Unit 1 : (2 Marks)

Q. 1) What is identifier? Write rules for writing identifier name.

Ans :- It is a name given to variable, function, array, structure, union etc.

Rules for writing an identifier

- A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
- The first letter of an identifier should be either a letter or an underscore. ...
- There is no rule on length of an identifier.

Q.2) What is keyword ? list any four keyword.

Ans :- It is a reserved keywords whose meaning already define in C compiler user cannot change them. C support 32 keywords.

1. auto 2. break 3. char 4. const

Q. 3) What is constant?

Ans :- It is a fixed entity which does not change during program execution.

For Example : p =3.14;

Q. 4) How variables are declared in c? Give an example.

Ans:- It is a name given to storage area and its value change during program execution. It is used to store data, and its value change at any time.

Syntax :- Data type <variable_name>;

Example : int a; float b;

Q. 5) Write down the syntax and example of scanf() function.

Ans :- Syntax : scanf("%d",&a);

Example : scanf("%d",&a);

Q. 6) Write down the syntax and example of printf() function.

Ans :- Syntax : printf("Enter any Two no.");
printf("%d",a);

Example : printf("Enter any Two no.");
printf("%d",a);

Q. 7)List relational operator in C?

Ans :- <,<=,>,>= these are the relational operator in c. Mostly it is used for performing a logical operation.

Q. 8) What is variable?

Ans :- It is a name given to storage area and its value change during program execution. It is used to store data, and its value change at any time.

Example : int a,b;

Q. 9) What is meant by compound statement?

Ans :- A **compound statement** (also called a "block") typically appears as the body of another **statement**, such as the if **statement**. Declarations and Types describes the form and **meaning** of the declarations that can appear at the head of a **compound statement**.

Q. 10) Differentiate '=' and '==' operator in c?

Ans :- '=' these operator is known as assignment operator it is used for storing a value L.H.S to R.H.S and '==' operator is used for comparision.

Q. 11) What is increment and decrement operator in c?

Ans:-There are two types of operator in C. '++' it is increment operator and '--' it is decrement operator.

Example : a++, a--; ++a,--a;

Q. 12)Differentiate printf() and puts().

Ans:- printf() function is used for printing a value like integer ,float and character and puts() function is used for printing a string like any name.

Q. 13) What do you mean by type conversion?

Ans:- Conversion of one datatype to another datatype is known as type casting , C support two types of typecasting .

1> Implicit typecasting 2> Explicit typecasting

Q. 14)What is the purpose of break statement?

Ans:- In C programming, break is used in terminating the loop immediately after it is encountered. the break statement is used with conditional if statement.

Q.15) Write down the format specifier and size for int, float and char data type?

Ans :-

Data type	Size	Symbol
Char	1 Byte	%c
Int	2 Byte	% d or % i
Long	4 Byte	%ld
Unsigned int	2 Byte	%u
Float	4 Byte	%f
Double	8 Byte	%lf

Q.16) What is sizeof() operator?

Ans) Sizeof operator returns the size of any variable and datatype.

e.g. int a; sizeof(int), sizeof(a); o/p= 2 2.

Q.17) What is Symbolic Constant?

Ans :- **symbolic constant** is name that substitute for a sequence of character that cannot be changed. The character may represent a numeric **constant**, a character **constant**, or a string. When the program is compiled, each occurrence of a **symbolic constant** is replaced by its corresponding character sequence.

Q. 18) What is loop?

Ans:- It is used to execute group of statement repeatedly till the condition is satisfy. C support three types of looping statement. 1. while loop 2. do while loop 3. for loop.

Q. 19) Differentiate pre increment and post increment operator?

Ans :- ref. Q. No. 11

Q. 20) What is the purpose '\n' , '\t' , ' \b' , and '\a' in C?

Ans:- '\n' is used for jump on next line.

'\t' is used for keeping one tab space.

'\b' is used for backspace.

'\a' is used for alarm.

Q.21) Describe the purpose of any four conversion characters commonly used in printf()?

Ans : - In the C language, if you want to print the value of a variable to the screen, you'll likely use the printf() function. To do so, you need to include a conversion character — a placeholder of sorts — in the literal string that you want to print. That conversion character is then replaced by the variable or value you indicate later in the command.

Q. 22) List the decision making statement in C?

Ans :- There are four types of decision making statement they are following.

1. Simple if 2. If_else 3. Nested if_ else 4. Else if ladder

Q.23) What is meant by unconditional jump? Give an example?

Ans :Definition of: unconditional branch. unconditional branch. In programming, a GOTO, BRANCH or JUMP instruction that passes control to a different part of the program. Contrast with conditional branch.

for example,

```
; Handle one case
label1:
    .
    jmp done

; Handle second case
label2:
    .
    jmp done
    .
done:
```

Q.24) What is unary operator in C?

Ans :- Those operator which require only one operand is known as unary operator. ++ and -- these two are unary operator.

Q. 25) What is modulo operator? How does it work in C?

Ans :- In computing, the **modulo** operation finds the remainder after division of one number by another (sometimes called **modulus**). Given two positive numbers, a (the dividend) and n (the divisor), a **modulo** n (abbreviated as a **mod** n) is the remainder of the Euclidean division of a by n.

The **modulus operator** does all that calculation for you, 13 % 5 = 3. Very simple: a % b is defined as the remainder of the division of a by b . **Modulus** division gives you the remainder of a division, rather than the quotient. ... **Modulus** division is simple.

Unit 1 (3 Marks)

Q.1) Write down the features of C Language.

Ans : 1) it is a robust language with rich set of built-in functions and operators that can be used to write any complex program.

2) The C compiler combines the capabilities of an assembly language with features of a high-level language.

3) Programs Written in C are efficient and fast. This is due to its variety of data type and powerful operators.

4) It is many time faster than BASIC.

5) C is highly portable this means that programs once written can be run on another machines with little or no modification.

6) Another important feature of C program, is its ability to extend itself.

7) A C program is basically a collection of functions that are supported by C library. We can also create our own function and add it to C library.

8) C language is the most widely used language in operating systems and embedded system development today.

Q.2) Explain the structure of C Program.

=> Any C program is consists of 6 main sections. Below you will find brief explanation of each of them.

Basic Structure of C Programs
Documentation Section
Link Section
Definition Section
Global Declaration Section

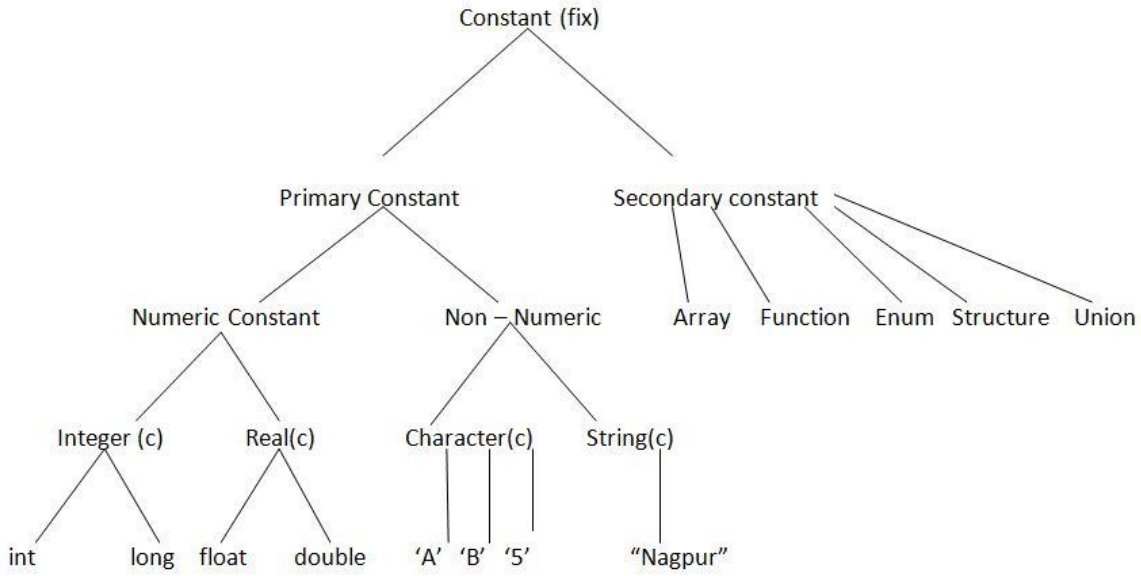
main() Function Section
{
Declaration Part
Executable Part
}

Subprogram Section Function 1 Function 2 Function 3 - - - Function n

Q.3) What is constant ? Explain different types of constant in C.

Ans : It is a fixed entity which does not change during program execution.

Classification of C constant



Q.4) what is the difference between # define and constant in c?

Ans: **#define** is a preprocessor directive. Things defined by #define are replaced by the pre-processor before compilation begins.

const variables are actual variables like other normal variable.

The big advantage of const over #define is type checking. We can also have pointers to constant variables, we can pass them around, typecast them and any other thing that can be done with a normal variable. One disadvantage that one could think of is extra space for variable which is immaterial due to optimizations done by compilers.

In general const is a better option if we have a choice. There are situations when #define cannot be replaced by const. For example, #define can take parameters (See this for example). #define can also be used to replace some text in a program with another text.

Q.5) what are the different types of statements used in c ?

Ans: There are two types of control statement branching statement and looping statement.

1. Branching statement
    a) if statement
    b) Switch case
    C) go to

It is used to execute group of statement repeatedly till the condition is satisfy.

2. looping statement.
    a) while loop
    b) do while loop
    c) for loop.

Q.6) what is variable ? write down rules to define variable name .

Ans: It is a name given to storage area and its value change during program execution.

It is used to store data , and its value change at any time.

Rules for defining variable

1. Variable name must start with alphabet and underscore.
2. Space are not allowed.
3. Keywords are not allowed.
4. Special symbol are not allowed.
5. Digits are allowed but first character must be alphabet or underscore.
6. Variable length should be 1 to 8 character long.

Q.7)what is conditional operator in c ? give is syntax and example.

Ans: Conditional operators return one value if condition is true and returns another value is condition is false. This operator is also called as ternary operator.

Syntax : (Condition? true_value: false_value);

Example : (A > 100 ? 0 : 1);

Q.8) What is mixed mode mathematical operation in c.state with suitable example.

Ans: In a statement or expression if one the operand is real (float) and another one is integer then expression is called as Mixed Mode Arithmetic Expression. If in an expression either operand is of real then output is always in real format. If both operands are real then output will be in real formats.

Example

Real operand and Integer operand = Real operand (Output)

Real operand and Real operand = Real operand (Output).

Types of Arithmetic Operators used in Mixed Mode Expression

"*" - Multiply use for Multiplication
"/" - Divide use for division
"+" - Plus use for addition
"-" - Minus use for subtraction
"%" - Modulus use for getting a reminder
"**" - Square use for getting square number

Rules for Evaluation Mixed Mode Arithmetic Expression

Rule 1

Evaluate Expressions always from Left to Right
For Example: 3 + 5 - 4 = 4

Rule 2

Priority of an operator is also considered while calculating an expression

Q.9) what is ternary operator in c?explain with in an example.

Ans: **ternary operator** that is part of the syntax for basic conditional expressions in several programming languages. It is commonly referred to as the conditional **operator**, inline if (iif), or **ternary**

Example :

```
#include <stdio.h>
main()
{
  int a , b;
  a = 10;
  printf( "Value of b is %d\n", (a == 1) ? 20: 30 );
  printf( "Value of b is %d\n", (a == 10) ? 20: 30 );
}
```

Q.10) what is escape sequence character constant ?

Ans: **Escape Sequences**. A backslash **character** (\ ) is used to introduce an **escape sequence**, which allows a visual representation of certain nongraphic **characters**. One of the most common **escape constants** is the newline **character** (\n).

Q.11) What is Assignment Operator?

Ans : Ans: An **assignment operator** is the **operator** used to **assign** a new value to a variable, property, event or indexer element in C# programming language. **Assignment operators** can also be used for logical operations such as bitwise logical operations or operations on integral operands and Boolean operands.

Q. 12) Explain if… else statement with its syntax ?

Ans : An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false.

Syntax :

```
if(boolean_expression)
        {
                True block statement;
        }
        else
        {
                Else Block Statement;
        }
```

Q.13) explain nested if statement with suitable example.

Ans: It is always legal in C programming to **nest** if-else statements, which means you can use one if or else if statement inside another if or else if statement(s).

Syntax

```
if( boolean_expression 1)
{
  /* Executes when the boolean expression 1 is true */
if(boolean_expression 2)
{
    /* Executes when the boolean expression 2 is true */
  }
}
```

Q.14)what is logical operator? State its necessity.

Ans: A **logical operation** is a special symbol or word which connects two or more phrases of information. It is most often used to test whether a certain relationship between the phrases is true or false.

Q.15) differentiate getch() and getchar().

Ans: **getchar()** : The getchar() function is a C standard library function (declared in *stdio.h*) that gets a character (an unsigned char) from stdin.

getch() : The getch() function is not a part of the C standard library. It is defined in *conio.h* header file. It is mostly used by MS-DOS compilers like: Turbo C. But, if you want to use it in some other compiler (not all), you need to download the *conio.h* file and include that in your code externally.

Q.16) what is scanf()function and how does it differ from the getchar () function?

Ans:

Q.17)differentiate entry control loop and exit control loop.

Ans :- 1. Entry Controlled Loop

2. 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

Q.18) Write program to find factorial of an inputted number?

Ans :- #include<stdio.h>

```
main()
{
int n,I,fact=1;
clrscr();
printf(“Enter Any Number”);
scanf(“%d”,&n);
for(i=n;i>=1;i++);
{
fact =fact *i;
}
printf(“\n factorial =\t%d”,fact);
getch();
}
```

Q.19) write a program in c to swap value of two variable .

Ans:

```
#include<iostream>
using namespace std;
main()
{
        int a,b,temp;
        cout<<"\nEnter two numbers : ";
        cin>>a>>b;
        temp=a;
        a=b;
        b=temp;
        cout<<"\nAfter swapping numbers are : ";
        cout<<a<<" "<<b;
}
```

Q.20)what is meant by sequence, selection and iteration process?

Ans:**Sequencing:** This means that the computer will run your code in order, one line at a time from the top to the bottom of your program. It will start at line 1, then execute line 2 then line 3 and so on till it reaches the last line of your program.

Selection: Sometimes you only want some lines of code to be run only if a condition is met, otherwise you want the computer to ignore these lines and jump over them. This is achieved using IF statements. e.g. If a condition is met then lines 4, 5, 6 are executed otherwise the computer jumps to line 7 without even looking at line 4,5 and 6.

Iteration: Sometimes you want the computer to execute the same lines of code several times. This is done using a loop. There are three types of loops: For loops, while loops and repeat until loops. That’s handy as it enables you not to have to copy the same lines of code many times.

Q.21) write down the compilation process of c program.

- Ans: Preprocessing - Using a Preprocessor program to convert C source code in expanded source code. "#includes" and "#defines" statements will be processed and replaced actually source codes in this step.

- Compilation - Using a Compiler program to convert C expanded source to assembly source code.
- Assembly - Using a Assembler program to convert assembly source code to object code.
- Linking - Using a Linker program to convert object code to executable code. Multiple units of object codes are linked to together in this step.
- Loading - Using a Loader program to load the executable code into CPU for execution.

Q.22) What printf()function and compare with putchar()function?

Ans:

Q.23) What is getch()function? Give an example.

Ans: Reads characters from the standard input (stdin) and stores them as a C string into str until a newline character or the end-of-file is reached.

Syntax for Accepting String :

```
char * gets ( char * str );
```

OR

gets(<variable-name>)

Live Example :

```
#include<stdio.h>
void main()
{
char name[20];
printf("\nEnter the Name : ");
gets(name);
}
```

Q.24)what is puts() function? Give an example.

Ans: puts() function is a file handling function in C programming language which is used to write a line to the output screen. Please find below the description and syntax for above file handling function.

Example :

```
#include<stdio.h>
#include<string.h>
Int main()
{
Char string[40];
strcpy(str, “This is a text string”);
puts(string);
return 0;
}
```

Q. 25) Explain three parts of for loop.

Ans :-Another form was popularized by the **C** programming language. It requires 3 **parts**: the initialization, the condition, and the afterthought and all these **three parts** are optional. for (INITIALIZATION; CONDITION; AFTERTHOUGHT) { // Code for the for-**loop's** body goes here. }

UNIT 1 (5 Marks)

Q. 1) Explain Keywords & Identifiers in Details.

Ans : Refer Q. No. 1 & 2 (2 Marks)

Q.2) Explain Build-in data type in details.

Ans : Primary Data types is also called as build in data type. Every c compiler supports five primary data type.

Void : As the name suggests it holds no value and is generally used for specifying the type of function or what it returns. If the function has void type, it means that the function will not return any value.

int : Used to denote an integer type.

Char : Used to denote a character type.

float, double : Used to denote a floating point type.

int*, float*,char* : Used to denote a pointer type.

Three more data type has been added in C99.

- _Bool
- _Complex
- _Imaginary

Example:

```
int    age;
char   letter;
float  height, width
```

Q.3) Explain Derived data type in details.

Ans :- C Supports three derived data types :

1) Arrays : Arrays are sequences of data items having homogeneous value. They have adjacent memory locations to store values.
2) Reference : Functions pointers allow referencing functions with a particular signature
3) Pointers : These are powerful C features which are used to access the memory and deal with their addresses.

Q.4) What is operator? Explain about operator precedence of c?

Ans : C operators are symbols that is used to perform mathematical or logical manipulations. C programming language is rich with built-in operators. Operators take part in a program for manipulating data and variables and form a part of the mathematical or logical expressions.

Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has a higher precedence than the addition operator.

For example, x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has a higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Operator	Associativity
1. Assignment operator		= += -= *= /= %=>>= <<= &= ^= |=	
2 .Arithmetic operator		+, -,/ ,%	
3. Relational operator		< <= > >=	
4. Conditional operator		?	
5. logical operator		&& , ||	
6. Increment/Decrement operator		++ - -	
7. Bitwise operator		^,|,&,	
8 .Special operator		(,) (size of)	
Postfix	() [] -> . ++ - -		Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof		Right to left
Multiplicative	* / %		Left to right
Additive	+ -		Left to right
Shift	<<>>		Left to right
Relational	<<= >>=		Left to right
Equality	== !=		Left to right
Bitwise AND	&		Left to right
Bitwise XOR	^		Left to right
Bitwise OR	|		Left to right
Logical AND	&&		Left to right
Logical OR	||		Left to right
Conditional	?:		Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=		Right to left
Comma	,		Left to right

Q. 5) What is Bitwise Operator? Explain different type of bitwise operator with suitable example.

Ans : C provide special operator for bit operation between two variables.

Type of Bitwise operator

Operator	Description
<<	Binary Left Shift Operator
>>	Binary Right Shift Operator
~	Binary ones complement Operator
&	Binary AND Operator
^	Binary XOR Operator
|	Binary OR Operator.

Example : This C program is used to swapping two numbers, using bitwise operators.

```
#include <stdio.h>
int main() {
   int i = 65;
   int k = 120;
   printf(" value of i=%d k=%d before swapping", i, k);
   i = i ^ k;
```

```
   k = i ^ k;
   i = i ^ k;
   printf("value of i=%d k=%d after swapping", i, k);

   return 0;
}
```

Q. 6) Explain Break and Continue Statement in C with Suitable example.

Ans : Reference Q. 14 (2 Marks)

Example :

```
#include<stdio.h>
main()
{
   int a;
   printf("Please enter a no between 1 and 5: ");
   scanf("%d",&a);

   switch(a)
   {
   case 1:
   printf("You chose One");
   break;
  case 2:
   printf("You chose Two");
   break;
   case 3:
   printf("You chose Three");
   break;
   case 4:
   printf("You chose Four");
   break;
   case 5:
   printf("You chose Five.");
   break;
   default :
   printf("Invalid Choice. Enter a no between 1 and 5");
   break;
   }
}
```

Q. 7) What is While loop? Differentiate while loop and do-while loop with suitable example.

Ans :- When we don't know the ending point of any execution at that time we used while loop .it is used execute the group of statement repeatedly till condition is satisfy.

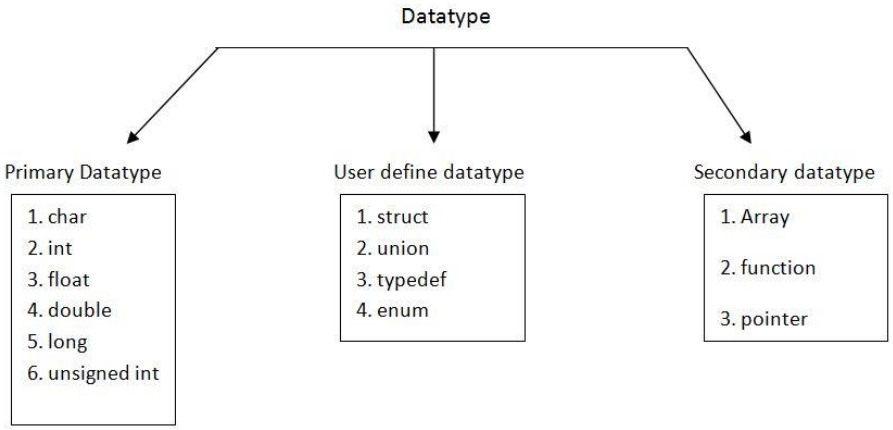
Basis for comparison	**While**	**do-while**

General Form	while (condition) {
	statements; //body of loop
	}
		do{
		.
		statements; // body of loop.
		.

Basis for comparison	While	do-while
		} while(Condition);
Controlling Condition	In 'while' loop the controlling condition appears at the start of the loop.	In 'do-while' loop the controlling condition appears at the end of the loop.
Iterations	The iterations do not occur if, the condition at the first iteration, appears false.	The iteration occurs at least once even if the condition is false at the first iteration.

Q. 8) Discuss primary data types with their size and format specifier?

=>Datatypes tells the which types of value hold by variable like as integer , float, char. C support three types of datatype 1.Primery datatype 2. User define datatype 3. Derived datatype.



Detail concept of dataypes their size, control string, range given below

Datatype	Size	Control string	Range
char	1 Byte	%c	-128 to 127
int	2 Byte	%d or %i	-32768 to 32767
long	4 Byte	%ld	-2147483648 to 214748
unsigned int	2 Byte	%u	0 to 65535
float	4 Byte	%f	3.4e-38 to 3.4e 38
double	8 Byte	%lf	1.7e-308 to 1.7e308

Q. 9) Explain scanf() and printf() functions with suitable example.

Ans : Ref. Q. No. 5 & 6 (2 Marks Each)

Example :

```
#include<stdio.h>
main()
{
        int a,b,c;
        clrscr();
        printf("Enter the Value of a & b");
        scanf("%d%d", &a,&b);
        c=a+b;
        printf("C = %d",c);
```

```
        getch();
    }
```

Q. 10) Explain the use of switch statement in C with its syntax and example.

Ans : Switch Case Statement

When we want to execute the multiple statement at that time we used switch case statement . It is like a else if ladder, but suppose in else if ladder there are five statement and our fifth statement is executed then compiler start the checking execution from first statement but in switch case there is no need to check each and every condition it will jump on particular statement.

Syntax:

```
switch(expression)
    {
        case1: statement  1;
            break;
        case 2: statement  2;
            break;
        case 3: statement 3;
            break;
            |
           |
        case 4: statement N;
            break;
        default:  default statement;
  }
```

The default statement is executed when no statement is executed.

Rules for switch case

1) The switch expression must be integer and character type.
2) The case value should be character or integer constant.
3) The floting point value does not allowed in expression.

EX:

```
#include<stdio.h>
    #include<conio.h>
    void main()
    {
int choice;
        printf("Enter your choice");
        scanf("%d" &choice);
            switch(choice)
            {
            Case 1:
                    printf(" First \n");
                    break; //optional
            case 2:
                    printf("Second \n");
                    break;
            case 3:
                    printf("Three \n");
                    break;
            default:
                    printf("Wrong choice \n");
            }
    getch();
```

```
}
```

Q.11) What are conditional Statement in C? Explain any two with example.

Ans : Ref. Q. No. 12 & 13 (3 Marks Each)

Q. 12) What is else… if ladder? Explain with suitable Example.

Ans : The conditions are evaluated from the top downward. As soon as a true condition is found, the statement associated with it is executed and the rest of the ladder is bypassed.

EX: Program to find out the grade of a student when the marks of 4 subject are given.

```
#include<stdio.h>
#include<conio.h>
main()
 {
    float m1,m2,m3,m4,total,per;
    char grade;
    clrscr();
    printf("Enter marks of subject:");
    scanf("%f%f%f%f",&m1,&m2,&m3,&m4);
    total = m1+m2+m3+m4;
    per = total/4.0;
    if(per>85)
                    grade='A';
    else if(per>=75)
                    grade='B';
    else if(per>=55)
                    grade='C';
    else if(per>=40)
                    grade='D';
    else if(per>=40)
                    grade='E';
    printf("percentage of %f, Grade is %c, \n",per,grade);
    getch();
 }
```

Q. 13) Explain For Loop with syntax and suitable Example.

Ans : Block of codes will be executed until test condition become false.If test codition give false value then control exits from loop and passes to next satement just after the loop.

working of for loop

1. When we are working with for loop always execution process will start from initialization block
2. After initialization block control will pass to condition block, if condition is executed as true then control will pass to statement block
3. After execution of the statement block control will pass to iteration block , from iteration it will pass to back to the condition.
4. Always repetition will happened between beginning condition, statement block and iteration statement.
5. Initialization block will be executed only once when we are entering into the loop first time.
6. When we are working with for loop everything is optional but mandatory to place 2 semicolon. for(; ;)
7. When we are working with the for loop if condition part is not given the it will repeats infinite times , because contional part will be replaced by a non zero value . so it will be always

true.Like
for(; 1 ;)

8. if we use like for(; 0 ;) the it will execute only one time.

The for loop

For loop contain 3 parts

- initialization
- test condition
- iteration

Syntax : **for**(initialization; test condition ; iteration;)
{
block of code to be executed
}

Example : Program to display number between 1 to 10.

```
#include<stdio.h>
main()
{
        int i;
        for(i=1;i<=10;i++)
        {
                printf("%d",i);
        }
}
```

Q. 14) What is type casting ? Explain with suitable example.

=>Typecasting

Conversion of one datatype to another datatype is known as type casting , C support two types of typecasting .

1>Implicit typecasting

2>Explicit typecasting

Implicit type casting

The conversion does by compiler itself it is called as implicit typecasting.

Ex:

```
#include<stdio.h>
#include<conio.h>
void main()
{
char  c1,c2;
int  i1,i2;
float  f1,f2;
clrscr();
c1='H';
i1=80.56; /*float converted to int only 80 assigned to i1 */
f1=12.6;
c2=i1;   /*int converted to char */
i2=f1;   /*float converted to int */
/*Now c2 has the charecter with ASCII value 80,i2 is assigned value 12 */
printf("\nc2 = %c , i2 = %d",c2,i2);
f2=i1;    /*int converted to float */
i2=c1;    /*char converted to int */
/*Now i2 contains ASCII value of charecter 'H'which is 72 */
printf("\nf2 = %.2f , i2 = %d",f2,i2);
```

```
getch();
}
```

Q. 15) How can getchar() function be used to read a set of characters from the standard input?

Ans : getchar() function is used to get/read a character from keyboard input. Please find below the description and syntax for above file handling function.

Declaration: int getchar(void)

getchar() function is used to get/read a character from keyboard input. In a C program, we can use getchar function as below.

getchar(char);

where, char is a character variable/value

Example :

```
#include <stdio.h>
#include <ctype.h>
main()
{
  char c;
  printf("Enter some character. Enter $ to exit...\n");
  while (c != '$');
  {
    c = getchar();
    printf("\n Entered character is: ");
    putchar(c);
    printf("\n")
  }
}
```

Q. 15) How can putchar() function be used to read a set of characters from the standard input?

Ans : putchar() function is a file handling function in C programming language which is used to write a character on standard output/screen.

Declaration: int putchar(int char)

putchar() function is used to write a character on standard output/screen. In a C program, we can use putchar function as below.

putchar(char);

where, char is a character variable/value.

Example Ref in above question

Q. 17) What are the format codes used along with the scanf() Function to display the various data types in c ?

Ans : Integer Data type : scanf(“%d”,&a); scanf(“%ld”, &a); scanf(“%u”, &a);

Float Data type : scanf(“%f”,&a); scanf(“%lf”,&a)

Char Data type : scanf(“%c”,&a)

Q. 18) What is meant by the precision of an output data and how can the precision be specified within a printf() Function.

Ans :-

Q. 20) What is significance of break statement and within the switch construct ?

Ans :- Ref Q. No 10 (Five Marks)

Q. 21) Write a program in C to print Fibonacci series of n terms.

Ans :

```
#include <stdio.h>
main()
{
   int i, n, t1 = 0, t2 = 1, nextTerm;
```

```
   printf("Enter the number of terms: ");
   scanf("%d", &n);
   printf("Fibonacci Series: ");
   for (i = 1; i <= n; ++i)
   {
      printf("%d, ", t1);
      nextTerm = t1 + t2;
      t1 = t2;
      t2 = nextTerm;
   }
}
```

Q.22) Write a program in c to check whether entered character is consonant or vowel using switch case block.

Ans :-
```
#include <stdio.h>
main()
{
 char ch;
  printf("Input a character\n");
 scanf("%c", &ch);
  switch(ch)
 {
   case 'a':
   case 'A':
   case 'e':
   case 'E':
   case 'i':
   case 'I':
   case 'o':
   case 'O':
   case 'u':
   case 'U':
     printf("%c is a vowel.\n", ch);
break;
   default:
     printf("%c is not a vowel.\n", ch);
 }
}
```

Q**. 23) Write a program in c to convert the total number of day into number of years, month and remaining day. Consider 360 days in a year and 30 day days in a month.**

Ans :- Ref. Practical Book

Q.24) Write a program in c language to check whether the even or old?

Ans : Ref. Practical Book

Q.25) Write a program the enter temperature in Celsius and convert into the Fahrenheit.?

Ans : Ref. Practical Book

UNIT 2 (2 Marks)

Q. 1) List the type of function?

Ans:- A function is a self contained block which design and execute separately. C support two types of function. 1) standard library function 2) user defined function

Q. 2) What is the use of 'return' statement in C?

Ans :- The return statement is used for returning the value from function. The return statement **terminates** the execution of a function and returns control to the calling function. Execution resumes in the calling function at the point immediately following the call. A return statement can also return a value to the calling function. See Return Type for more information.

Q.3) what is meant by scope of variable?

Ans: A **variable** can be either of global or local **scope**. A global **variable** is a **variable** declared in the main body of the source code, outside all functions, while a local **variable** is one declared within the body of a function or a block.

Q.4) What is 'void' data type?give an example.

Ans: The data type void refers to an object that does not have a value of any type. We have already seen examples of its use when we have defined functions that return no value, i.e. functions which only print a message and have no value to return. Such a function is used for its side effect and not for its value.

Here is a simple program using a message printing function which takes a void parameter and returns type void:

```
/*   File: msg.c
    This program introduces data type void.
*/
void printmsg(void);
main()
{
    /* print a message */
    printmsg();
}
/* Function prints a message. */
void printmsg(void)
{
    printf("****HOME IS WHERE THE HEART IS****\n");
}
```

Q.5) Describe the two ways of passing parameters to function.

Ans:-Best Answer: There are two ways of passing parameters:
1) pass by value :-In this method value of the variable is passed. Changes made to formal will not affect the actual parameters. i.e. different memory locations will be created for both variables.
2) pass by reference : In Pass by reference address of the variable is passed to a function. Whatever changes made to the formal parameter will affect to the actual parameters i.e. same memory location is used for both variables.

Q. 6) What is meant by calling and call function.

Ans:- A **function** is a group of statements that together perform a task. A **function** declaration tells the compiler about a **function's** name, return type, and parameters. A **function definition** provides the actual body of the **function**. The C standard library provides numerous built-in **functions** that your program can **call**.

Q.7) define 'nested function'.

Ans: A **nested function** is a **function defined** inside another **function**. **Nested functions** are supported as an extension in GNU C, but are not supported by GNU C++. The **nested function's** name is local to the block where it is **defined**.

Q.8) What is user defined function?

Ans: A function is a block of code that performs a specific task.C allows you to define functions according to your need. These functions are known as user-defined functions.

Q.9) What is actual parameter?

Ans: **Actual parameters** and formal **parameters**. ... They are called "**parameters**" because they define information that is passed to a function. **Actual parameters** are **parameters** as they appear in function calls. Formal **parameters** are **parameters** as they appear in function declarations.

Q.10) what is formal parameter?

Ans: The term **parameter** (sometimes called **formal parameter**) is often used to refer to the variable as found in the function definition, while **argument** (sometimes called actual **parameter**) refers to the actual input passed.

Q.11) what is an array?

Ans: An array is a data structure that contains a group of elements. Typically these elements are all of the same data type, such as an integer or string. Arrays are commonly used in computer programs to organize data so that a related set of values can be easily sorted or searched.

Q.12) List different type of array?

Ans :- there are two types of array in C C support two types of array 1 one dimensional array 2 Two dimensional array

Q.13)What is multidimensional array in C?

Ans :- Array represent two subscript values are called as two dimensional array. First subscript represent row and second represent column .

Q. 14) What is recursion?

Ans: **Recursion** is a programming technique that allows the programmer to express operations in terms of themselves. In **C**, this takes the form of a function that calls itself. A useful way to think of **recursive** functions is to imagine them as a process being performed where one of the instructions is to "repeat the process".

Q. 15) What is local variable? give an example.

Ans:- These variables only exist inside the specific function that creates them. They are unknown to other functions and to the main program.

Q.16.what is global variable?give an example.

Ans:A variable that is declared outside the function or block or body is called global variable. If any function changes the value of the global variable so it will be changed for all. It must be declared at the start of the block.

Q.17)How we can declare a function in c?

Ans: To create a **function** that **can** accept a variable-length argument list, you must first include the standard library header stdarg.h.

Next, **declare** the **function** as you would normally.

Next, add as the last argument an ellipsis ("...").

This indicates too the compiler that a variable list of arguments is to follow.

Syntax

```
function_name (parameterlist)
{
//function body
}
```

Q. 18) How 'auto' storage class is differ from 'register' storage class.

Ans: Auto Storage Class : Any variable declared within a bracket are comes under auto storage class because they gets memory automatically whenever function is call and they destroy automatically whenever function is destroy.

Register storage class: To increase your program efficiency or making a fast program we used register variable. Register contain by default garbage and it scope within a bracket.

Q.19) list the various Types of Storage Classes in C?

Ans:There are four storage classes in C they are as follows:

1. Automatic Storage Class
2. Register Storage Class
3. Static Storage Class
4. External Storage Class

Q. 20)What are storage variable in c?

Ans : Every **C variable** has a **storage** class and a scope. The storage class determines the part of memory where storage is allocated for an object and how long the storage allocation continues to exist. It also determines the scope which specifies the part of the program over which a **variable** name is visible, i.e. the **variable** is accessible by name. The are four **storage** classes in **C** are automatic, register, external, and static.

Q.21) Why do you need "static" keyword in an array declaration?

Ans : The static keyword in C is a storage-class specifier. It has different meanings, depending on the context. Inside a function it makes the variable to retain its value between multiple function calls. Outside of a function it restrains the visibility of the function or variable to the current file (compilation unit).

Q. 22) what is subscripted variable?

Ans: A **subscripted variable** is the combination of the array name and a **subscript**. You can use a **subscripted variable** anyplace an ordinary **variable** can go. For example, in the above program a value was copied from a **subscripted variable** array an ordinary **variable**: LET HOTDATE$ = DATE$(5)

Q. 23) what do you mean by array indexing?

Ans: In computer science, an **array** data structure, or simply an **array**, is a data structure consisting of a collection of elements (values or variables), each identified by at least one **array index** or key. An **array** is stored so that the position of each element **can** be computed from its **index** tuple by a mathematical formula.

Q.24) what is mean by passing an array?

Ans: in **passing an array** as a parameter to a function it is **passed** as a reference parameter. ... Since **arrays** are **passed** by reference this **means** that if the function changes the value of an element in an **array** that is a parameter of the function then the corresponding actual **array** of the call will have that element changed.

Q.25) state the limitation of array.

Ans: Following are some listed limitations of Array in C Programming.

a) Static Data b) Can hold data belonging to same Data types c) Inserting data in Array is Difficult d) Deletion Operation is difficult e) Bound Checking

f) Shortage of Memory g) Wastage of Memory

UNIT 2 (3 Marks)

Q. 1) What is the advantages of function ?

Ans : Advantages of Functions

There are several advantages in using functions. They are discussed below.

1. Functions allow the **divide and conquer strategy** to be used for the development of programs.

2. Functions help avoid duplication of effort and code in programs. During the development of a program, the same or similar activity may be required to be performed more than once. 3. Functions enable us to hide the implementation details of a program, e. g., we have used library functions such as sqrt, log, sin, etc.

4. The divide and conquer approach also allows the parts of a program to be developed, tested and debugged independently and possibly concurrently by members of a programming team.

5. The functions developed for one program can be used, if required, in another with little or no modification. This further reduces program development time and cost.

Q. 2) What is function prototype?

Ans : In computer programming, a **function prototype** or **function** interface is a declaration of a **function** that specifies the **function's** name and type signature (arity, data types of parameters, and return type), but omits the **function** body. While a function definition specifies *how* the function does what it does (the "implementation"), a function prototype merely specifies its interface, i.e. *what* data types go in and come out of it. The term function prototype is particularly used in the context of the programming languages C and C++ where placing forward declarations of functions in header files allows for splitting a program into translation units, i.e. into parts that a compiler can separately translate into object files, to be combined by a linker into an executable or a library.

Q. 3) What is structure of function? Give an example.

Ans :

```
return_type function_name(parameter list)
{
body of the function
}
```

Example : /*Function returning the max between two numbers*/

```
int max(int num1, int num2)
{
int result;
if(num1>num2)
result=num1;
else
```

result=num2;
return result;
}

Q. 4) What is the purpose of return statement?

=>The **return statement** terminates the execution of a **function** and **returns** control to the calling **function**. Execution resumes in the calling **function** at the point immediately following the call. A **return statement** can also **return** a value to the calling **function**. See **Return** Type for more information.

Syntax

jump-statement:
return expression opt**;**

Q. 5) How a function declaration is different from the function definition?

Ans : A **declaration** is code that declares an identifier and its type. A **function declaration** is a **declaration** of a **function**. In other words a **function declaration** declares the name of the **function** and the type of what it returns.

A *function definition* defines the function itself. It also acts as a declaration, and if the declaration includes the types of its parameters, a prototype as well.

Q. 6) Difference between global and local variable ?

Ans : Differentiate between Global & Local variables with examples

Local Variable	Global Variable
Local variables are declared inside a function.	Global Variables are declared before the main function.
Local Variables cannot be accessed outside the function.	Global Variables can be accessed in any function.
Local Variables are alive only for a function.	Global Variables are alive till the end of the program.

Q. 7) What is argument? List different types of argument?

Ans : An **argument** in context with functions is the actual value that is passed to the function (as input) ,when it is called. However **parameter** refers to the variables that are used in the function declaration/definition to represent those **arguments** that were send to the function during the function call.

List of argument :

Q. 8) List out the rules normally governing the use of the return statement.

Ans :

Return statement rules as follows:

1. Every function always returns something.
2. Return statement must be last statement of a function.
3. Return statement depends on return type of function.
4. If nothing is to be return, use void.

Q.9) Differentiate user defined function and standard library function.

Ans : The user-defined functions are defined by a user as per its own requirement and library functions come with compiler.

Library Function	User Defined Functions
• LF(library functions) are Predefined functions.	• UDF(user defined functions) are the function which r created by user as per his own requirements.
• UDF are part of the program which compile runtime	• LF are part of header file (such as MATH.h) which is called runtime.
• In UDF the name of function id decided by user	• in LF it is given by developers.
• in UDF name of function can be changed any time	• LF Name of function can't be changed.
Example : SIN, COS, Power	*Example : fibo, mergeme*

Q.10) How can data be initialized the automatic variable?

Ans : Any variable declared within a bracket are comes under auto storage class because they gets memory automatically whenever function is call and they destroy automatically whenever function is destroy.

e.g auto int a=10;

Q.11) How is a register variable different from an automatic variable?

Ans : To increase your program efficiency or making a fast program we used register variable. Register contain by default garbage and it scope within a bracket.

e.g register int i;

Q.12) How data elements initialized in the case of static type variable?

Ans : Static by default contain zero value, and always static variable initialize once.

e.g static int a=20;

Q.13) How a static variable different from an automatic variable ?

Ans : Static by default contain zero value, and always static variable initialize once where as automatic variable initialise many times.

Q.14) What is the use of external data type in C.

Ans : Whenever any variable declared outside the function then that variable are called as global variable or external variable.

e.g extern int i=20;

Q.15) What is multifunction program? State its syntax?

Ans : A function is a self contained block of code that performs a particular task. One a function has been designed and packed, it can be treated as a 'black box' that takes some data from the main program and returns a value. Thus program, which has been written using a number of functions, is treated as a multifunction program.

Syntax :

Q.16) How is a recursive function different from an ordinary function?

Ans : 1) A voidance of unnecessary calling of functions. 2) A substitute for iteration where the interactive solution is very complex. For example to reduce the code size for tower of honai applications, a recursive function is best suited. 3) Extremely useful when applying the same solution.

Q.17) What is array? How do you initialize an array?

And : Ref. Qu. No. 11(2marks)

Following are the statements in which show the initialization of array :

i) int students[10] ={1,2,3,4,5,6,7,8,9,10};

ii) int students[10] = {1,2};

iii) int students[10];

Q.18) What are the rules used to declare a one dimensional array?

Ans :- In a C Program an array is declared before using it. Following is the syntax used to declare an array

Syntax : <data_type>array_Name[size1][size2]….[sizen];

Rules : 1) give the data type name 2) Give the Array Name 3) After that declare size of array

Representation of array elements in a single row is known as one dimensional array.

Q. 19) What is array and how is an array variable different from an ordinary variable.

Ans : Ref. Qu. No. 11(2 marks)

Array variable different from ordinary variable

1) All sorts of lists can be presented by using array.
2) Various other data structures such as stacks, queues, heaps, etc., are implemented by using array.

Q. 20) Give an example of "Passing one-dimensional arrays to function"

Ans : double getAverage(int arr[ ], int size)

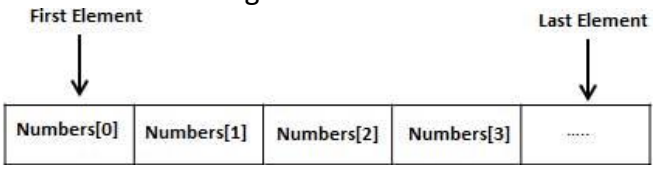
```
{
int I;
double avg;
double sum;
for(i=0;i<size;++i)
{
Sum +=arr[i];
}
avg=sum/size;
return avg;
}
/*Now call the above function as follows :*/
#include<stdio.h>
double getAverage(int arr[ ], int size[ ]);
int main()
{
int balance[5] = {1000,2,3,17,50};
double avg;
avg=getAverage(balance, 5);
printf("Average value is : %f", avg);
return 0;
}
```

Q.21) What is the need of an array?

Ans : Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



Q.22) Can an array name be used as an argument to a function?

Ans : The **function** calls in main() pass the **name** of the **array**, exam_scores, as an **argument** because the **name** of an **array** in an expression evaluates to a pointer to the **array**. Its type is, therefore, int *, and a called **function uses** this pointer (passed as an **argument**) to indirectly access the elements of the **array**.

Q.23) What is meant by passing an array? State the limitation of an array.

Ans : If you want to pass a single-dimension array as an argument in a function, you would have to declare a formal parameter in one of following three ways and all three declaration methods produce similar results because each tells the compiler that an integer pointer is going to be received.

Limitation of an Array : Following are the statements in which show the initialization of array :

i) int students[10] ={1,2,3,4,5,6,7,8,9,10};

ii) int students[10] = {1,2};

iii) int students[10];

Q.24) How to initialize character array? Illustrate with Example.

Ans : The following initialization create a string consisting of the word “Hello”. To hold the null character at the end of the array, the size of the character array containing the string is one more than the number of characters in the word “Hello”.

Char firsstring[6]={‘H’,’e’,’l’,’l’,’o’,’\0’};

One can write the above statement as follows : char greeting[]=”Hello”;

Char greeting[ ]= “Hello”;

Following is an example of character Array :

```
#include<stdio.h>
#include<conio.h>
main()
{
char greeting[6] = {‘H’,’e’,’l’,’l’,’o’,’\0’};
printf(“Greeting message : %s\n”,greeting);
getch();
}
```

Q. 25) Distinguish between a character array and strings.

Ans :

Basis for Comparison	Character Array	String
Basic	Character array is collection of variables, of character data type.	String is class and variables of string are the object of class "string".
Syntax	char array_name [size];	string string_name;
Indexing	An individual character in a character array can be accessed by its index in array.	In string the particular character can be accessed by the function "string_name.charAt(index)".
Data Type	A character array does not define a datatype.	A string defines a datatype in C.
Operators	Operators in C can not be applied on character array.	You may apply standard C operator on the string.
Boundary	Array boundaries are easily overrun.	Boundaries will not overrun.
Access	Fast accessing.	Slow accessing.

UNIT 2 (5 Marks Each)

Q.1) What is function and list out the advantages and disadvantages of using functions in C?

Ans : A Function is a group of statement that together perform a task. Your can divide up your code into separate functions. There are numbers of functions are used in the C program. The main() function is one of them. Every C program has at least this main() function. The compiler executes a program by carrying out the instruction in main() function.

Advantages of Functions:

i) The length of a source program can be reduced by using functions at appropriate places. This factor is particularly critical with microcomputers where memory space is limited.

ii) It is easy to locate and isolate a faulty function for further investigations.

iii) A function may be used by many other programs. This means that a C programmer can build on what others have already done, instead of starting all over again from scratch.

iv) It facilitates top-down modular programming. In this programming style, the high level logic of the overall problem is solved first while the details of each lower-level function are addressed later.

v) Its interface to the rest of the program is clean and narrow.

Disadvantages of Functions:

1) While adding a user function can speed up code that is best written in C rather than a scripting language, it is not always the best choice for implementation:

2) It requires the programmer to be well versed in C, including pointers, function pointers, dynamic memory allocation, and debugging. Often the headaches C causes, especially for the neophyte, far outweigh any run-time savings. Bugs in the code might not manifest themselves until well after the C function ends, making debugging a nightmare.

3) There may not be any speed advantage. Vortex is pretty fast at most operations; for small functions it may be just as fast - and much easier - to write the function in Vortex. Since Vortex already has powerful data processing functions, and the ability to execute external programs, it may be faster to <EXEC> the C code in a separate program and parse it in Vortex, especially as a quick prototype.

4) It's less portable. A C function means a new Vortex executable must be made if the hardware platform changes. Other Vortex users won't have the custom function in their taxis executable.

Q. 2) What is meant by the function arguments, function call and return values?

Ans : Function Arguments : Refe. Q. No. 7(3 Marks),

Function Call : In order to call access any function, one has to specify the name of function and list of parameters surrounded by the parantheses.

The function call can be located either in simple expression or can take the from of operand in a complex expression. It a function call exists within an expression, then it will be assessed initially next the parentheses will be executed. Thus, a function call is a postfix expression. The operator used (.) shows the high level of procedure.

Return Values : The return statement terminates the execution of a function and returns a value to the calling function. The program control is transferred to the calling function after return statement. In the above example, the value of variable *result* is returned to the variable *sum* in the main() function.

Q. 3) Explain user defined function in details.

Ans : A function is a block of code that performs a specific task.

C allows you to define functions according to your need. These functions are known as user-defined functions. For example:

Suppose, you need to create a circle and color it depending upon the radius and color. You can create two functions to solve this problem:

- createCircle() function
- color() function

Example: User-defined function

Here is a example to add two integers. To perform this task, a user-defined function addNumbers() is defined.

```
#include <stdio.h>
int addNumbers(int a, int b);       // function prototype
int main()
{
   int n1,n2,sum;

   printf("Enters two numbers: ");
   scanf("%d %d",&n1,&n2);
   sum = addNumbers(n1, n2);       // function call
   printf("sum = %d",sum);
   return 0;
}
int addNumbers(int a,int b)       // function definition
{
   int result;
   result = a+b;
   return result;              // return statement
}
```

Q. 4) What is multi function program? Explain with suitable example.

=> Refe. Q. No. 15(3 Marks>

Example.

```
#include<stdio.h>
#include<conio.h>
main()
{
printine();
printf("This illustrates the use of C functions \n");
printline();
}
printline()
{
        int i;
        for(i=1;i<40;i++)
                print("_");
                print("\n");
}
```

Q.5) Different between actual and formal argument to a function.

Ans : **Argument:** An argument is an expression which is passed to a function by its caller in order for the function to perform its task. It is an expression in the comma-separated list bound by the parentheses in a function call expression.

Actual arguments:

The arguments that are passed in a function call are called actual arguments. These arguments are defined in the calling function.

Formal arguments:

The formal arguments are the parameters/arguments in a function declaration. The scope of formal arguments is local to the function definition in which they are used. Formal arguments belong to the called function. Formal arguments are a copy of the actual arguments. A change in formal arguments would not be reflected in the actual arguments.

Q.6) What is an automatic variable and what is the use of it?

Ans : Ref. No. No. 10(3Marks Each)

Use of Automatic Variable : . All **variables** declared within a block of code are **automatic** by default. An uninitialized **automatic variable** has an undefined value until it is assigned a valid value of its type. In **C**, using the storage class register is a hint to the compiler to cache the **variable** in a processor register.

Q. 7) What is register variable and what is the scope of it?

=> The register variable is used to define local variable that should be stored in a register instead of RAM. This means that the variable has a maximum size equal to the register size and cannot have the unary '&' operator applied to it.

```
{
/*Register variable declaration*/
register int miles;
}
```

The register should only be used for variables that require quick access such as counters. It should also be noted that defining 'register' goes not mean that the variable will be stored in a register. It means that it might be stored in a register depending on hardware and implementation restrictions.

Q. 8) What is static variable and what is its scope?

Ans :The static variable instructs the compiler to keep a local variable in existence during the lifetime of the program instead of creating and destroying it each time it comes into and goes out of scope. Therefore making local variable static allows them to maintain their values between functions calls. The static modifier may also be applied to global variable. When this is done, it causes that variable's scope to be restricted to the file in which it is declared.

When static is used on a class data member, it causes only one copy of that member to be shared by all objects of its class.

Q.9) What is recursive function ? List out their merits and demerits of the function.

Ans : **Recursion** is a programming technique that allows the programmer to express operations in terms of themselves. In **C**, this takes the form of a **function** that calls itself. A useful way to think of **recursive functions** is to imagine them as a process being performed where one of the instructions is to "repeat the process".

Merits : 1) A voidance of unnecessary calling of function. 2) A substitute for iteration where the iterative solution is very complex. For example to reduce the code size for tower of honai application, a recursive function is b0 est suited. 3) Extremely useful when applying the same solution.

Demerits : 1)A recursive function is often confusing.

2) The exit point must be explicitly coded.

3) It is difficult to trace the logic of the function.

Q.10) Distinguish between call by value and call by reference. Illustrate with example.

Ans : **Call By Value :** Call by value means sending the values of the arguments to functions. When a single value is passed to a function via an actual argument, the value of the actual argument is copied into the function. Therefore, the value of the corresponding formal argument can be altered within the function, but the value of the actual argument within the calling routine will not change. This procedure for passing the value of an argument to a function is known as passing by value or call by value.

WAP for call by value

```
#include<conio.h>
#include<stdio.h>
#include
int swap(int,int);
main()
{
 int a,b;
 clrscr();
 printf("\nEnter Value of a and b ");
 scanf("%d%d",&a,&b);
 printf("\n Before Printing ");
 printf("\na = %d and b = %d",a,b);
 swap(a,b);
 getch();
```

```
}
int swap(int a,int b)
{
 int temp;
 temp=a;
 a=b;
 b=temp;
 printf("\n After Printing ");
 printf("\na = %d and b = %d",a,b);
}
```

Call By reference : Call by reference means sending the addresses of the arguments to the called function. In this method the addresses of actual arguments in the calling function are copied into formal arguments of the called functions. Thus using these addresses we would have an access to the actual arguments and hence we would be able to manipulate them. Using a call by reference intelligently, it is possible to make a function return more than one value at a time, which involves the study of pointer.

Example :

```
#include<conio.h>
#include<stdio.h>
#include
int swap(int*,int*);
main()
{
 int a,b;
 clrscr();
 printf("\nEnter Value of a and b ");
 scanf("%d%d",&a,&b);
 printf("\n Before Printing ");
 printf("\na = %d and b = %d",a,b);
 swap(&a,&b);
 printf("\n After Printing ");
 printf("\na = %d and b = %d",a,b);
 getch();
}
int swap(int *q,int *p)
{
 int temp;
 temp=*q;
 *q=*p;
 *p=temp;
}
```

Q.11) Distinguish between iteration and recursion.

Ans :

Basis For Comparison	Recursion	Iteration
Basic	The statement in a body of function calls the function itself.	Allows the set of instructions to be repeatedly executed.

Basis For Comparison	Recursion	Iteration
Format	In recursive function, only termination condition (base case) is specified.	Iteration includes initialization, condition, execution of statement within loop and update (increments and decrements) the control variable.
Termination	A conditional statement is included in the body of the function to force the function to return without recursion call being executed.	The iteration statement is repeatedly executed until a certain condition is reached.
Condition	If the function does not converge to some condition called (base case), it leads to infinite recursion.	If the control condition in the iteration statement never become false, it leads to infinite iteration.
Infinite Repetition	Infinite recursion can crash the system.	Infinite loop uses CPU cycles repeatedly.
Applied	Recursion is always applied to functions.	Iteration is applied to iteration statements or "loops".
Stack	The stack is used to store the set of new local variables and parameters each time the function is called.	Does not uses stack.
Overhead	Recursion possesses the overhead of repeated function calls.	No overhead of repeated function call.
Speed	Slow in execution.	Fast in execution.
Size of Code	Recursion reduces the size of the code.	Iteration makes the code longer.

Q. 12) Explain ANSI function standard in details.

Ans :

Q. 14) Explain the salient features of an array and their uses?

Ans : 1) An array holds elements that have the same data type 2) Array elements are stored in subsequent memory locations3) Two-dimentional array elements are stored row by row in subsequent memory locations.4) Array name represents the address of the starting element5) Array size should be mentioned in the declaration. Array size must be a constant expression and not a variable.

Uses : Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An *array* is *used* to store a collection of data, but it is often more useful to think of an *array* as a collection of variables of the same type.

Q.15) How are the array usually processed in C ?

Ans : An array is a collection of data that holds fixed number of values of same type. For example: if you want to store marks of 100 students, you can create an array for it.

float marks[100];

The size and type of arrays cannot be changed after its declaration.

Arrays are of two types:

1. One-dimensional arrays
2. Multidimensional arrays (will be discussed in next chapter)

How to declare an array in C?

data_type array_name[array_size];

For example,

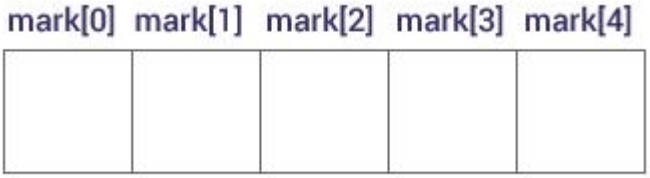
float mark[5];

Here, we declared an array, *mark*, of floating-point type and size 5. Meaning, it can hold 5 floating-point values.

Elements of an Array and How to access them?

You can access elements of an array by indices.

Suppose you declared an array *mark* as above. The first element is *mark[0]*, second element is *mark[1]* and so on.



Few key notes:

- Arrays have 0 as the first index not 1. In this example, *mark[0]*
- If the size of an array is *n*, to access the last element, (n-1) index is used. In this example, *mark[4]*
- Suppose the starting address of mark[0] is 2120d. Then, the next address, a[1], will be 2124d, address of a[2] will be 2128d and so on. It's because the size of a float is 4 bytes.

Q. 16) What is multi dimensional array? How is it different from a one dimensional array?

Ans : A array represent two subscript values are called as two dimensional array. First subscript represent row and second represent column .

Syntax:- array-name [rows][col];

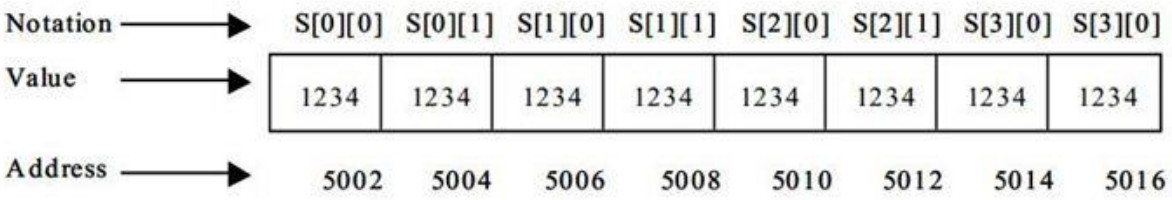
EX.

int a[3][4];

char name [10][15];

In memory, whether it is a one dimensional or a two dimensional array, the array elements are stored in one continuous chain.

The arrangement of array elements of a two dimensional array of students, which contains roll numbers in one column and the marks in the other (in memory) is shown below:



Basis for Comparison	One-Dimensional	Two-Dimensional
Basic	Store single list of elements of similar data type.	Store 'list of lists' or 'array of arrays' or 'array of one dimensional arrays'.
Declaration	/*declaration in C++ type variable_name[size];*/ /*declaration in Java type variable_name [ ]; variable_name = new type[size]; */	/*declaration in C++ type variable_name[size1][size2]; */ /*declaration in Java type variable_name= new int[size1][size2]; */

Basis for Comparison	One-Dimensional	Two-Dimensional
Alternative Declaration	/* In Java int [ ] a= new int [10]; */	/* In Java int [ ] [ ] a= new int [10][20]; */
Total Size in Bytes	Total Bytes =sizeof(datatype of array variable)* size of array.	Total Bytes= sizeof(datatype of array variable)* size of first index*size of second index.
Receiving parameter	It can be received in a pointer, sized array or an unsized array.	Parameter receiving it must define the rightmost dimension of an array.
Dimensions	One dimensional.	Two dimensional.

Q.17) Explain multi dimensional array with suitable example.

Ans. : Ref. No 16(5 Mark)

```
Example : #define ROWS 5
#define COLUMNS 5
main( )
{
int row, column, product [ROWS] [COLUMNS];
int i, j;
printf ("MULTIPLICATION TABLE \N");
printf (" ");
for (j = 1; j < = COLUMNS; j++)
printf ("%4d", j);
printf ("\n");
for (i = 0; i < ROWS; i++)
{
row = i + 1;
printf ("%2d\n", row);
for (j = 1; j < = COLUMNS; j++)
{
column = j;
product [i] [j] = row *column;
printf ("%4d", product [i] [j]);
}
printf ("\n");
}
}
```

Q. 18) Explain passing of one dimensional array to a function.

Ans : #include <stdio.h>

```
void display(int age)
{
   printf("%d", age);
}

int main()
{
   int ageArray[] = { 2, 3, 4 };
   display(ageArray[2]); //Passing array element ageArray[2] only.
   return 0;
}
```

Q. 19) Explain "Passing multi-dimensional array to a function" with example.

Ans :

```
#include <stdio.h>
void displayNumbers(int num[2][2]);
int main()
{
   int num[2][2], i, j;
   printf("Enter 4 numbers:\n");
   for (i = 0; i < 2; ++i)
      for (j = 0; j < 2; ++j)
         scanf("%d", &num[i][j]);
   // passing multi-dimensional array to displayNumbers function
   displayNumbers(num);
   return 0;
}
void displayNumbers(int num[2][2])
{
   // Instead of the above line,
   // void displayNumbers(int num[][2]) is also valid
   int i, j;
   printf("Displaying:\n");
   for (i = 0; i < 2; ++i)
      for (j = 0; j < 2; ++j)
         printf("%d\n", num[i][j]);
}
```

Q. 20) What is character Array? How it is different from other data type of array?

Ans :

Q.21) Write a program to insert a new element into the given array?

Ans :

```
#include <stdio.h>
int main()
{
  int array[100], position, c, n, value;
  printf("Enter number of elements in array\n");
  scanf("%d", &n);
  printf("Enter %d elements\n", n);
  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  printf("Enter the location where you wish to insert an element\n");
  scanf("%d", &position);
  printf("Enter the value to insert\n");
  scanf("%d", &value);
  for (c = n - 1; c >= position - 1; c--)
    array[c+1] = array[c];
```

```
  array[position-1] = value;
  printf("Resultant array is\n");
  for (c = 0; c <= n; c++)
    printf("%d\n", array[c]);
  return 0;
}
```

Q.22)Write a program to find length of an inputted string without using standard library function.

Ans :

```
#include <stdio.h>
int main()
{
   char s[1000], i;
   printf("Enter a string: ");
   scanf("%s", s);
   for(i = 0; s[i] != '\0'; ++i);
   printf("Length of string: %d", i);
   return 0;
}
```

Q. 23) Write a program to find out the factorial of inputted number using function .

Ans :

```
#include <stdio.h>
long int multiplyNumbers(int n);
int main()
{
   int n;
   printf("Enter a positive integer: ");
   scanf("%d", &n);
   printf("Factorial of %d = %ld", n, multiplyNumbers(n));
   return 0;
}
long int multiplyNumbers(int n)
{
   if (n >= 1)
      return n*multiplyNumbers(n-1);
   else
      return 1;
}
```

Q. 24) Write a program in C to input 10 element for an array and find sum and average of them.

Ans :

```
#include <stdio.h>

int main()
{
   int Arr[100], n, i, sum = 0;

   printf("Enter the number of elements you want to insert : ");
```

```
    scanf("%d", &n);

    for (i = 0; i < n; i++)
    {
        printf("Enter element %d : ", i + 1);
        scanf("%d", &Arr[i]);
        sum += Arr[i];
    }

    printf("\nThe sum of the array is : %d", sum);
    printf("\nThe average of the array is : %0.2f", (float)sum / n);

    return 0;
}
```

Q.25) WAP in c to input 10 elements for an array and print them in reverse order.

Ans :

```
#include<stdio.h>

int main() {
  int arr[30], i, j, temp;
   //Read elements in an array
  for (i = 0; i <10; i++) {
     scanf("%d", &arr[i]);
  }
   j = i - 1;   // j will Point to last Element
  i = 0;       // i will be pointing to first element
 while (i < j) {
     temp = arr[i];
     arr[i] = arr[j];
     arr[j] = temp;
     i++;           // increment i
     j--;         // decrement j
  }
   //Print out the Result of Insertion
  printf("\nResult after reversal : ");
  for (i = 0; i < 10; i++) {
     printf("%d \t", arr[i]);
  }
   return (0);
}
```

UNIT 3 (2 Marks Each)

1) What are the different uses of pointer ?

Ans : Pointers are used (in the C language) in three different ways: To create **dynamic data structures**. To pass and handle variable parameters passed to functions. To**access** information stored in arrays.

2) What is the use of address operator in pointer ?

Ans : Address operators commonly serve two purposes: To conduct parameter **passing**by reference, such as by name. To establish pointer values. Address-of operators point to the location in the memory because the value of the pointer is the memory address/location where the data item resides in memory.

3) What is NULL Pointer ?

Ans : If a **null pointer** constant is converted to a **pointer**type, the resulting **pointer**, called a **null pointer**, is guaranteed to compare unequal to a **pointer** to any object or function." ... But **C** standard is saying that 0 is also a **null pointer** constant.

4) What do you mean by pointer variable ?

Ans : A **pointer** is just like an int: a number. It happens to be a number that identifies a memory location, and if something is stored in that memory location **you** can call it an address. Like an int, a **pointer** can be stored in a **variable**. A **variable** that stores a **pointer** could be called a **pointer variable**.

5) How to declare a pointer variable ?

Ans : Dereferencing of **Pointer**. Once a **pointer** has been assigned the address of a**variable**. To access the value of **variable**, **pointer** is dereferenced, using the indirection operator * . int a,*p; a = 10; p = &a; printf("%d",*p); //this will print the value of a

6) what is indirection operator ?

Ans : The **indirection operator** is a unary **operator** represented by the symbol (*). The**indirection operator** can be used in a pointer to a pointer to an integer, a single-dimensional array of pointers to integers, a pointer to a char, and a pointer to an unknown type.

7) What is pointer operator ?

Ans : Advertisements. A **pointer** in **c** is an address, which is a numeric value. Therefore, you can perform arithmetic operations on a **pointer** just as you can on a numeric value. There are four arithmetic **operators** that can be used on **pointers**: ++, --, +, and –

8) How many different operations we can perform using pointer arithmetic ?

Ans : Valid Pointer Arithmetic Operations

- Adding a number to pointer.
- Subtracting a number form a pointer.
- Incrementing a pointer.
- Decrementing a pointer.
- Subtracting two pointers.
- Comparison on two pointers.

9) state the arithmetic operations that cannot be perform on pointer ?

Ans : One can perform different arithmetic operations on Pointer such as increment,decrement but still we have some more arithmetic operations that cannot be performed on pointer –

1. Addition of two addresses.
2. Multiplying two addresses.

3. Division of two addresses.
4. Modulo operation on pointer.
5. Cannot perform bitwise AND,OR,XOR operations on pointer.
6. Cannot perform NOT operation or negation operation.

10) List any four commonly used header files ?

Ans : #include<stdio.h>
#include<conio.h>
#include<math.h>
#inlude<file>

11) List any four string library function ?

Ans : he nine most commonly used functions in the string library are:

- strcat - concatenate two strings.
- strchr - string scanning operation.
- strcmp - compare two strings.
- strcpy - copy a string.
- strlen - get string length.
- strncat - concatenate one string with part of another.
- strncmp - compare parts of two strings.

More items...

12.) Define C pre-processor ?

Ans : The **C Preprocessor**. The **C preprocessor** is a **macro processor** that is used automatically by the **C** compiler to transform your program before actual compilation. It is called a **macro processor** because it allows you to **define macros**, which are brief abbreviations for longer constructs. ... **C** preprocessors vary in some details.

13) Write down the rules for C pre-processor ?

Ans : The C preprocessor provides four separate facilities that you can use as you see fit:

- Inclusion of header files. These are files of declarations that can be substituted into your program.
- Macro expansion. You can define *macros*, which are abbreviations for arbitrary fragments of C code, and then the C preprocessor will replace the macros with their definitions throughout the program.
- Conditional compilation. Using special preprocessing directives, you can include or exclude parts of the program according to various conditions.
- Line control. If you use a program to combine or rearrange source files into an intermediate file which is then compiled, you can use line control to inform the compiler of where each source line originally came from.

14) List any four pre-processor directives along with its purpose ?

Ans : Before a C program is compiled in a compiler, source code is processed by a program called preprocessor. This process is called preprocessing.

Commands used in preprocessor are called preprocessor directives and they begin with "#" symbol.

15) State any two mathematical function in C with its use .

Ans :

abs labs llabs	computes absolute value of an integer value
fabs	computes absolute value of a floating point value

div ldiv lldiv	computes the quotient and remainder of integer division
fmod	remainder of the floating point division operation
remainder	signed remainder of the division operation
remquo	signed remainder as well as the three last bits of the division operation
fma	fused multiply-add operation
fmax	larger of two floating point values
fmin	smaller of two floating point values

16) What are predefined macros ?

Ans : This **macro** is **defined** when the C++ compiler is in use. You can use __cplusplus to test whether a header is compiled by a **C** compiler or a C++ compiler. This **macro** is similar to __STDC_VERSION__ , in that it expands to a version number. ... This**macro** is **defined**, with value 1, when the Objective-**C** compiler is in use.

17) List the pre-processor conditional commands.

Ans : 1 Directives. 1.1 #include. 1.1.1 Headers. 1.2 #pragma. 1.3 #define. 1.4 macros. 1.5 #error. 1.6 #warning. 1.7 #undef. 1.8 #if,#else,#elif,#endif (conditionals) 1.9 #ifdef,#ifndef.

- 2 Useful Preprocessor Macros for Debugging. 2.1 Compile-time assertions. 2.2 X-Macros.

18) Define pointer to pointer ?

Ans : A **pointer** is a variable which contains the address in memory of another variable. We can have a **pointer**to any variable type. The unary or monadic operator & gives the ``address of a variable''. The indirection or dereference operator * gives the ``contents of an object pointed to by a **pointer**''.

19) what is the use of <math.h> file ?

Ans : C Programming/**math.h**. **math.h** is a header file in the standard library of the C programming language designed for basic mathematical operations. Most of the functions involve the use of **floating** point numbers.

20) what is the purpose of <string.h> file.

Ans : C Programming/string.h. **string.h** is the **header** in the **C** standard library for the **C programming** language which contains macro definitions, constants, and declarations of functions and types used not only for **string** handling but also various memory handling functions; the name is thus something of a misnomer.

21) What is difference between #include <file.h> and #include ”file.h”.

22) what is meant by ‘call by reference’ ?

Ans : The **call by reference** method of **passing** arguments to a **function** copies the address of an argument into the formal parameter. Inside the **function**, the address is used to access the actual argument used in the **call**. It **means** the changes made to the parameter affect the passed argument.

23) under what conditions two pointer variables are added ?

Ans : C program to add **two** numbers using **pointers**. This program performs **addition of two** numbers using **pointers**. In our program we have **two two** integer variables x, y and **two pointer** variables p and q. Firstly we assign the addresses of x and y to p and q respectively and then assign the **sum** of x and y to variable **sum**.

24) under what conditions two pointer variables are subtracted ?

25) under what conditions two pointer variables are compared ?

Unit 3 (3 Marks)

1) Define pointer? State its advantages ?

Define :- A **pointer** is a variable which contains the address in memory of another variable. We can have a **pointer**to any variable type. The unary or monadic operator & gives the ``address of a variable''. The indirection or dereference operator * gives the ``contents of an object pointed to by a **pointer**''.

Advantages :-

Major **advantages** of **pointers** are: (i) It allows management of structures which are allocated memory dynamically. (ii) It allows passing of arrays and strings to functions more efficiently. (iii) It makes possible to pass address of structure instead of entire structure to the functions.

2) Difference between arrays and pointers?

Ans : **Differences between arrays**, **pointers** and strings in **C**. ... An **array** of chars is created by the compiler and it has the value String . Then, this **array** is considered as a **pointer** and the program assigns to the **pointer** string a **pointer** which points to the first element of the **array** created by the compiler.

Pointer	Array
1. A pointer is a place in memory that keeps address of another place inside	1. An array is a single, pre allocated chunk of contiguous elements (all of the same type), fixed in size and location.
2. Pointer can't be initialized at definition.	2. Array can be initialized at definition. Example int num[] = { 2, 4, 5}
3. Pointer is dynamic in nature. The memory allocation can be resized or freed later.	3. They are static in nature. Once memory is allocated , it cannot be resized or freed dynamically.
4. The assembly code of Pointer is different than Array	4. The assembly code of Array is different than Pointer.

1. What does it mean when a pointer is used in an if statement ?

   Any time a pointer is used as a condition, it means ?Is this a non-null pointer?? A pointer can be used in an if, while, for, or do/while statement, or in a conditional expression. Source: CoolInterview.com

   A pointer in an if statement is normal, and may have many meanings, not just limited to non-null. It may mean
   - whether two pointers are the same;
   - whether the pointer pointed value is a special value, like if((*p)==1);
   - if the pointer is char *, more string related function calling may appeared, like if(strcmp(a,b))
   Source: CoolInterview.com

2. When would you use a pointer to a function ?

   Function pointers can be useful when **you** want to create callback mechanism, and need to pass address of an **function** to another **function**. They **can** also be useful when **you** want to store an array of **functions**, to call dynamically for example. One common **use** is to implement a callback **function**.
3. What are the disadvantages of using pointer?

   Advantages And **Disadvantages Of Using** A **Pointer** Computer Science Essay. ...
   2)since **using** return statement a function can only pass back a single value to the calling function, **pointers** allows a function to pass back more than one value by writing them into memory locations that are accessible to calling function.

4. How is pointer variable different from ordinary variable ?

   The **declaration** of a **pointer variable** and assigning an address to it can be done as follows. int *p; p=# ... A **variable** holds a value as per the specified datatype(like int, float or char) whereas a **pointer variable** holds the address or the memory location of **another variable** or **another pointer variable**.
5. What are the scope rules of a pointer variable ?
6. What is the use of an indirection operator ?

   The indirection operator is a **unary** operator represented by the symbol (*). The indirection operator can be used in a pointer to a pointer to an integer, a single-dimensional array of pointers to integers, a pointer to a char, and a pointer to an unknown type.
7. What is the relationship between a pointer and an array ?

   That is, a is exactly the **same** as &a[0]. The only difference between a and a pointer**variable** is that the array name is a constant pointer - you cannot change the location it points at. When you write an expression such as a[i] this is converted into a pointer expression that gives the value of the appropriate element.
8. What is pointer to pointer ? Give an example.

   A **pointer** to a **pointer** is a form of multiple indirection, or a chain of **pointers**. Normally, a **pointer** contains the address of a variable. When we define a **pointer** to a **pointer**, the first **pointer** contains the address of the second **pointer**, which points to the location that contains the actual value as shown below.
   Example:-
9. How can an indirection operator by used to access a multidimensional array ?
10. How can a portion of an array be used to a function ?

11. How can a one dimensional array of pointer be used to represent a collection of strings ?
12. What are the advantages of declaring pointer variable in a function declaration?
13. What is the different between the array of pointer and pointer to the array ?
14. What is the role of preprocessor in a C program ?

    The **C preprocessor** is a macro processor that is used automatically by the **C** compiler to transform your program before actual compilation. It is called a macro processor because it allows you to define macros, which are brief abbreviations for longer constructs. **C preprocessor**. ... The **preprocessor** provides the ability for the inclusion of header files, macro expansions, conditional compilation, and line control. In many **C**implementations, it is a separate **program** invoked by the compiler as the first part of translation.
15. Explain the need for #define preprocessor directive in a c program.

    The **preprocessor** provides the ability for the inclusion of header files, macro expansions, conditional compilation, and line control. In many **C** implementations, it is a separate **program** invoked by the compiler as the first part of translation.

The #define directive is used to define values or macros that are used by the preprocessor to manipulate the program source code before it is compiled. Because preprocessor definitions are substituted before the compiler acts on the source code, any errors that are introduced by #define are difficult to trace.

By convention, values defined using #define are named in uppercase. Although doing so is not a requirement, it is considered very bad practice to do otherwise. This allows the values to be easily identified when reading the source code.

Today, #define is primarily used to handle compiler and platform differences. E.g., a define might hold a constant which is the appropriate error code for a system call. The use of #define should thus be limited unless absolutely necessary; typedef statements and constant variables can often perform the same functions more safely.

16. Explain the usage of #include directive in a c program.

    #include

    C has some features as part of the language and some others as part of a **standard library**, which is a repository of code that is available alongside every standard-conformant C compiler. When the C compiler compiles your program it usually also links it with the standard C library. For example, on encountering a #include <stdio.h> directive, it replaces the directive with the contents of the stdio.h header file.

    When you use features from the library, C requires you to *declare* what you would be using. The first line in the program is a **preprocessing directive** which should look like this:

```
#include <stdio.h>
```

17. What is meant by macro operator ?Given an example.

    The double-number-sign or "token-pasting" **operator** (##), which is sometimes called the "merging" **operator**, is used in both object-like and function-like **macros**. It permits separate tokens to be joined into a single token and therefore cannot be the first or last token in the **macro definition**.
18. Differentiate between function and macros .

No	Macro	Function
1	Macro is **Preprocessed**	Function is **Compiled**
2	**No Type Checking**	**Type Checking** is Done
3	**Code** Length **Increases**	**Code** Length remains **Same**
4	Use of macro can lead to **side effect**	No **side Effect**
–		
5	Speed of Execution is **Faster**	Speed of Execution is **Slower**
6	Before Compilation macro name is replaced by macro value	During function call , Transfer of Control takes place
7	Useful where small code appears many time	Useful where large code appears many time
8	Generally Macros do not extend beyond one line	Function can be of any number of lines
9	Macro does not Check **Compile Errors**	Function Checks **Compile Errors**

19. Distinguish between #include and #define .
20. Explain the usage of #undef directive in a c program
21. What is the purpose of #error directive in a C program?
22. What is macro and how is it different from a preprocessor?
23. List any three string library function with example .

IV [3marks]

1)define a structure give an example

Ans:- A **struct** in the **C** programming language (and many derivatives) is a composite data type declaration that **defines** a physically grouped list of variables to be placed under one name in a block of memory, allowing the different variables to be accessed via a single pointer, or the **struct** declared name which returns the ...

Example:-

```
#include<stdio.h>
```

```
#include<conio.h>

void display(struct Student *);
void inc_marks(struct Student *);
struct Student
{
 char name[10];
 int rollno;
 int marks;
};
main()
{
 struct Student stud1={"Mary",33,87};
 struct Student stud2={"John",77,47};
 clrscr();
 inc_marks(&stud1);
 inc_marks(&stud2);
 display(&stud1);
 display(&stud2);
// clrscr();
 getch();
}
void inc_marks(struct Student *studptr)
{
 (studptr->marks)++;
}
 void display(struct Student *studptr)
 {
 printf("
Name : %s",studptr->name);
 printf("
Rollno : %d",studptr->rollno);
```

```
 printf("
Marks : %d",studptr->marks);
}
```

2)how can the size of structure be determined.

Ans:- In the programming languages C and C++, the unary operator **sizeof** generates the size of a variable or datatype, measured in the number of *char* size storage units required for the type. As such, the construct sizeof (char) is guaranteed to be 1. The actual number of bits of type char is specified by the preprocessor macro CHAR_BIT, defined in the include file limits.h. On most modern systems this is eight bits

OR

"sizeof(a) == 8", on a 32-bit machine. The total size of the structure will depend on the packing: In my case, the default packing is 4, so 'c' takes 4 bytes, 'b' takes one byte, leaving 3 padding bytes to bring it to the next multiple of 4: 8. If you want to alter this packing, most compilers have a way to alter it, for example, on MSVC:

Example:-

```
#include <stdio.h>

typedef struct { char* c; char b; } a;

int main()
{
   printf("sizeof(a) == %d", sizeof(a));
}
```

3)how member of structure can be accessed.?

Ans:- Structure and union members are accessed using the following two selection operators:

- . (period)
- -> (right arrow)

The operator . is called the direct member selector and it is used to directly access one of the structure's members. Suppose that the object s is of the struct type S and m is a member identifier of the type M declared in s, then the expression

```
s.m   // direct access to member m
```

is of the type M, and represents the member object m in S.

The operator -> is called the indirect (or pointer) member selector. Suppose that the object s is of the struct type S and ps is a pointer to s. Then if m is a member identifier of the type M declared in s, the expression

```
ps->m   // indirect access to member m;

    // identical to (*ps).m
```

is of the type M, and represents the member object m in s. The expression ps->m is a convenient shorthand for (*ps).m.

4)how can a structure be inisialize and what are the scop rules for that?

Ans:-1) When we declare a structure, memory is not allocated for un-initialized variable.

2)Let us discuss very familiar example of structure student , we can initialize structure variable in different ways –

1) One cannot initialize individual member inside the structure template.

2) The order of values enclosed in braces must match the order of member in the structure definition.

3) It is permitted to have partial initialization. We can initialize only the first few member and leave the remaining blank. The unutilized membered should be only at the end of the list.

5) What is the different between structure declaration and structure initialization?

Ans:- Declaration of a Structure:

Structures are syntactically defined with the word struct. So struct is another keyword

that cannot be used as variable name. Followed by the name of the structure. The data,

contained in the structure, is defined in the curly braces. All the variables that we have

been using can be part of structure.

For example:

```
struct student{
char name[60];
char address[100];
float GPA;
};
```

Initializing Structures

We have so far learnt how to define a structure and declare its variables. Let's see how

can we put the values in its data members. The following example can help us understand

the phenomenon further.

```
struct student{
char name[64];
char course[128];
int age;
int year;
};
```

Q 6) What do you mean by an array of field in a structure and how is it different from an array?

Ans:- As you know, C Structure is collection of different datatypes (variables) which are grouped together. Whereas, array of structures is nothing but collection of structures. This is also called as structure array in C.

or

In computer science, an array data structure, or simply an array, is a data structure consisting of a collection of elements (values or variables), each identified by at least one array index or key. An array is stored so that the position of each element can be computed from its index tuple by a mathematical formula.[1][2][3] The simplest type of data structure is a linear array, also called one-dimensional array

Q 7) Write the difference between an array and a structure.

Ans:-

Arrays:-

1.An array is a collection of related data elements of same type.

2.An array is a derived data type

3. Any array behaves like a built-in data types. All we have to do is to declare an array variable and use it.

4. Array allocates static memory and uses index / subscript for accessing elements of the array

5. Array is a pointer to the first element of it

6. Element access takes relatively less time.

Structures

1. Structure can have elements of different types

2. A structure is a programmer-defined data type

3.But in the case of structure, first we have to design and declare a data structure before the variable of that type are declared and used.

4. Array allocates static memory and uses index / subscript for accessing elements of the array.

.5. Structure is not a pointer

6. Property access takes relatively large time.

Q8) Distinguish a structure data type with other data type variable.

Ans:- Structured data types

Structured data types hold a collection of data values. This collection will generally consist of the primitive data types. Examples of this would include arrays, records (structs), classes and files. These data types, which are created by programmers, are extremely important and are the building block of data structures. So what exactly is a data structure

Data type:-

A data type (in computing terms) is a set of data values having predefined characteristics. You will have encountered data types during computer programming classes. Example data types would be integer, floats, string and characters. Generally in all programming languages we have a limited number of such data types. The range of values that can be stored in each of these data types is defined by the language and the computer hardware that you are using the high level language on.

Q 9) How the formal argument of a structure is passes in a faction call?

Ans:-

Q 10) Summarize a few application of a structure data type in a real application.

Ans:- The types of data structure are:
Lists: A group of similar items with connectivity to the previous or/and next data items.
Arrays: A set of homogeneous values
Records: A set of fields, where each field consists of data belongs to one data type.
Trees: A data structure where the data is organized in a hierarchical structure. This type of data structure follows the sorted order of insertion, deletion and modification of data items.
Tables: Data is persisted in the form of rows and columns. These are similar to records, where the result or manipulation of data is reflected for the whole table.

11)what is bit field ? give an example.

Ans: -A bit field is a term used in computer programming to store multiple, logical, neighboring bits, where each of the sets of bits, and single bits can be addressed. A bit field is most commonly used to represent integral types of known, fixed bit-width.

Example: -

```
#include <stdio.h>


    typedef union {
        struct {
            signed int immed:16;
            unsigned int rt:5;
            unsigned int rs:5;
            unsigned int opcode:6;
        };
        unsigned int w;
    }
   I_format_t;



    int main()
    {
        I_format_t ia;
     ia.w = 0xAFBE0010;
        printf("\ninstruction: %X\n",ia.w);
        printf("opcode: %X\n",ia.opcode);
```

```
    printf("rs: %d rt: %d\n", ia.rs, ia.rt);
    printf("immed: %d\n",ia.immed);

    return 0;

}
```

Q12) How can bit field be used within a structure declaration?

Ans:- Both C and C++ allow integer members to be stored into memory spaces smaller than the compiler would ordinarily allow. These space-saving structure members are called

bit fields

, and their width in bits can be explicitly declared. Gagandeep Singh Bitfields can only be declared inside a structure or a union, and allow you to specify some very small objects of a given number of bits in length. struct { /* field 4 bits wide */ unsigned field1 :4; /* * unnamed 3 bit field * unnamed fields allow for padding */ unsigned :3; /* * one-bit field * can only be 0 or -1 in two's complement! */ signed field2 :1; /* align next field on a storage unit */ unsigned :0; unsigned field3 :6; }full_of_fields; The main use of bitfields is either to allow tight packing of data or to be able to specify the fields within some externally produced data files.

Q 13) How can a bit field be used within union data type?

Ans:- You are given a gun and bullets. Is it okay to shoot your self in foot with it? Of course not, but nobody can stop you from doing this if you want to.

My point is, just like gun and bullets, union and bit fields are tools and they have their purpose, uses and "abuses". So using bitfields in union, as you have written above, is perfectly valid C but a useless piece of code. All the fields inside union share same memory so all the bitfields you mention are essentially same flag as they share same memory.

Q 14) Is the structure tag required? Give an example of structure with a tag.

Ans:-

Q 15) What is file? Explain the primary purpose of a file.

Ans:- A file represents a sequence of bytes on the disk where a group of related data is stored. File is created for permanent storage of data. It is a ready made structure

Purpose

The purpose of the file extension is to identify the type of file. The computer uses this information to open the file with the correct software application

Q 16) What is a Random access file? Explain its advantages.

Ans:- Random access files consist of records that can be accessed in any sequence. This means the data is stored exactly as it appears in memory, thus saving processing time (because no translation is necessary) both in when the file is written and in when it is read.

Random files are a better solution to database problems than sequential files, although there are a few disadvantages. For one thing, random files are not especially transportable. Unlike sequential files, you cannot peek inside them with an editor, or type them in a meaningful way to the screen. In fact, moving a PowerBASIC random file to another computer or language will probably require that you write a translator program to read the random file and output a text (sequential) file.

Sequential File

A computer program makes a sequential file simply by writing data records, one after the other, into a newly created file area

Speed

Compared to direct-access files, programs process sequential access files faster.

Simplicity

Sequential files are easy to read because of their simple organization. It is a simple matter to write new programs to read existing sequential files, since the program reads the records as a simple series until it encounters an end-of-file (EOF) mark

Data Sharing

Programs which share data use the sequential access file format as a "common language." For example, a database program may export a file for a spreadsheet

Q 17) How do you search data in a data file using random access method?

Ans:-

Q 18) What is data file? State any tow advantages of it.

Ans:- A Data file is a computer file which stores data to be used by a computer application or system. It generally does not refer to files that contain instructions or code to be executed (typically called program files), or to files which define the operation or structure of an application or system (which include configuration files, etc.); but specifically to information used as input, or written as output by some other software program. This is especially helpful when debugging a program

Q 19) What is different between the getchar() and getch() function in c.

Ans:- Firstly, getch() are nonstandard library functions found in conio.h. Hence, they are useless and should never be used (nor the Turbo C/C++ compilers).

The difference between getchar() and getc(FILE *stream) is that, getc can read input from any stream, while getchar can only read from the standard input. Thus you can say that

getchar() = getc(stdin)

Q 20) Differentiate putchar() and putch() ?

Ans:- int putchar(char c); writes the character c to stdout. It is equivalent to fputc(c, stdout). You can't use putchar to output to arbitrary files; it always outputs to stdout.

putc is nearly identical to fputc, except that putc is allowed to be a macro. I haven't encountered putc in much code

Q 21) How prinf() is differ from fprintf()?

Ans:- printf()

printf, buy default, will send its formatted result to the console window/display. (You can override this behavior by *redirecting* the program's output to a file or device either outside the program from the command line or inside the program using the freopen function.)

`printf` is equivalent to writing `fprintf(stdout, ...)` and writes formatted text to wherever the standard output stream is currently pointing

fprintf()

fprintf is just like printf, except that it sends its formatted result to a specified file. fprintf has an extra first parameter, which is the file pointer (FILE *) of the file to which you want the formatted result sent. Typically, you would use the fopen function to open a file for writing or appending, and then pass to fprintf the file pointer you got back from the call to fopen. (If you specify stdout in that first parameter to fprintf, it will behave like printf.)

`fprintf` writes formatted text to the output stream you specify

Q 22) State fgetc() and fputc() file function with suitable example.

Ans:- The fputc() function

The fputc() function is used to write characters to the file.

Syntax of fputc() function

```
fputc(char ch, FILE *fp);
```

The fputc() function takes two arguments, first is the character to be written to the file and second is the file pointer where the character will be written.

Example of fputc() function

```
#include<stdio.h>

void main()
{
    FILE *fp;
    char ch;

    fp = fopen("file.txt","w");          //Statement   1

    if(fp == NULL)
    {
        printf("\nCan't open file or file doesn't exist.");
        exit(0);
    }

    while((ch=getchar())!=EOF)         //Statement   2
      fputc(ch,fp);

    printf("\nData written successfully...");

    fclose(fp);
}
```

Output :

```
    Hello friends, my name is kumar.^Z
    Data written successfully...
```

The fgetc() function

The fgetc() function is used to read characters form the file.

Syntax of fgetc() function

```
char fgetc(FILE *fp);
```

The fgetc() function takes the file pointer indicates the file to read from and returns the character read from the file or returns the end-of-file character if it has reached the end of file.

Example of fgetc() function

```
#include<stdio.h>

void main()
{
    FILE *fp;
    char ch;

    fp = fopen("file.txt","r");          //Statement   1

    if(fp == NULL)
    {
        printf("\nCan't open file or file doesn't exist.");
        exit(0);
    }

    printf("\nData in file...\n");

    while((ch = fgetc(fp))!=EOF)         //Statement   2
       printf("%c",ch);

    fclose(fp);
}
```

Output :

Data in file...

Hello friends, my name is kumar.

Q 23) State the salient feature of a typedef.

Ans:-

Q 24)state fwrite () function with its syntax and example.

Ans:- The fwrite() function is used to write records (sequence of bytes) to the file. A record may be an array or a structure

Syntax:- fwrite(ptr, int size, int n, FILE *fp);

Example:- include<stdio.h>

```
struct Student
{
    int roll;
    char name[25];
    float marks;
};

void main()
{
    FILE *fp;
    char ch;
    struct Student Stu;

    fp = fopen("Student.dat","w");          //Statement   1

    if(fp == NULL)
    {
        printf("\nCan't open file or file doesn't exist.");
        exit(0);
```

```
        }

        do
        {
            printf("\nEnter Roll : ");
            scanf("%d",&Stu.roll);

            printf("Enter Name : ");
            scanf("%s",Stu.name);

            printf("Enter Marks : ");
            scanf("%f",&Stu.marks);

            fwrite(&Stu,sizeof(Stu),1,fp);

            printf("\nDo you want to add another data (y/n) : ");
            ch = getche();

        }while(ch=='y' || ch=='Y');

        printf("\nData written successfully...");

        fclose(fp);
    }
```

Output :

```
        Enter Roll : 1
        Enter Name : Ashish
        Enter Marks : 78.53

        Do you want to add another data (y/n) : y
```

Enter Roll : 2

Enter Name : Kaushal

Enter Marks : 72.65

Do you want to add another data (y/n) : y

Enter Roll : 3

Enter Name : Vishwas

Enter Marks : 82.65

Do you want to add another data (y/n) : n

Data written successfully...

Q 25) what is the difference between the fread() and fscanf() function

Ans:- * **fread :**

1. This Function is Used in **Binary Mode.**
2. Function **Reads Block of Data from Binary Mode File and Assign it to the Region of Memory Specified**.[i.e Reads Block of Data From Binary File and Assign it to Some Memory]
3. **Returns :** Total number of Values Read

Syntax_:- size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)

***Fscanf():**

The C library function **int fscanf(FILE *stream, const char *format, ...)** reads formatted input from a stream.

Syntax:- int fscanf(FILE *stream, const char *format,

Unit IV[5 marks]

1) **What is a structure and what are the uses of it ?**

Ans:- Structures in C. A **structure** is a collection of variables under a single name. These variables can be of different types, and each has a name which is used to select it from the **structure**. A **structure** is a convenient way of grouping several pieces of related information together

There are two types of operators used for accessing members of a structure.

1. Member operator(.)
2. Structure pointer operator(->) (is discussed in structure and pointers tutorial)

Any member of a structure can be accessed as:

```
structure_variable_name.member_name
```

Suppose, we want to access salary for variable `person2`. Then, it can be accessed as:

```
person2.salary
```

2) **how are the data element of a structured accessed and processed?**