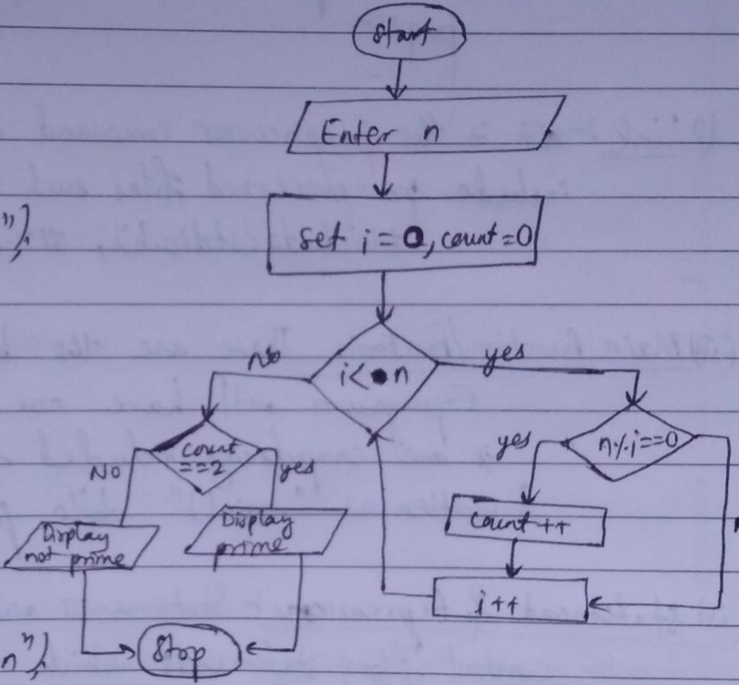
1. A process or set of rules to be followed in calculations or other problem solving operations by a computer is called algorithm.

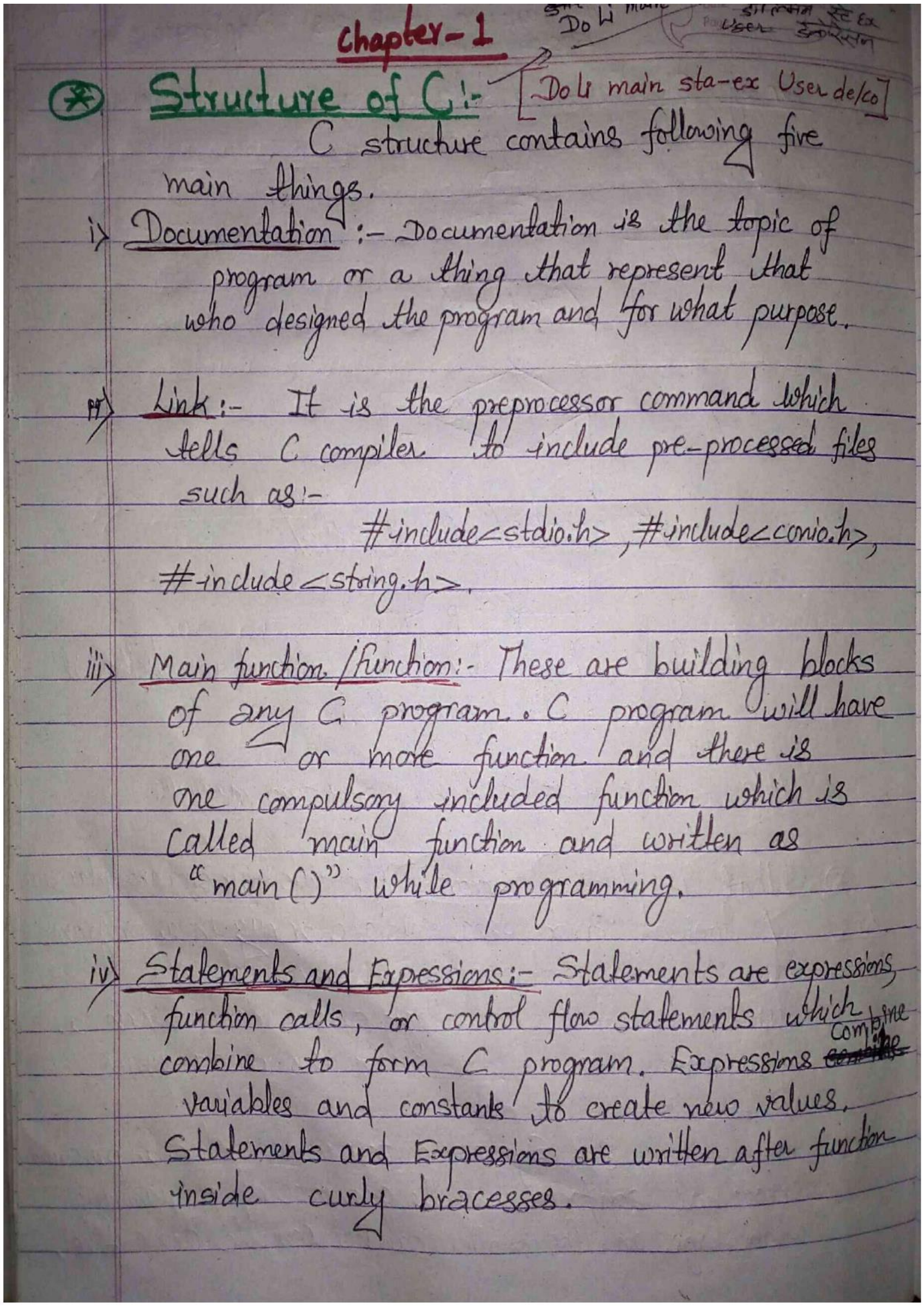
```
#include<stdio.h>
#include<stdlib.h>
#include <math.h>
#include <string.h>
int main(){
  int n, count = 0;
  printf("Enter number \n");
  scanf("%d",&n);
  for(int i=0; i<n; i++){
    if(n%i==0){
      count++;
    }
  }
   if(count==2){
   printf ("prime number \n");
    }
   else{
   printf("composite number \n");
    }
   return 0;
    }
```

Flowchart



Step 1: Start
Step 2: Enter number
Step 3: Initialize variable i=0 and count=0
Step 4: Repeat the step n%i until i<n.
Step 5: If n%i = 0 then add 1 to count.
Step 6: If count=2 then display prime
else not prime
Step 7: End.

2. Explain the structure of C-Program with example.



Do Li [illegible] Ex User [illegible]

Chapter-1

Structure of C :- [Do Li main sta-ex User de/co]

C structure contains following five main things.

i) Documentation :- Documentation is the topic of program or a thing that represent that who designed the program and for what purpose.

ii) Link :- It is the preprocessor command which tells C compiler to include pre-processed files such as:-

#include<stdio.h>, #include<conio.h>, #include<string.h>.

iii) Main function / function :- These are building blocks of any C program. C program will have one or more function and there is one compulsory included function which is called main function and written as "main ()" while programming.

iv) Statements and Expressions :- Statements are expressions, function calls, or control flow statements which combine to form C program. Expressions ~~combine~~ combine variables and constants to create new values. Statements and Expressions are written after function inside curly bracesses.

iv) **User declearation/comments:-** User creates some space for input data as varible name which is called user declearation. User declearation is done by writing as: input data_type variable;
Comments are used to give additional information inside a C program. For single line comment we use double slash and then we write comment but for multi-line comment we use ~~back~~slash asterik then we write comments and finally we use asterik with ~~back~~slash.

Note:- C is case sensative, semicolon at end of each statement, multiple statement can be on same line, white spaces are ignored and statements can continue over multiple lines.

3. Describe the four basic data types with example.

A C language programmer has to tell the system before-hand, the type of numbers or characters he is using in his program. These are data types. There are many data types in C language. A C programmer has to use appropriate data type as per his requirement.

Integer Type : Integers are whole numbers with a machine dependent range of values. A good programming language as to support the programmer by giving a control on a range of numbers and storage space. C has classes of integer storage namely short int, int and long int. All of these data types have signed and unsigned forms. A short int requires half the space than normal integer values. Unsigned numbers are always positive and consume all the bits for the magnitude of the number. The long and unsigned integers are used to declare a longer range of values.

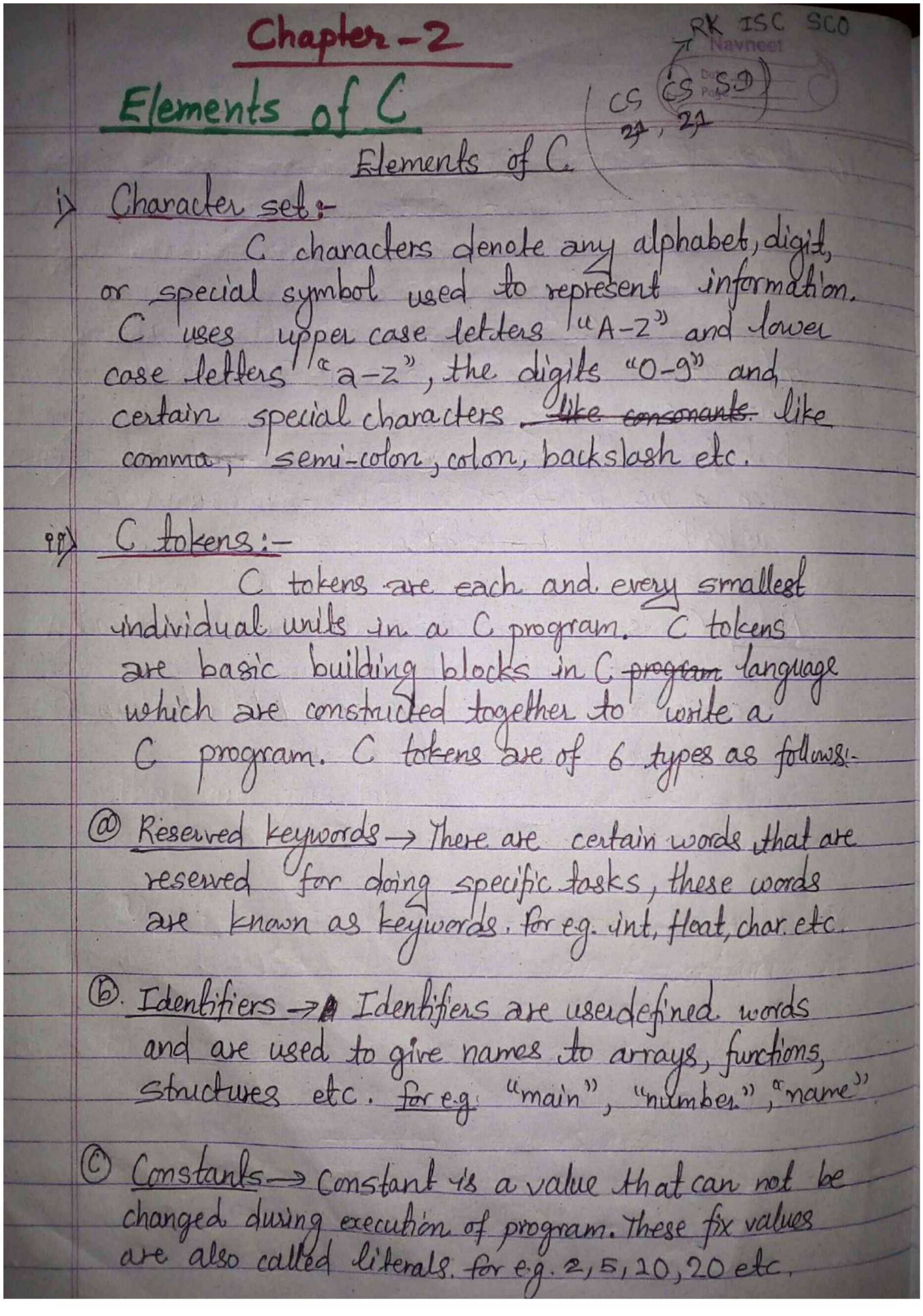
Floating Point Types : Floating point number represents a real number with 6 digits precision. Floating point numbers are denoted by the keyword float. When the accuracy of the floating point number is insufficient, we can use the double to define the number. The double is same as float but with longer precision. To extend the precision further we can use long double which consumes 80 bits of memory space.

Void Type : Using void data type, we can specify the type of a function. It is a good practice to avoid functions that does not return any values to the calling function.

Character Type : A single character can be defined as a defined as a character type of data. Characters are usually stored in 8 bits of internal storage. The qualifier signed or unsigned can be explicitly applied to char. While unsigned characters have values between 0 and 255, signed characters have values from –128 to 127.

Example: **Int sum;** **Int number, salary;** **Double average, mean;**

4. Elements of C



RK ISC SCO

Navneet

CS S.D

CS 21, 21

Chapter-2

Elements of C

Elements of C

i) Character set:-

C characters denote any alphabet, digit, or special symbol used to represent information. C uses upper case letters "A-Z" and lower case letters "a-z", the digits "0-9" and certain special characters, ~~like consonants.~~ like comma, semi-colon, colon, backslash etc.

ii) C tokens:-

C tokens are each and every smallest individual units in a C program. C tokens are basic building blocks in C ~~program~~ language which are constructed together to write a C program. C tokens are of 6 types as follows:-

(a) Reserved keywords → There are certain words that are reserved for doing specific tasks, these words are known as keywords. for e.g. int, float, char etc.

(b). Identifiers → Identifiers are userdefined words and are used to give names to arrays, functions, structures etc. for e.g. "main", "number", "name"

(c) Constants → Constant is a value that can not be changed during execution of program. These fix values are also called literals. for e.g. 2, 5, 10, 20 etc.

ⓓ. Strings → Sequence of characters is called string.
for e.g. "total", "mul", "Ram", "school" etc.

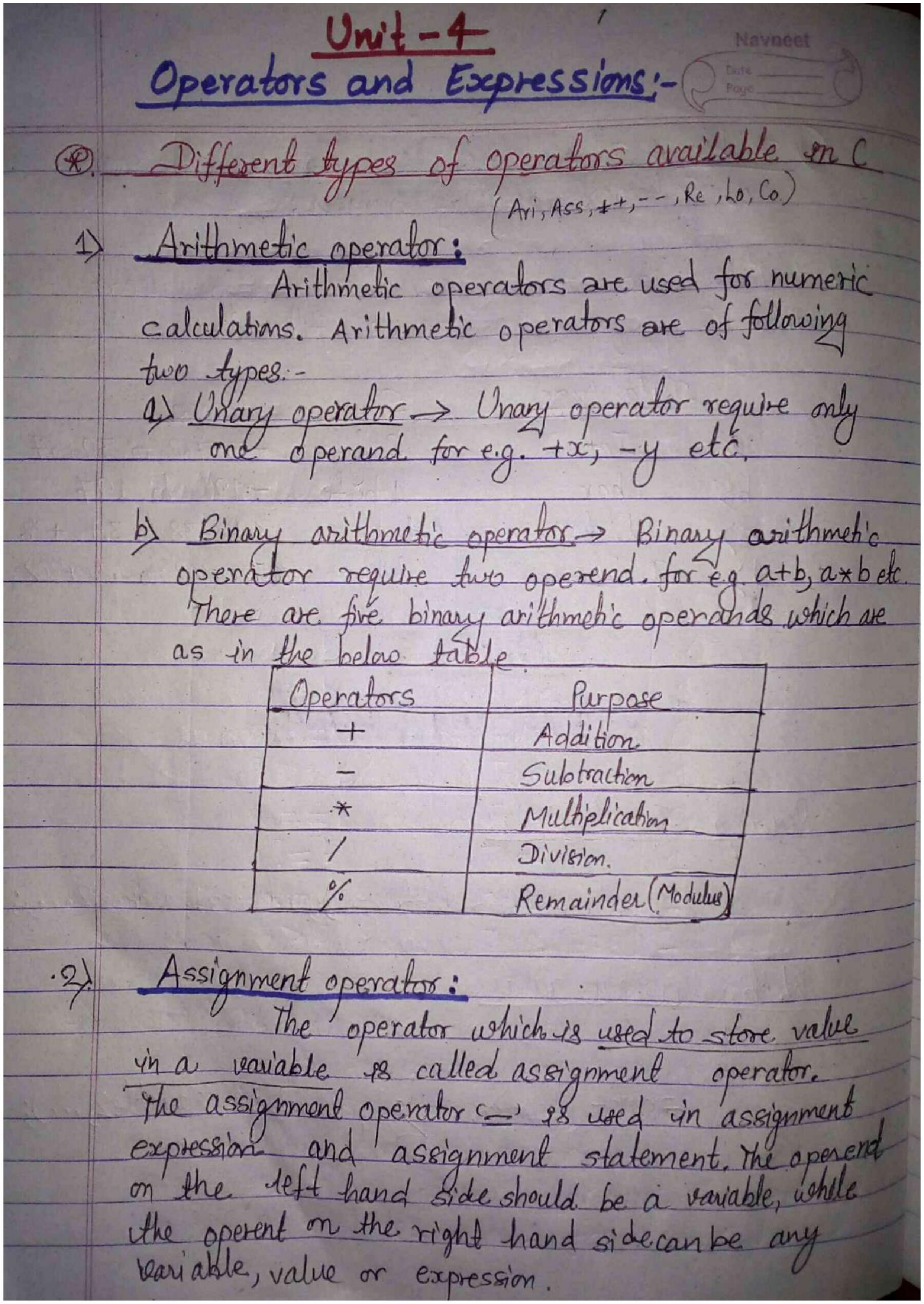
ⓔ. Special characters/symbols → Some special characters and symbols are used in C program like (), { }, &, !, = etc.

ⓕ Operators → Operators like +, /, −, *, % are used to perform differnt operations.

iii) Scape sequence :- Some characters such as new line, tab and back space can not be printed like other normal characters. C uses backslash (\) and some other characters to print those characters. These character combinations are known as scape sequence characters. for e.g. \b is used for backspace, \n is used for new line.

iv) Delimetres:- A delimeter is a unique character or ~~set~~ series of characters that indicate the begining or end of a specific statement, strings or function body. colon (:), semicolon (;), # etc are some delimeters.

4. What are the different types of operators available in C? Explain.



Unit-4

Operators and Expressions:-

(*) Different types of operators available in C

(Ari, Ass, ++, --, Re, Lo, Co)

1) Arithmetic operator:

Arithmetic operators are used for numeric calculations. Arithmetic operators are of following two types:-

a) Unary operator → Unary operator require only one operand for e.g. +x, -y etc.

b) Binary arithmetic operator → Binary arithmetic operator require two operend. for e.g. a+b, a*b etc. There are five binary arithmetic operands which are as in the below table.

Operators	Purpose
+	Addition
-	Subtraction
*	Multiplication
/	Division.
%	Remainder (Modulus)

2) Assignment operator:

The operator which is used to store value in a variable is called assignment operator. The assignment operator '=' is used in assignment expression and assignment statement. The operend on the left hand side should be a variable, while the operent on the right hand side can be any variable, value or expression.

3> **Increament / Decreament operator:**

C has two useful operators (++) and (--). These are unary operators. The increament operator (++) increments the value of variable by 1 and decreament operator (--) operator decrements the value of variable by 1.

for e.g. ++x is equivalent to $x = x+1$
--x is equivalent to $x = x-1$.

These operators are used only with the variables. These are of two types as follows:-

(a) Prefix increament / decreament :- Here, first the value of variable is increamented and decreamented then the new value is used in the operation.

(b). Post fix increament / decreament:- Here, first the value of variable is used in the operation and then increment / decreament is performed.

4> **Relational operator:-**

These operators are used to compare values of two expressions. An expression that contains relational operator is called relational expression. If the relation is true then the value of relational expression is 1 and if the relation is false, the value of expression is 0. The relational operators are as follows:-

Operator	Meaning
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to

Truth table:

Let us suppose two variables, a=9 and b=5

Expression	Relation	Value of expression.
a<b	False	0
a>b	True	1
a<=b	False	0
a>=b	True	1
a==b	False	0
a!=b	True	1
b!=0	True	1
a>8	True	1

5> Logical operator:-

An expression that combines two or more expressions is termed as a logical expression. For combing these expressions we use some operators which are called logical operators. These operators return 0 for false and 1 for true. C has three logical operators which are as follows:-

Operator	Meaning
&&	AND
||	OR
!	NOT

Here logical NOT is a unary operator while the other two are binary operators.

Types with truth table:

ⓐ AND (&&) operator → This operator gives the net result true only if both the conditions are true, otherwise the result is false.

Truth table:

Condition 1	Condition 2	Result
False	False	False
False	True	False
True	False	False
True	True	True

ⓑ OR (||) operator → This operator gives the net result false only if both the conditions are false, otherwise the result is true.

Truth table:

Condition 1	Condition 2	Result
False	False	False
False	True	True
True	False	True
True	True	True

Ⓒ Not (!) operator →
This is unary operator and neglects the value of the condition. If the value of condition is false then it gives the result true. If the value of the condition is true then it gives the result false.

Condition	Result
False	True
True	False.

6) Conditional operator :→
Conditional operator is a ternary operator which requires three expressions as operant, which is written as:
Test expression ? expression 1 : expression 2.
For eg. obtained marks >=35? Pass : Fail
First the test expression is evaluated, if the test expression is true, the expression 1 is evaluated and it becomes the value of overall conditional expression.
But if the test expression is false, then expression 2 is evaluated and it becomes the value of conditional expression.

5. Discuss logical operators along with the truth table.

6. Explain the relational operator and Increment/decrement operator with example.

7. Define variable. Explain the importance of variable of variable declaration. State the rules for the variable declaration.

A variable is a value that can change any time. It is a memory location used to store a data value. A variable name should be carefully chosen by the programmer so that its use is reflected in a useful way in the entire program. Variable names are case sensitive. Example of variable names are : Sun, number, Salary, Emp_name, average1

Importance: variables make it easier to store, read and change the data within the computer's memory by allowing you to associate easy-to-remember labels for the memory addresses that store your program's data. Rather than entering data directly into a program, a programmer can use variables to represent the data. Then, when the program is executed, the variables are replaced with real data.

Rules for the variable declaration:

1. They must always begin with a letter, although some systems permit underscore as the first character.
2. The length of a variable must not be more than 8 characters.
3. White space is not allowed.
4. A variable should not be a keyword.
5. It should not contain any special characters.
6. A variable declared should be of basic data types.

8.Explain preprocessor directive. Discuss #define directive with example.

Preprocessor directives are the special instructions which are executed before code passes through the compilation process. Every C programs are checked for the preprocessor statements before compiling and if any preprocessor statements are used in the program then they are processed first and then preprocessed program is handed over to the compiler for further compilation. Since every preprocessor statements are processed first and this is the reason for calling them preprocessor directives. It begins with hash (#) symbol and do not require semicolon at the end.

#define is commonly used preprocessor directive which is used for creating symbolic constants and for defining macros.

Syntax: **#define token value**

The types of preprocessor directive are as follows:

1. **Object-like Macros:** The object-like macro is an identifier that is replaced by value. It is widely used to represent numeric constants. For example:
   Example:

```
                #include <stdio.h>

                #define PI 3.1415

                main(){

                printf("%f",PI);

               }
```

2. **Function-like Macros:** The function-like macro looks like function call. For example:

```
        #include <stdio.h>
        #define MIN(a,b) ((a)<(b)?(a):(b))
        void main() {
        printf("Minimum between 10 and 20 is: %d\n", MIN(10,20));
         }
```

9. What is logical error?
A logic error (or logical error) is a mistake in a program's source code that results in incorrect or unexpected behavior. It is a type of runtime error that may simply produce the wrong output or may cause a program to crash while running. Many different types of programming mistakes can cause logic errors.

10. Define printf() function, header file and main function.
printf() is formatted output function which is used to display some information on standard output unit. It is defined in standard header file <stdio.h>. So, to use printf(), we must include header file <stdio.h> in our program.
Syntax: printf("String");

```
           or
         printf("format_string", var1, var2, var3, …, varN);
```

A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files.
Syntax: #include <file>

A main function is a predefined keyword or function in C. It is the first function of every C program that is responsible for starting the execution and termination of the program. It is a special function that always starts executing code from the 'main' having 'int' or 'void' as return data type. In other words, a main() function is an entry point of the programming code to start its execution.
Syntax:
```
main()
 {
// codes start from here
 }
```

11. What is type conversion? Discuss type casting with suitable example.
The type conversion process in C is basically converting one type of data type to other to perform some operation. The conversion is done only between those datatypes where the conversion is possible.

There are two types of typecasting.

a. Implicit Type casting – This conversion is done by the compiler. When more than one data types of variables are used in an expression, the compiler converts data types to avoid loss of data. Example:

```
#include <stdio.h>
int main() {
  int a = 10;
  char b = 'S';
  float c = 2.88;
  a = a+b;
  printf("Implicit conversion from character to integer : %d\n",a);
  c = c+a;
  printf("Implicit conversion from integer to float : %f\n",c);
  return 0;
}
```

Output:
```
Implicit conversion from character to integer : 93
Implicit conversion from integer to float : 95.879997
```

b.Explicit Type casting – This conversion is done by user. This is also known as typecasting. Data type is converted into another data type forcefully by the user. Example:

```
#include <stdio.h>
int main() {
float c = 5.55;
int s = (int)c+1;
printf("Explicit Conversion : %d\n",s);
return 0;
}
```

Output: Explicit Conversion : 6

12. What do you mean by statements? Explain the different types of statement with example.

A statement is a command given to the computer that instructs the computer to take a specific action, such as display to the screen, or collect input. A computer program is made up of a series of statements.

Expression Statements: It is combination of variables, Constants, operators, Function Calls and followed by a semicolon. Expression can be any operation like Arithmetic operation or Logical Operation. Few Examples for expression Statements:

```
X = Y + 10 ;
20 > 90;
a ? b : c ;
a = 10 + 20 * 30;
;   (This is NULL Statement ).
```

Compound Statement : Compound statement is combination of several expression statements. Compound Statement is enclosed within the Braces { }. Compound statement is also called as Block Statement. Example for Compound Statement:

```
{
    int a=10,b=20,c;
    c = a + b;
    printf(“value of C is : %d n”,c);
}
```

Selection Statements : Selection Statements are used in decisions making situations we will look about selections statements in Later Tutorials. Here are the few examples of Selection statements:
if
if…else
switch

Iterative Statements : These are also Called as Loops. If we want to Execute a part of program many times we will use loops. We will going to explain each and Every loop in Detail in Later Tutorials. Here is the List of Basic loops in C language:
for loop.
while loop.
do-while loop.

Jump Statements : These are Unconditional statements Jump statements are useful for Transfer the Control one part of program to other part of Program. There are few Jump Statements in C:
goto.
continue.
break.
return.

13. What are the four types of control statement use in C programming? Explain.
In C programs, statements are executed sequentially in the order in which they appear in the program, But sometimes we may want to use a condition for executing only a part of a program. Control statements enable us to specify the order in which the various instructions in the program are to be executed. C language supports four types of control statements, which are given below.
1. if……else
2. goto
3. switch
4. loop

If Statement: The if statement is used to express conditional expressions. The general form of if statement is;

<u>Syntax:</u>
```
if(condition){

statement1;

statement2;
```

```
        ……………………..
         statementN; }
```

If ……… else Statement: This is a bi-directional conditional control statement. This statement is used to test a condition and take one of the two possible actions. If the condition is true then a single statement or a block of statements is executed (one part of the program), otherwise another single statement or a block of statements is executed (other part of the program).

Syntax:
```
if(condition)
 {
 statement;
 …………………….;
}
else{
statement;
…………………;
}
```

goto: This is an unconditional control statement that transfer the flow of control to another part of the program .

Syntax:
```
goto label;
   ………..
   ………
 label:
statements;
…………………
 ……………
```

Switch: This is a multi-directional condition control statements. Sometimes there is a need in program to make choice among number of alternatives. For making this choice, we use the switch statements.

Syntax:

```
 switch (expression)
{
 case constant1:
 statement 1;
 break;
case constant2:
statement 2;
break;
default:
statement;
}
```

looping Repetition/looping means executing the same section of code more than once until the given condition is true. A section of code may either be executed a fixed number of times, or while condition is true. C provides three looping statements:

1. for loop
2. while loop
3. do while loop

14. What is looping statement? Explain different looping statements with suitable examples of each.

Repetition/looping means executing the same section of code more than once until the given condition is true. A section of code may either be executed a fixed number of times, or while condition is true. C provides three looping statements:

For loop: For statement makes programming more convenient to count iterations of a loop and works well where the number of iterations of the loop is known before the loop entered. The syntax is as follows:

```
for (initialization; test condition; loop update)

{

Statement(s);

}
```

While loop: A while loop statement in C programming language repeatedly executes a target statement as long as a given condition is true. The syntax of a while loop in C programming language is:

```
while(condition)

{

 statement(s);

}
```

do while: In C, do...while loop is very similar to while loop. Only difference between these two loops is that, in while loops, test expression is checked at first but, in d o...while loop code is executed at first then the condition is checked. So, the code are executed at least once in do...while loops. The syntax is:

```
do

{

 statement(s);

}while(condition);
```

15. What do you mean by entry-controlled and exit-controlled loop?

Entry Controlled Loop: Loop, where test condition is checked before entering the loop body, known as Entry Controlled Loop. Example: **while loop, for loop**

Exit Controlled Loop Loop, where test condition is checked after executing the loop body, known as Exit Controlled Loop. Example: **do while loop**

16. Define array. What are the benefits of using array?

Array is the collection of similar data types that can be represent by a single variable name. The individual data items in an array are called elements. A data items is stored in memory in one or more adjacent storage location dependent upon its type. The data types may be any valid data types like char, int or float.

Benefits of array:

- In an array, accessing an element is very easy by using the index number.
- The search process can be applied to an array easily.
- 2D Array is used to represent matrices.
- For any reason a user wishes to store multiple values of similar type then the Array can be used and utilized efficiently.

17. What is function? Discuss the benefits of using function.
A function is a self contained sub program that is meant to be some specific, well defined task. A C-program consists of one or more functions. If a program has only one function then it must be the main( ) function.

Advantages of using function:

1. Functions increases code reusability by avoiding rewriting of same code over and over.

2. If a program is divided into multiple functions, then each function can be independently developed. So program development will be easier.

3. Program development will be faster.

4. Program debugging will be easier.

5. Function reduces program complexity.

6. Easier to understand logic involved in the program.

7. Recursive call is possible through function.

18. Explain the functions and its types use in C programming with example.
C programs have two types of functions:

Library Functions: Library functions are supplied with every C compiler. The source code of the library functions is not given to the user. These functions are precompiled and the user gets only the object code. This object code is linked to the object code of your program by the linker. Different categories of library functions are grouped together in separate library files. When we call a library function in our program, the linker selects the code of that function from the library file and adds it to the program.

Some examples are:
printf(), scanf() are defined in header file stdio.h
getch(), clrscr() are defined in header file conio.h
strlen(), strupr() are defined in header file string.h
pow(), sqrt() are defined in header file math.h

User defined Functions: Users can create their own functions for performing any specific task of the program. These types of functions are called user defined functions. To create and use these functions, we should know about these three things

1. Function definition
2. Function declaration
3. Function call

```
#include<stdio.h>
#include<conio.h>
void sum(int,int);                    //function declaration
void main(){
int a,b;
clrscr();
printf("Enter two numbers\n");
scanf("%d%d",&a,&b);
sum(a,b);                              //function call
getch();
}
void sum(int x,int y){                 //function definition
int s;
s=x+y;
printf("Sum =%d",s);
}
```

19. Explain pass by value and pass by reference with suitable example of each.

Passing by value (call by value): In pass by value, values of variables are passed to the function from the calling function. This method copies the value of actual parameters into formal parameters. In other words, when the function is called, a separate copy of the variables is created in the memory and the value of original variables is given to these variables. So if any changes are made in the value (of the called function) is not reflected to the original variable (of calling function). It can return only one value.

```
#include<stdio.h>

#include<conio.h>

void change(int);

void main(){

int a=15;

clrscr();

printf("Before calling function, a=%d\n",a);

change(a);

printf("After calling function, a=%d",a);

getch();

}

void change(int x){

x=x+5;

}
```

<u>**Output:**</u>

```
Before calling function, a=15

After calling function, a=15
```

Passing by reference (call by reference): In passing by reference we pass the address or location of a variable to the function during function call. Pointers are used to call a function by reference. When a function is called by reference, then the formal argument becomes reference to the actual argument. This means that the called function doesn’t create its own copy of values rather, it refers to the original values only by reference name. Thus function works with original data and the changes are made in the original data itself. It can return more than one value at a time.

```
#include<stdio.h>

#include<conio.h>

void change(int*);

void main()

{

int a=15;

clrscr();

printf("Before calling function, a=%d\n",a);

change(&a);

printf("After calling function, a=%d",a);

getch();

}

void change(int *x)

{

*x=*x+5;

}
```

Output:

```
Before calling function, a=15

After calling function, a=20
```

20. Discuss any five string library functions.

strlen(): This function returns the length of the string. Example: strlen(name); This will return the length of the string stored in the variable name[].

strcat(): This function concatenates two strings. Example: strcat(name,name1); This will concatenate the strings stored in the variables name[] and name1[] in the order in which it is written.

strcpy(): This function copies the value of the second string to the first string. Example: strcpy(name1,name); This will copy the string in name[] to the variable name1[].

strcmp(): It compares two strings. Example: strcmp(name,name1); This compares the string in name[] with the string in name1[]. It returns 0 if the strings are same. It returns a value less than 0 if name[]<name1[]. Otherwise, it returns a value greater than 0.

strlwr(): It changes all the characters of the string to lower case. Example: strlwr(name); It shall convert the whole string stored in the variable name[] to lowercase.

21. Differences between:

a. Algorithm and Flowchart

	Algorithm	**Flowchart**
1.	Algorithm is step by step procedure to solve the problem.	Flowchart is a diagram created by different shapes to show the flow of data.
2.	Algorithm is complex to understand.	Flowchart is easy to understand.
3.	In algorithm plain text are used.	In flowchart, symbols/shapes are used.
4.	Algorithm is easy to debug.	Flowchart it is hard to debug.
5.	Algorithm is difficult to construct.	Flowchart is simple to construct.
6.	Algorithm does not follow any rules.	Flowchart follows rules to be constructed.

7.	Algorithm is the pseudo code for the program.	Flowchart is just graphical representation of that logic.

b. Increment and decrement operator

Increment Operators	**Decrement Operators**
Increment Operator adds 1 to the operand.	Decrement Operator subtracts 1 from the operand.
Postfix increment operator means the expression is evaluated first using the original value of the variable and then the variable is incremented(increased).	Postfix decrement operator means the expression is evaluated first using the original value of the variable and then the variable is decremented(decreased).
Prefix increment operator means the variable is incremented first and then the expression is evaluated using the new value of the variable.	Prefix decrement operator means the variable is decremented first and then the expression is evaluated using the new value of the variable.
++i, i++, etc.	--i, i--, etc.

c. if and switch statement

If else	**switch**
Which statement will be executed depend upon the output of the expression inside if statement.	Which statement will be executed is de by user.
if-else statement uses multiple statement for multiple choices.	switch statement uses single expression for multiple choices.
if-else statement test for equality as well as for logical expression.	switch statement test only for equality.
if statement evaluates integer, character, pointer or floating-point type or boolean type.	switch statement evaluates only character or integer value.
Either if statement will be executed or else statement is executed.	switch statement execute one case after another till a break statement is appeared or the end of switch statement is reached.

d. break and continue

break	**continue**
break statement is used to terminate the control from the switch case as well as in the loop.	continue statement is used to by-pass the execution of the further statements.
When the break statement encountered it terminates the execution of the entire loop or switch case.	When the continue statement is encountered it by-passes single pass of the loop.
It uses keyword break.	It uses keyword continue.
Syntax: break;	Syntax: continue;

#include<stdio.h>
#include<conio.h>
void main()
{
int i;
clrscr();
for(i=1;i<=10;i++)
{
if(i==5)
break;
printf("%d\n",i);
}
getch();
}
 	
output: 1 2 3 4
	#include<stdio.h>
	#include<conio.h>
	void main()
	{
	int i;
	clrscr();
	for(i=1;i<=10;i++)
	{
	if(i==5)
	continue;
	printf("%d\n",i);
	}
	getch();
	}
	
	Output: 1 2 3 4 6 7 8 9 10

e. while and do while

while	**do while**
It is entry control loop.	It is an exit control loop.
Test condition is evaluated before the loop is executed	Test condition is evaluated after the loop is executed.
It use keyword while.	It uses keywords do and while.
Loop is not terminated with semicolon.	Loop is terminated with semicolon.
It doesn’t execute the body of loop until and unless the condition is true.	Body of the loop is always executed at least once since the test condition is not checked until end of the loop

#include<stdio.h>
void main()
{
int i;
i=1;
	#include<stdio.h>
	void main()
	{
	int i;
	i=1;

while(i<=5)
{
printf("%d\n",i);
}
	do
	{
	printf("%d\n",i);
	} while(i<=5);

f. call by value and call by reference

call by value	**call by reference**
A copy of the variable is passed.	A variable itself is passed.
Change in a copy of variable doesn't modify the original value of variable out side the function.	Change in the variable affect the value of variable outside the function also.
function_name (variable_name1, variable_name2, ...);	function_name (&variable_name1, &variable_name2, ...);
Primitive type are passed using "call by value".	Objects are implicitly passed using "call by reference".
program	**program**

22. What is dynamic memory allocation? Discuss the use of malloc() in dynamic memory allocation.

Dynamic Memory Allocation is manual allocation and freeing of memory according to your programming needs. Dynamic memory is managed and served with pointers that point to the newly allocated memory space in an area which we call the heap.

malloc() function in C: The C malloc() function stands for memory allocation. It is a function which is used to allocate a block of memory dynamically. It reserves memory space of specified size and returns the null pointer pointing to the memory location. The pointer returned is usually of type void. It means that we can assign C malloc() function to any pointer.

Syntax of malloc() Function: ptr = (cast_type *) malloc (byte_size);

Here, ptr is a pointer of cast_type. And malloc() function returns a pointer to the allocated memory of byte_size.

Example of malloc(): ptr = (int *) malloc (50)

23. Define recursion.

Recursion is a powerful technique of writing a complicated algorithm in an easy way. According to this technique a problem is defined in terms of itself. The problem is solved by dividing it into smaller problems, which are similar in nature to the original problem. These smaller problems are solved and their solutions are applied to get the final solution of our original problem.

A function will be recursive, if it contain following features:

i. Function should call itself.
ii. Function should have a stopping condition (base criteria) and every time the function calls itself it must be closer to base criteria.

24. What is structure? How is it different from array and Union? Discuss.
A structure is a collection of logically related data items grouped together under a single name. In structure the individual elements may differ in type, that’s why we can regard structure as a heterogeneous user-defined data type. The data items enclosed within a structure are known as members.

Syntax for structure definition is:
```
struct struct_name
{
data_type mem1;
data_type mem2;
…………………
data_type memn;
};
```

Array	Structure
1. Array is a built-in data type.	1. Structure is a derived data type.
2. Array holds the group of same elements under a single name.	2. Structure holds the group of different elements under a single name.
3. We cannot have array of array.	3. We can have array of structure.
4. Memory occupied by an array is the multiple of no of index	4. Memory occupied by structure is sum of individual data type.
5. Cannot take part in complex data structure.	5. Can take part in complex data structure.
6. Cannot be used in program to interact with hardware.	6. Can be used in program to interact with hardware.

7. Syntax for declaration
`data_type  arrayname[subscript];`
	7. Syntax for declaration
	`struct struct_name`
	`{`
	`   data_type mem1;`
	`   data_type mem2;`
	`   ....................`
	`   data_type memn;`
	`};`

Structure	Union
1. Memory occupied by structure is sum of individual data type.	1. Memory occupied by union is of highest data type of all.
2. Can take part in complex data structure.	2. Cannot take part in complex data structure.
3. Keyword struct is used.	3. Keyword union is used.
4. Every element value's are independent to each other.	4. If any of the values of any element has been changed there is direct impact to the other elements values.
5. Memory allocation of every element is independent to each other thereby the memory allocation is sum of every element.	5. Memory allocation is performed by sharing the memory with highest data type.
6. All members can be accessed simultaneously.	6. Only one member is active at a time, so only one member can be accessed at a time.

7. Syntax
`struct struct_name`
`{`
`   data type mem1;`
`   data_type mem2;`
`   ....................`
`   data_type memn;`
`};`
	7. Syntax
	`union union_name`
	`{`
	`   data type mem1;`
	`   data_type mem2;`
	`   ....................`
	`   data_type memn;`
	`};`

25. Define pointer. Explain the use of pointer in C-programming. What is the function of a pointer variable? Explain the declaring and initializing pointers with example.

A pointer is a variable that stores the address of another variable. Memory can be visualized as an ordered sequence of consecutively numbered storage locations. A data item stored in memory in one or more adjacent storage locations depends upon its type.

Some uses of pointers are:

1. Accessing array elements.
2. Returning more than one value from a function.
3. Accessing dynamically allocated memory.
4. Implementing data structures like linked lists, trees, and graphs.

<u>Declaration of pointer:</u> A pointer variable is declared with data type that of the variable it has to point. Suppose if a pointer has to point to a integer variable than it is declared using pointer data type. A pointer declaration is different from other variable as it uses a character '*' in its declaration.

data type * pointer name;

Example: int *ptr;   here ptr is an integer type pointer.

<u>Assigning value to pointer:</u> As pointer variable only stores address of another variable, while assigning its value we use address resolution operator "&" in front of the variable the pointer is suppose to point.

```
Example:     void main(){
              int x=25;
              int *ptr;
              ptr=&x;
              printf("%d %x", x, ptr);
              getch();
               }
```

This Program will give following output: 25, fff0

Here 25 is the value of x and fff0 is the value of ptr, which in turn is the address of x.

26. Discuss the relationship between pointer and one-dimensional array.

The elements of an array are stored in contiguous memory locations. Suppose we have an array arr[5] of type int.

int arr[4]={6,9,12,4};

arr[0]	arr[1]	arr[2]	arr[3]
6	9	12	4
5000	5002	5004	5006

Here 5000 is the address of first element, and since each element (type int) takes 2 bytes so address of next element is 5002, and so on. The address of first element of the array is also known as the base address of the array.

Program to print the value of array element using pointer.

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
int arr[100],i,n;
clrscr();
printf("How many elements are there: ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\n Enter element: ");
scanf("%d",(arr+i));
}
printf("\n Printing array element:");
for(i=0;i<n;i++)
{
printf("\n %d",*(arr+i));
}
getch();
 }
```

28. Explain the passing structure to function with example.
A structure can be passed to any function from main function or from any sub function. We can pass whole structure as an argument to a function.

Example:

```
#include<stdio.h>
#include<conio.h>
void display(struct customer);
struct customer{
  int id;
char name[15];
char address[15];
};
void main(){
 struct customer cu={1,"Binod","Bharatpur"};
clrscr();
display(cu);
getch();
    }
void display(struct customer c){
printf("\n ID :%d",c.id);
printf("\n Name :%s",c.name);
printf("\n Address :%s",c.address);
    }
```

29. Why do we need data files? Discuss different file openings modes.

Data file is a data structure which helps us to store our data permanently on the secondary storage devices (e.g. hard drive, floppy disks etc).

Data files allow us to store information permanently and to access later on and alter that information whenever necessary. So we need data files.

The different file modes that can be used in opening file are:

a. **"w" (write):** If the file doesn‟t exist then this mode creates a new file for writing, and if the file already exists then the previous data is erased and the new data entered is written to the file.

b. **"a" (append):** If the file doesn‟t exist then this mode creates a new file and if the file already exists then the new data entered is appended at the end of existing data. In this mode, the data existing in the file is not erased as in "w" mode.

c. **"r" (read):** This mode is used for opening an existing file for reading purpose only. The file to be opened must exist and the previous data of the file is not erased.

d. **"w+" (write + read):** This mode is same as "w" mode but in this mode we can also read and modify the data. If the file doesn‟t exist then a new file is created and if the file exists then previous data is erased.

e. **"r+" (read + write):** This mode is same as "r" mode but in this mode we can also write and modify existing data. The file to be opened must exist and the previous data of file is not erased. Since we can add new data and modify existing data so this mode is also called updata mode.

f. **"a+" (append + read):** This mode is same as the "a" mode but in this mode we can also read the data stored in the file. If the file doesn‟t exist, a new file is created and if the file already exists then new data is appended at the end of existing data. We cannot modify existing data in this mode.

30. Short Notes:

Qn.A) **<u>Formatted I/O:</u>** C language provide us console input/output functions. As the name says, the console input/output functions allow us to -

- Read the input from the keyboard by the user accessing the console.
- Display the output to the user at the console.

Formatted console input/output functions are used to take one or more inputs from the user at console and it also allows us to display one or multiple values in the output to the user at the console.

Some of the most important formatted console input/output functions are -

scanf(): This function is used to read one or multiple inputs from the user at the console.

printf(): This function is used to display *one or multiple* values in the output to the user at the console.

sscanf(): This function is used to read the characters from a string and stores them in variables.

sprintf(): This function is used to read the values stored in different variables and store these values in a character array.

Qn.B) Documentation: It refers to the details that describe a program. It is final report of the whole program development phase. It consists of detail about what activity has been performed and problems faced in each phase. Generally, it is prepared for the future references.

Qn.C) Structure: A structure is a collection of logically related data items grouped together under a single name. In structure the individual elements may differ in type, that's why we can regard structure as a heterogeneous user-defined data type. The data items enclosed within a structure are known as members. Syntax for structure definition is:

```
 struct struct_name{
 data_type mem1;
data_type mem2;
   …………………
data_type memn;
};
```

Qn.D) **Benefits of Data Files:**

- **Reusability:** It helps in preserving the data or information generated after running the program.
- **Large storage capacity:** Using files, you need not worry about the problem of storing data in bulk.
- **Saves time:** There are certain programs that require a lot of input from the user. You can easily access any part of the code with the help of certain commands.
- **Portability:** You can easily transfer the contents of a file from one computer system to another without having to worry about the loss of data.