbuilt string functions :-

strcpy(), strcat(), strnicmp(), strncpy(), strcmp(), strrev(), strlen(), strncmp(), strchr(), strncat(), stricmp()

Q.5. What is string ? In how many ways can you accept data in string ? Also mention the significante of null character.

A string is a sequence of charakteers that is treated as a single data item. It can also contain spaces & numbers.

Like numeric arrays character arrays may be initiallized when they are declared. C permits a character array to be intialized in either of following 2 terms :-

```
char city[9] = "New York";
char city[9] = {'N','e','w',' ','Y','o','r','k'};
```

The reason that city had to be

9 elements long in that the string New York. contains 8 characters and one element space is provided for the null terminator.

A null character is a character with all its bits set to zero, It has a numeric value of zero & can be used to represent the end of a string of characters, such as a word. This helps to determine the length of strings.

Q.6: Explain getch(), getchar(), gets(), puts().

a) getch()

It is a way to get a user inputted character. It can be used to hold programme execution. getch() function is used to provide an aclanowledgement for characters.

ex getch();

b) getchar()

We can use getchar() function repeatedly to read successive single characters from the input & place them into a character array.

Thus, an entire line of text can be read & stored in an array. The reading is terminated when the newline character ("\n") is entered and the null character is then inserted at the end of the string.

ex: char ch;
ch = getchar();

c) gets()

More convenient method of reading a string of text containing whitespace is to use the library function gets available in the <stdio.h>.

ex gets(str);

d) puts()

A convenient way of printing

string values is to use the function puts declared in the header file <stdio.h>.

ex: puts (str);

Q.7: Explain following string manipulation functions.

a) strcmp()

int strcmp(s1, s2)

It will compare both the string & if the string are equal, it will return 0.

$s_1 = s_2$, return 0
$s_1 > s_2$, return > 0
$s_1 < s_2$, return < 0

b) strlen()

int strlen(s1)

It will find a length of string.

c) strcat()

strcat (S1, S2)

It will merge both the string & return S1.

d) strcpy ()

strcpy (char *S1, char *S2)

It will copy string 1 into string 2.

e) strstr ()

The strstr () function searches the given string in the specified main string & returns the pointer to the first occurrence of the given string.

Q.8. Write your own C code for above mentioned string in-built function.

```
#include <stdio.h>
#include <string.h>
void main()
{
   int t;
   char nm[50];
```

```
    printf("Enter string : \n");
    gets(nm);
    t = strlen(nm);
    printf("The string length is %d.", t);
}
```

Output

```
Enter string : Nisarg
The string length is 6.
```

Q.9 Write a C program to read 10 numbers from user and store them in an array. Display Sum, Minimum and Average of the number.

```
#include <stdio.h>
Void main()
{
    int a[10], i, sum=0, min;
    float ave;
    printf("Enter 10 numbers : ");
    for(i=0; i<10, i++)
    {
        scanf("%d", &a[i]);
        sum = sum + a[i];
        min = a[i];
```

```
    ave = sum/10.;
    if (min > a[i])
    min = a[i];
    }
    printf(" The sum of numbers is %d", sum);
    printf(" The average of numbers is %f", ave);
    printf(" The minimum number is %d.", min);
}
```

Output

```
Enter 10 numbers :
4 6 1 2 8 10 7 3 5 9
The sum of numbers is 55.
The average of numbers is 5.5
The minimum number is 1
```

Q.10 Write a program to check whether a given string palindrome or not.

```
#include <stdio.h>
void isPalindrome(char str[])
{
    int l=0; h = strlen(str)-1;
    while (h > l)
    {
        if (str[l++] != str[h--])
```

```
        {
            printf("%s is not palindrome", str
        }
    }
    printf("%s is palindrome", str);
    return 0;
}
```

Q.11 Write a C program to multiply two N x N matrix.

```
#include <stdio.h>
int main()
{
    int a[10][10], b[10][10], result[10][10], r1,
    c1, r2, c2, i, j, k;
    Printf("%d %d", &r1, &c1);
    Printf("Enter rows and column for second
            matrix :");
    scanf("%d %d", &r2, &c2);
    while (c1 != r2)
    {
        Printf("Error ! Column of first matrix
                is not equal to row of second
                \n");
        printf("Enter rows and column for
                first matrix :");
```

```
    scanf("%d %d", &r1, &c1);
    Printf("Enter row and column for
            second matrix:");
    scanf("%d %d", &r2, &c2);
}
printf("\n Enter element of matrix 1: \n");
For (i=0; i<r1; ++i)
For (j=0; j<c1; ++j)
{
    printf("Enter element a %d %d",
            i+1, j+1);
    scanf("%d", &a[i][j]);
}
printf("\n Enter element of matrix 2: \n");
For (i=0; i<r2; ++i)
For (j=0; j<c2; ++j)
{
    printf("Enter element b %d %d",
            i+1, j+1);
    scanf("%d", &b[i][j]);
}
For (i=0; i<r1; ++i)
For (j=0; j<c2; ++j)
{
    result[i][j]=0;
}
```

```
For (i=0; i<r1; ++i)
For (j=0; j<c2; ++j)
For (k=0, k<c1; ++k)
{
  result[i][j]+= a[i][k] * b[k][j];
}
printf("\n Output matrix : \n");
For (i=0; i<r1; ++i).
For (j=0; j<c2; ++j)
{
  printf("%d", result[i][j]);
  if (j==c2-1)
  printf("\n \n");
}
return 0;
}
```

output

```
Enter row and column for first matrix : 3 2
Enter row and column for second matrix : 3 2
Error ! Colum of first matrix is not
equal to row of second.

Enter row and column for first matrix : 2 3
Enter row and column for second matrix : 3 2
```

Enter element of matrix 1:
$a_{11} = 3$, $a_{12} = -2$, $a_{13} = 5$
$a_{21} = 3$, $a_{22} = 0$, $a_{23} = 4$

Enter element of matrix 2:
$a_{11} = 2$, $a_{12} = 3$,
$a_{21} = -9$, $a_{22} = 0$,
$a_{31} = 0$, $a_{32} = 4$

Output Matrix:

24 29
6 25

Q.13 Write a program to accept a string and count the number of vowels present in a string.

```
#include <stdio.h>
int main()
{
  int c=0, count=0;
  char s[1000];
  printf("Input a string \n");
  gets(s);
  while (s[c] != '\0')
  {
```

```
if (s[c] == 'a' || s[c] == 'A'
    s[c] == 'e' || s[c] == 'E'
    s[c] == 'i' || s[c] == 'I'
    s[c] == 'o' || s[c] == 'O'
    s[c] == 'u' || s[c] == 'U' )
count++;
c++;
}
printf("Number of vowels in the
        entered string %d", count);
return 0;
}
```

Output

```
You Can learn Anything.
Number of volwels in the string = 7.
```

Q.14. Write a program to revers the input string.

```
#include<stdio.h>
#include<string.h>
int main()
{
  char arr[100];
```

```
printf("Enter a string to revers \n");
gets(arr);
strrev(arr);
printf("Revers of the string is %s \n", arr);
return 0;
}
```

Output.

```
Enter a string to revers.
computer is an amazing device.

Reverse of enterd string is
.ecived gnizama na si retupmoc
```

Q.15. Write a program to concatenate two string without using built in function.

```
#include <stdio.h>
#include <string.h>
void concat(char[], char[]);
int main()
{
  char s1[50], s2[30];
  printf("Enter string 1 \n");
  gets(s1);
```

```
printf("Enter string 2 \n");
gets(s2);
concat(s1, s2);
printf("\n concated string is %s", s1);
return 0;
}

Void concat(char s1[], char s2[])
{
  int i, j;
  i = strlen(s1);
  For(j=0; s2[j] = '\0'; ++i, s1[i] = s2[j])
}
```

Output

```
Enter string 1 : Ankit
Enter string 2 : Patel
Concated string : AnkitPatel
```

Q.16 Write a program for transposing matrix.

```
#include <stdio.h>
Int main()
{
  int m, n, c, d, matrix[10][10], transpose
                                  [10][10];
```

```
printf("Enter the number of row and
        column of matrix = \n");
scanf("%d %d",&m,&n);
printf("Enter element of the matrix \n");
For (c=0; c<m; c++)
For (d=0; d<n; d++)
scanf("%d", &matrix[c][d]);
For (c=0; c<m; c++)
For (d=0; d<n; d++)
transpose[d][c] = matrix[c][d];
printf("Transpose of the matrix : \n");
For (c=0; c<n; c++)
{
   For (d=0; d<m; d++)
   printf("%d \t", transpose[c][d]);
   printf("\n");
}
return 0;
}
```

Output

```
Enter the number of row and column : 2 3
Enter the element of matrix
1   2   3
4   5   6
```

Transpose of entered matrix

1 4
2 5
3 6

Q.17 Write a program to compute row-wise sum of 2-D Array and then store row-wise sum into 1-D array.

```
#include <stdio.h>
int main()
{
  int i, j, rows, columns, a[10][10], sum;
  printf("Please enter number of rows
          and columns \n");
  scanf("%d %d", &i, &j);
  printf("Please enter number of rows
          and columns \n");
  For (rows=0; rows<i; ++rows)
  {
    For (column=0; columns<j; ++columns)
    {
      scanf("%d", &a[rows][columns]);
    }
  }
  For (rows=0; rows<i; ++rows)
```

```
{
    sum = 0
    For (columns = 0; columns < j; ++columns)
    {
        sum = sum + a[rows][columns];
    }
    Printf("sum of elements of a row
            in matrix = %d \n", a[i]);
    return 0;
  }
}
```

Output

```
Please enter number of row and column = 4 3
Please enter number of row and column
    10   20   30
    12   22   32
    44   55   121
    33   91   73
Sum of element of a row in matrix
    60   66   220   197
```

Sardar Vallabhbhai Patel Institute of Technology, Vasad
B.E. Semester – 1
Subject: Programming for Problem Solving (3110003) Assignment – 5
Topic: Functions, Recursion, Structures, Pointers, Dynamic memory allocation and Files **Note: Write any 10 questions from the given list**

1. What is function? Explain the function definition, function prototype and function call with relative example.
2. Explain Call by value and Call by reference. (OR) Compare "call by value" & "call by reference/address" with suitable example.
3. Briefly discuss about scope of variable.
4. Explain storage classes.
5. What is Recursion? Write a program to find factorial of a given number using recursion. OR
6. What do you mean by recursive function? Write a program in c to find factorial of a number using recursive function.
7. List out the categories of functions of C. (OR Explain different type of functions used in c language).
8. What is a pointer? How can pointers be used in a program?
9. How can we assign the value of the variable through pointers?
10. What is array pointer? What is the difference between array pointer (arr[ ]) and normal pointer variable (p)? (int arr[5]; int *p;)
11. What are indirect pointer and array of pointer?
12. How can you pass pointer in function?
13. Explain
    a. Structure
    b. Array of Structure and pointer to structure
    c. Structure vs Union
14. Explain the following in-built functions:
    a. malloc()
    b. calloc()
    c. realloc()
    d. free()
    e. fopen()
    f. fseek()
    g. ftell()
15. Write a program using function to find maximum if two numbers.
16. Write a function which takes 2 numbers as parameters and returns the gcd of the 2 numbers. Call the function in main().
17. Write a function to swap 2 numbers.
18. Write a function to find whether the string is palindrome or not.

Assignment - 5 (Unit - 5)

Q.1. What is function ? Explain the function defination prototype and function call with relative example.

Function is a block of statements that performs a specific task.

Function defination also known as function implementation shall include following elements.

1. Function name
2. Function type
3. List of parameters
4. Local variable declaration
5. Function statement
6. Return statement

All 6 element are grouped into two parts, namely.

1. Function header (1st three)
2. Function body (2nd three)

Function prototype is simply the declaration of a function that specifies function names, parameters & return type. It doesn't contain function body.

Function call are two ways that a c-function can be called from a program.

1. Call by value.
2. Call by reference.

1. Call by value.

The value of the variable is passed to the function as parameter. The value of actual parameter can't be modified by formal parameter. Different memory is allocated for both actual & formal parameter.

e.g:
```
#include <stdio.h>
void swap (int a, int b);
int main ()
{
    {
    int m=22 ; n=44;
```

```
    printf(" Value before swap m=%d
            and n=%d \n", m, n);
    swap (m, n);
}
void swap (int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
    printf(" \n Value after swap m=%d
            and n=%d \n", a, b);
}
    return 0;
}
```

Output

Value before swap m = 22 & n = 44
Value after swap m = 44 & n = 22

2. Call by refrence.

The address of variable is passed to the function as parameter. The value of the actual parameter can be modified by formal parameter same

memory is used for both actual & normal parameter since only address is used by both parameters.

e.g:
```
#include <stdio.h>
void swap (int a, int b)
int main ()
{
    {
      int m = 22, n = 44;
      printf (" Value before swap m = %d
              and n = %d \n", m, n);
      swap ("&n, &m");
    }
    void swap (int a, int b)
    {
      int temp;
      temp = a;
      a = b;
      b = temp;
      printf (" \n values after swap a = %d
              and b = %d ", a, b);
    }
    return 0;
}
```

Output

Value before swap m=22 & n=44
Value after swap m=44 & n=22

Q.2. Explain call by value and call by refrence.

Answer in Q.1.

Q.3. Briefly discuss about scope of variable.

Scope of variable is the position of the program where a defined variable can be excessed. The variable scope defined the visibility of variable in the program. Scope of variable depends on position of variable declaration.

In C-programming language, a variable can be declared in three ways.

1. Before the function definition.
2. Inside the function of block.
3. The function definition parameter.

Q.4. Explain storage classes.

A storage class represents the visibility and a location of a variable. It tells from what part of code we can access a variable.

A storage class is used to describe the following thing.

- Variable scope.
- Location where the variable will be stored.
- unitilized value of variable
- Life time of a variable.
- who can acces a variable.

There are total 4 type of standard storage classes.

- Auto - It is a default storage class.
- Extern - It is a global variable.
- static - It is a local variable which is capable of returning a value even when control is transfered to the function call.
- register - It is a variable which is stored inside a register.

Q.5. What is recursion ? Write a program to find factorial of a given number using recursion.

Recursion is a process of repeating items in a self-similar way. In programming langauges, if a program allows you to call a function inside the same function, then it is called a recursive call of the function.

e.x:

```
#include <stdio.h>
int factorial
{
  if (i <=1)
  {
    return 1;
  }
  return i * factor (i-1);
}
int main ()
{
  int i = 6;
  printf ("factorial of 6 is %d \n",
                    factorial (i) );
  return 0;
}
```

Output:

Factorial of 6 is 720.

Q.7: List out the categories of function of C.

Function of C can be catogorized in two ways

(1) Library function
(2) User / defined function

1. Library Function.

Function defined by C distributors and are included with compilers are known as library functions. These functions are built in, pre-compiled and ready to use.

eg: printf( ), scanf( ), etc.

2. User defined function.

Function defined by an end programmer is known as user

defined function. A programmer can define any no. of function depending on the need. We can also compile a set of library function and use them later in another program.

(a) Function with no return and no argument.
(b) Function with no return but argument.
(c) Function with return but no argument.
(d) Function with return and argument.

Q.8. What is a pointer? How can pointers be used in a program?

A pointer is a variable whose value is the address of another variable i.e. Direct address of the memory location like any variable or constant, you must declare a pointer before using it to store any variable address

General form :- type * variable name

(a) We define a pointer variable.
(b) Assign the address of variable to a pointer.
(c) Finally access the value at the address variable in the pointer variable.

e.g.

```
#include<stdio.h>
int main()
{
    int var = 20;
    int *ip;
    ip = &var;
    printf("Address of var variable %d \n",
                                    &var);
    printf("Address stored in ip variable
                        %d \n", ip);
    printf("Value of *ip variable %d \n",
                                    *ip);
    return 0;
}
```

Output

```
Address of var variable = bffd8b3c
Address stored in ip variable = bffd8b3c
Value of *ip variable = 20
```

Q.9. What is array pointer? What is the difference between array pointer (arr[]) and normal pointer variable?

In computer programming an array of pointers is an indexed set of variable where the variable are pointers.

Pointer are an important tool in computer science for creating, using & destroying all types of data structure. An array of pointers is useful for the same reason that all the arrays are useful, it allows you to numerically index a large set of variable.

	Array	Pointer
1.	type var_name [size]	type * var_name
2.	An array of pointer can be generated.	A pointer to an array can be generated.

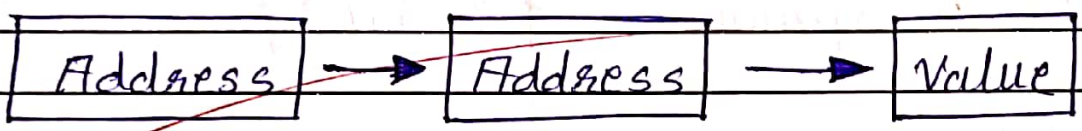
3.	Stores the value of variable of homogeneous data type.	Stores the address of another variable of same data type as pointer's data type.
4.	Store the address of variable.	Pointers are specially designed to store the address of variable.
5.	An array can store the number of element mentioned on size of array variable.	It can store the address of only one variable at a time.

Q.10 How can we assign the value of the variable through pointers?

(1) Declare a normal variable, assign the value.
(2) Declare a pointer variable with the same type as normal variable.
(3) Initialize the pointer variable with the address of normal variable.
(4) Access the value of the variable by using (*) it is known as operator.

Q.11. What are indirect pointer and array of pointer ?

The closest thing related to it is pointer to pointer. This technique is sometimes called a handle and is useful in certain situation where the operating system wants to be able to move blocks of memory on the heap around of its discretion.



Q.12. How can you pass pointer in function ?

Pass by pointers means to pass a pointer argument in the calling function to the corresponding formal parameter of the called function. The called function can modify the values of the variable to which the pointer argument points.

When we use pass by pointer, a copy of pointer Ps passed to the function.

If you modify the pointer the called function you only modify the copy of the pointer, but original pointer remain unmodified & still points to the original variable.

When the function swapnum() is called the value of variable a & b are exchanged because they are passed by pointers.

e.g.

```
#include <stdio.h>
void swapnum (int *i, int *j)
{
    int temp = *i;
    *i = *j;
    *j = temp;
}
int main ()
{
    int a = 10;
    int b = 20;
    swapnum (&a, &b);
    printf("A is %d & B is %d \n");
    return 0;
}
```

Output.

A is 20 & B is 10.

Q.13. Explain the following term

(i) Structure.

Structure is a user defined datatype in C - language which types together. Structure helps to construct a complex data type which is more meaningful. It can store data of any type, which is practically more useful.

(ii) Array of structure and pointer to structure.

An array of structure in C can be defined as the collection of multiple structure variable where each variable contains information about different entities. The array of structures in C are used to store information about multiple entities of different data type. The array of structure is also known as the collection of structures.

(iii) Structure Vs Union

	Structure	Union
1.	The keyword structure is used to defined structure.	The keyword union is used to defined union.
2.	Size of structure is greater then or equal to the sum of sizes of its members.	Size of union is equal to the size of largest member.
3.	Each member within a structure is assigned unique storage are of location.	Memory allocated is shared by individual members of union.
4.	Alter the value of a member will not after member of the structure.	Alterning the value of any of the member after other member value.

5.	Individual member can be accessed at a time.	Only one member can be accessed at a time.
6.	Several member can unitilized at once.	Only first member can be unitilized.

Q.14. Explain the following in-built function.

(i) malloc ()

"malloc" or "memory allocation" method is used to dynamically allocate a single large block of memory with the specified size. It returns a pointer of type void which can be cast into a pointer of any form.

(ii) calloc ()

"calloc" or "contiguous allocation" method in C is used to dynamically allocate the specified number of block of memory of the specified type. It initialize each block with a default value '0'

(iii) realloc ()

"re-allocation" in C is used to dynamically change the memory allocation of the previous allocated memory. In other words, it can be used to dynamically re-allocation memory.

(iv) free ()

"free" method in C is used to dynamically de-allocate the memory. The memory allocated using functions malloc() & calloc() is not de-allocated on their own. Hence, the free() method is used, whenever the dynamic memory allocation of memory by freeing it.

(v) fopen ()

fopen() function creates a new file or opens an existing file.

(vi) fseek()

fseek() function moves file pointer position to given location.

(vii) ftell()

ftell() function gives a current position of file pointer.

Q.15. Write a program using function to find maximum if two numbers.

```
#include <stdio.h>
int max (int num 1, int num 2);
int main()
{
   int num 1, num 2, maximum;
   printf ("Enter any two no. : \n");
   scanf (" %d %d", &num 1, &num 2);
   maximum = max (num 1, num 2);
   printf ("\n maximum = %d ", &maximum);
   return 0;
}
int max (int num 1, num 2)
{
   return (num 1 > num 2) ? num 1 : num 2;
}
```

Output:

```
Enter any two no. : 10 30
```

Q-16. Write a function which takes 2 numbers as parameters and returns the gcd of the 2 numbers. Call the function in main().

```
#include <stdio.h>
int gcd (int a, int b)
{
  if (a==0) {
  return b;
  if (b==0)
  return a;
  if (a==b)
  return a;
  if (a>b)
  return gcd (a-b, b);
  return gcd (a, b-a);
}
int main()
{
  int a=98 , b=56
  printf("GCD of %d & %d is %d",
                    a, b, gcd (a,b));
  return 0;
}
```

Output

GCD of 98 & 56 is 14.

Q.17 Write a function to swap 2 number.

```
#include <stdio.h>
void swap (int *c, int *b);
int main ()
{
   int a, b;
   printf("Enter any two integer :");
   scanf("%d %d", &a, &b);
   printf("Before swapping a=%d, b=%d",
                                  a, b);
   swap (&a, &b);
   printf ("\n After swapping a=%d, b=%d",
                                  a, b);
   return 0;
}
void swap (int* a, int* b)
{
   int* temp;
   temp = *a;
   *a = *b;
   *b = *temp;
}
```

Output.

Enter any 2 integers : 3 6
Before swapping : a = 3, b = 6
After swapping : a = 6, b = 3

Q.18. Write a function to find whether the string is palindrome or not.

```
#include <stdio.h>
void is palindrome (char* string)
{
  char* ptr, * rev;
  ptr = string;
  while (* ptr != '\0')
  {
    ++ ptr;
  }
  -- ptr;
  For (rev = string; ptr >= rev);
  {
    if (* ptr == * rev)
    {
      -- ptr;
      rev ++;
    }
    else break;
  }
  if (rev > ptr)
```

```
    printf(" string is palindrome.");
    else
    printf(" string is not palindrome.");
}
int main()
{
    char str[1000] = "madam";
    is palindrome(str);
    return 0;
}
```

Output.

String is palindrome.