V.S.B. ENGINEERING COLLEGE, KARUR.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR 2017 -2018(EVEN SEMESTER)

QUESTION BANK (2017 Regulation)

CS8251 - PROGRAMMING IN C

UNIT I
BASICS OF C PROGRAMMING

1. What are the different programming paradigms?

- Imperative: Programming with an explicit sequence of commands that update state.
- Declarative: Programming by specifying the result you want, not how to get it.
- Structured: Programming with clean, goto-free, nested control structures.
- Procedural: Imperative programming with procedure calls.
- Functional (Applicative): Programming with function calls that avoid any global state.
- Function-Level (Combinator): Programming with no variables at all.
- Object-Oriented: Programming by defining objects that send messages to each other.

  Objects have their own internal (encapsulated) state and public interfaces. Object orientation can be:
  - Class-based: Objects get state and behavior based on membership in a class.
  - Prototype-based: Objects get behavior from a prototype object.

- Event-Driven: Programming with emitters and listeners of asynchronous actions.
- Flow-Driven: Programming processes communicating with each other over predefined channels.
- Logic (Rule-based): Programming by specifying a set of facts and rules. An engine infers the answers to questions.
- Constraint: Programming by specifying a set of constraints. An engine finds the values that meet the constraints.
- Aspect-Oriented: Programming cross-cutting concerns applied transparently.
- Reflective: Programming by manipulating the program elements themselves.
- Array: Programming with powerful array operators that usually make loops unnecessary.

2. Define programming paradigm.

Programming paradigms are a way to classify programming languages based on their features. Languages can be classified into multiple paradigms.

Some paradigms are concerned mainly with implications for the execution model of the language, such as allowing side effects, or whether the sequence of operations is defined by the execution model. Other paradigms are concerned mainly with the way that code is

organized, such as grouping a code into units along with the state that is modified by the code. Yet others are concerned mainly with the style of syntax and grammar.

3.What is C language?

The programming language C was developed in the early 1970s by Dennis Ritchie at Bell Laboratories to be used by the UNIX operating system. It was named 'C' because many of its features were derived from an earlier language called 'B'.

4.Uses of C.

- C language is primarily used for system programming. The portability, efficiency, the ability to access specific hardware addresses and low runtime demand on system resources makes it a good choice for implementing operating systems and embedded system applications.
- C has been so widely accepted by professionals that compilers, libraries and interpreters of other programming languages are often implemented in C.
- C is widely used to implement end-user applications.

5.Why is C language is Popular?

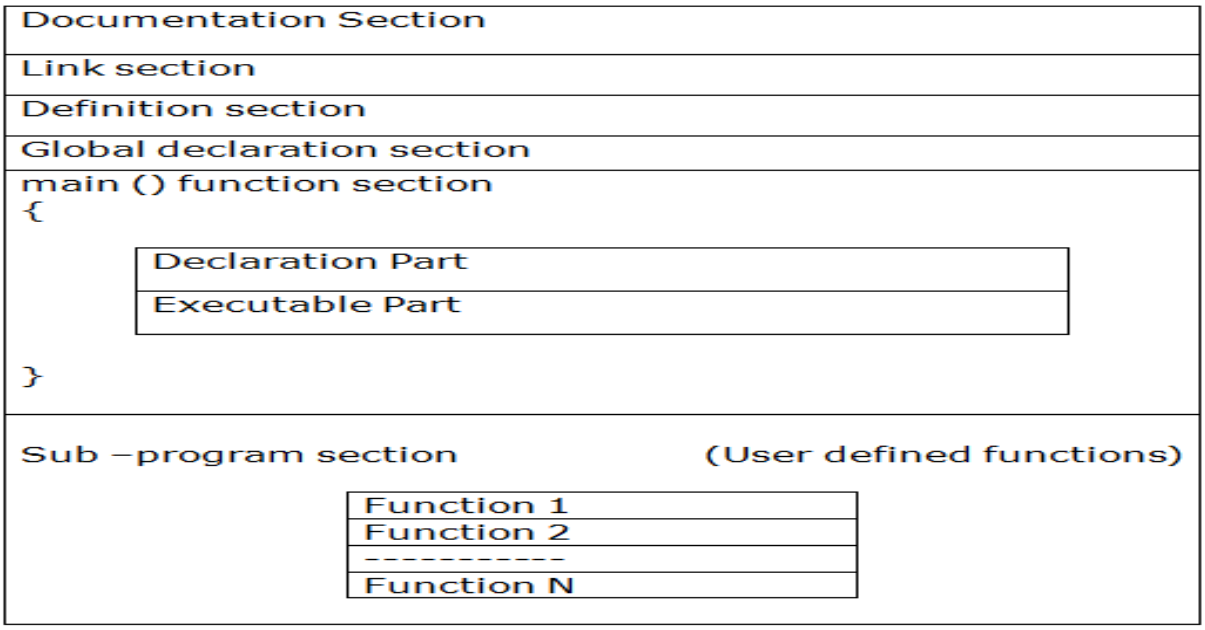
- C is a robust language.
- Programs written in C can be reused.
- Program written in C are fast and efficient.
- It is machine independent and highly portable language.
- C language allows manipulation of bits, bytes and addresses.
- C is relatively small programming language. C has only 32 keywords.
- C encourages users to write additional library function of their own.
- Many other high level languages have been developed based on C. For example C++ and PERL.

6.Structure of a C Program.

Documentation Section :- The documentation section usually consists of a set of comments lines demonstrating Name of the program, Name of the author and other details about the program. Comments are non-executable statement of a C program. Comments can be of single line or multi line.
For single line comments :- Use two slashes (//) begining of a comment line.
For multi line comments :- Use a slash and an asterisk (/*) and ends with */.
Example :-

```
// Program Name is : hello_world.c

/* This program prints hello world on screen
```



Link section :- The link section provides instructions to the compiler to link functions from the system library such as using the #include directive.
Example :-

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

- Definition Section :- Sometime we require some fix values in the entire program, called as constants. The definition section defines all these symbolic constants.
  Example :-

  ```
  #define pi 3.14
  ```

- Global declaration section :- Some variables are required in more than one function. These variable has to defined outside the main function, hence as global variable. Global declaration section defines these variables.
  Example :-

  ```
  int globalX;
  ```

We will discuss more about variable, data types and their scope in upcoming sections.

- main () function section :- Every C program must have one main() function section. As the execution of the program always begins with the first executable statement of the main function. This section contains two parts; declaration part and executable part.
  - Declaration part declares all the variable used in the executable part.
  - Executable part consists of at least one statement and these two parts must appear between the opening and the closing braces.

Example :-

```
void main()
    {
    int a=10; // declaration part
    printf("%d",a); // executable part
    }
```

7. Enlist Keywords in C.

C has a set of reserved words often known as keywords. These cannot be used as an identifier. All keywords are basically a sequence of characters that have a fixed meaning. By convention all keywords must be written in lowercase letters. following table contains a list of keywords in C.

auto	break	case	char	const	continue	default
double	else	enum	extern	float	for	goto
int	long	register	return	short	signed	sizeof
struct	switch	typedef	union	unsigned	void	volatile
do	if	static	while			

When you complete this tutorial, the meaning and utility of each keyword will automatically become clear.

8.Write short notes on Compilation and Linking of C program.

Each high-level language needs a compiler or interpreter to translate instructions written in the HLL in to machine language that a computer can understand and execute.

During the execution of C program, three process are involved. They are :-

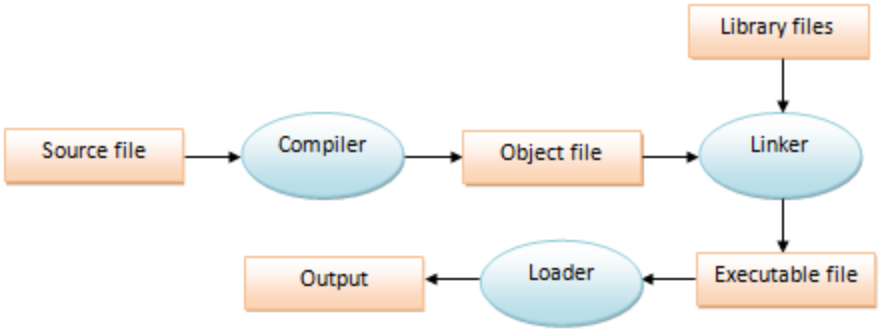
- Compilation - Carried out by Compiler.
- Linking - Carried out by Linker.
- Loading - Carried out by Loader.

Here we will discuss each process briefly.

- Compilation :- Compilation refers to the processing of source code files (.c, .cpp) and the creation of an object file. This step doesn't create anything the user can actually run. The object code contains the machine instructions for the CPU and calls to the operating system application programming interface (API). Object file has .obj extension.
- Linking :- Linking refers to the creation of a single executable file (.exe) from multiple object files.

- Loading :- Loading is the process of put the executable file (.exe) into memory prior to execution.

The complete compilation and execution process is shown in following figure.



- First a program written in c, called source code and the file is known as source file. It has .c extension.
- Then the source code is compiled by a Compiler. If there exists some syntax error(s), it won't generate the object code. Go to editor and rectify the error(s). Otherwise, it create a new file, called as object file (.obj).
- Then a special program called linker is involved to link the object file with some library files and other object files. The result of linking is saved into a new file called as an executable file (.exe).
- Now this executable file is loaded into primary memory for execution by a special component of OS known as Loader.

9. State basic tasks of Compiler, Linker and Loader

Basic tasks of compiler

- It translates one language to another.
- Takes source program as input and produces an equivalent target program typically in assembly or machine language.
- It generates object files.
- Reports error messages as part of the translation process.

Basic tasks of Linker

- It combines object modules to form a executable file.
- It searches the program to find library routines used by program , eg. printf(), math routines etc.
- Determines the memory location that code from each module will occupy and relocates its instructions by adjusting absolute references.
- Resolves references among files.

Basic tasks of Loader

- Bring an executable file residing on disk into memory and start it running.
- It creates new address space for the program before executing.
- Copies instructions and data into address space.

- Copies arguments passed to the program on the stack.

10. What is error?

An error is anything in our code that prevent a program from running correctly. In the computer world, program errors are also called bugs and the art of pinpointing a bug and fixing it is called debugging. When all the bugs are out, the program is correct.

11. Enlist types of error.

Errors can be broadly classified into following types :-

- Compile time errors
- Linker errors
- Runtime errors
- Logical errors

12. What is Data type?

Data types are means to identify the type of data along with set of rules for permissible operations. The data type of a variable determines how much space it occupies in storage (memory) and how the bit pattern stored is interpreted.

13. Classify Data types in C.

The two major aspects of data in C are data type and storage class . C language is rich in its data types. It provides a predefined set of data types for handling the data of different types such as character, integer, real, string etc. Data in C belongs to one of the following types :-

- Fundamental types
- Derived types

Fundamental Data Types

C language provides very few basic data types. Following table lists the data type, their size, range, and usage.

Data type	Keyword Used	Size in bytes	Range	Use
Character	char	1	-128 to 127	To store characters
Integer	int	2	-32768 to 32767	To store integer numbers

Floating point	float	4	3.4E-38 to 3.4E+38	To store floating point numbers
Double	double	8	1.7E-308 to 1.7E+308	To store big floating point numbers
Valueless	void	0	Valueless	Used in functions and pointers

Derived Types

The data type that has been constructed from the fundamental data types is called as derived data type. The derived data types in C classified into two groups. They are :-

- Built-in types
- User-defined data types

Following is the list of built-in derived data types in C.

1. Arrays
2. Pointers
3. Structure
4. Union

There are some derived data types that are defined by the user. These are called as user defined derived data types. These are :-

1. typedef
2. enum

14. Define Compilation process.

Compilation refers to the processing of source code files (.c, .cc, or .cpp) and the creation of an 'object' file. This step doesn't create anything the user can actually run. Instead, the compiler merely produces the machine language instructions that correspond to the source code file that was compiled.

15. What do you meant by linking?

Linking refers to the creation of a single executable file from multiple object files. In this step, it is common that the linker will complain about undefined functions (commonly, main itself). During compilation, if the compiler could not find the definition for a particular function, it would just assume that the function was defined in another file. If this isn't the case, there's no way the compiler would know it doesn't look at the contents of more than one file at a time. The linker, on the other hand, may look at multiple files and try to find references for the functions that weren't mentioned.

16. Define Constants in C. Mention the types.

The constants refer to fixed values that the program may not alter during its

execution. These fixed values are also called literals.

Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string literal. There are also enumeration constants as well.

The constants are treated just like regular variables except that their values cannot be modified after their definition.

17. What are the different data types available in 'C'?

There are four basic data types available in 'C'.

a. int
b. float
c. char
d. double

18. What is meant by Enumerated data type.

Enumerated data is a user defined data type in C language.

Enumerated data type variables can only assume values which have been previously declared.

Example :

enum month { jan = 1, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec };

19. What are Keywords?

Keywords are certain reserved words that have standard and pre-defined meaning in 'C'. These keywords can be used only for their intended purpose.

20. What do you mean by variables in 'C'?

A variable is a data name used for storing a data value. It can be assigned different values at different times during program execution.

It can be chosen by programmer in a meaningful way so as to reflect its function in the program.

21. Difference between Local and Global variable in C.

Local

These variables only exist inside the specific function that creates them. They are unknown to other functions and to the main program. As such, they are normally implemented using a stack. Local variables cease to exist once the function that created them is completed. They are recreated each time a function is executed or called.

Global

These variables can be accessed (ie known) by any function comprising the program. They are implemented by associating memory locations with variable names. They do not get recreated if the function is recalled.

22. What are Operators? Mention their types in C.

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C language is rich in built-in operators and provides following type of

operators:

a. Arithmetic Operators
b. Relational Operators
c. Logical Operators
d. Bitwise Operators
e. Assignment Operators
f. Misc Operators

23. What is the difference between '=' and '==' operator?

Where = is an assignment operator and == is a relational operator.

Example:

while (i=5) is an infinite loop because it is a non zero value and while (i==5) is true only when i=5.

24. What is type casting?

Type casting is the process of converting the value of an expression to a particular data type.

Example:

int x,y;

c = (float) x/y; where a and y are defined as integers. Then the result of x/y is converted into float.

25.What is the difference between ++a and a++?

++a means do the increment before the operation (pre increment)
a++ means do the increment after the operation (post increment)

Example:

```
a=5;
x=a++; /* assign x=5*/ y=a;
/*now y assigns y=6*/
x=++a; /*assigns x=7*/
```

26. Distinguish between while..do and do..while statement in C.

While	DO. while
(i) Executes the statements within the while block if only the condition is true	(i) Executes the statements within thwhile block at least once.
(ii) The condition is checked at the starting of the loop	(ii) The condition is checked at the end of the loop

27. Mention the various Decisions making statement available in C.

Statement	Description
if statement	An **if statement** consists of a boolean expression followed by one or more statements.

if...else statement	An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is false.
nested if statements	You can use one **if** or **else if** statement inside another **if** or **else if** statement(s).
switch statement	A **switch** statement allows a variable to be tested for equality against a list of values.

28. What do you meant by conditional or ternary operator?

? : If Condition is true ? Then value X : Otherwise value Y

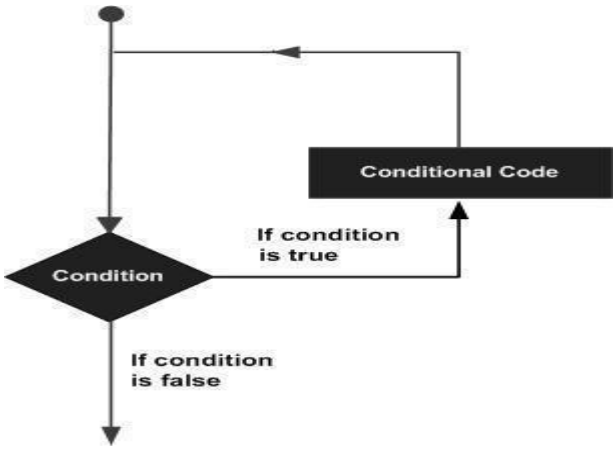
29. What is the use of sizeof() operator in C.

Sizeof operator is used to return the size of a variable.

Example: sizeof(a), Where a integer, will return 4.

30. Define Looping in C.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general from of a loop statement in most of the programming languages:



31. What are the types of I/O statements available in 'C'?

There are two types of I/O statements available in 'C'.

1. Formatted I/O Statements
2. Unformatted I/O Statements

32. Write short notes about main ( ) function in 'C' program. (MAY 2009)

1. Every C program must have main ( ) function.
2. All functions in C, has to end with '( )' parenthesis.
3. It is a starting point of all 'C' programs.
4. The program execution starts from the opening brace '{' and ends with closing brace '}', within which executable part of the program exists.

33. Define delimiters in 'C'.

Delimiters Use

: Colon

; Semicolon

( ) Parenthesis
[ ] Square Bracket
{ } Curly Brace
Hash
, Comma

34. Why header files are included in 'C' programming?

This section is used to include the function definitions used in the program. · Each header file has 'h' extension and include using '# include' directive at the beginning of a program.

35. What is the difference between scanf() and gets() function?

In scanf() when there is a blank was typed, the scanf() assumes that it is an end. gets() assumes the enter key as end. That is gets() gets a new line (\n) terminated string of characters from the keyboard and replaces the '\n' with '\0'.

36. What is the difference between if and while statement?

If	**While**
(i) It is a conditional statement	(i) It is a loop control statement
(ii) If the condition is true, it executes some statements.	(ii) Executes the statements within the while block if the condition is true.
(iii) If the condition is false then it stops the execution the statements.	(iii) If the condition is false the control is transferred to the next statement of the loop.

37.Differentiate between formatted and unformatted you input and output functions?

Formatted I/P functions:

These functions allow us to supply the input in a fixed format and let us obtain the output in the specified form. Formatted output converts the internal binary representation of the data to ASCII characters which are written to the output file.

Unformatted I/O functions:

There are several standard library functions available under this category-those that can deal with a string of characters. Unformatted Input/Output is the most basic form of input/output. Unformatted input/output transfers the internal binary representation of the data directly between memory and the file.

38. What are the pre-processor directives?

1. Macro Inclusion
2. Conditional Inclusion
3. File Inclusion

39. What are conditional Inclusions in Preprocessor Directive?

Conditional inclusions (#ifdef, #ifndef, #if, #endif, #else and #elif)

These directives allow including or discarding part of the code of a program if a certain condition is met. #ifdef allows a section of a program to be compiled only if the macro that is specified as the parameter has been defined, no matter which its value is.

For example:

```
1 #ifdef TABLE_SIZE
2 int table[TABLE_SIZE];
3 #endif
```

40. What you meant by Source file Inclusion in Preprocessor directive?

Source file inclusion (#include)

This directive has also been used assiduously in other sections of this tutorial. When the preprocessor finds an #include directive it replaces it by the entire content of the specified file. There are two ways to specify a file to be included:

1 *#include "file"*

2 *#include <file>*

UNIT II
ARRAYS AND STRINGS

1. What is an array?

An array is a group of similar data types stored under a common name. 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.

Example:

int a[10];

Here a[10] is an array with 10 values.

2. What are the main elements of an array declaration?

1. Array name
2. Type
3. Size

3. How to initialize an array?

You can initialize array in C either one by one or using a single statement as follows:

double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};

The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ]. Following is an example to assign a single element of the array:

4. Why is it necessary to give the size of an array in an array declaration?

When an array is declared, the compiler allocates a base address and reserves enough space in the memory for all the elements of the array. The size is required to allocate the required space. Thus, the size must be mentioned.

5. What is the difference between an array and pointer?

Difference between arrays and pointers are as follows.

Array	**Pointer**
Array allocates space automatically	1.Pointer is explicitly assigned to point to an allocated space.
2.It cannot be resized.	2.It can be resized using realloc ().
3.It cannot be reassigned.	3.Pointers can be reassigned.
4.Size of(array name) gives the number of bytes occupied by the array.	4.Sezeof(pointer name) returns the number of bytes used to store the pointer variable.

6. List the characteristics of Arrays.

All elements of an array share the same name, and they are distinguished form one another with help of an element number. Any particular element of an array can be modified separately without disturbing other elements.

7. What are the types of Arrays?

1. 1.One-Dimensional Array
2. Two-Dimensional Array
3. Multi-Dimensional Array

8. Define Strings.

Strings:

The group of characters, digit and symbols enclosed within quotes is called as Stirng (or) character

Arrays. Strings are always terminated with '\0' (NULL) character. The compiler automatically adds '\0' at the end of the strings.

Example:

char name[]={'C','O','L','L','E','G','E','E','\0'};

9. Mention the various String Manipulation Functions in C.

S.NO.	FUNCTION & PURPOSE
strcpy(s1, s2);	Copies string s2 into string s1.
strcat(s1, s2);	Concatenates string s2 onto the end of string s1.
strlen(s1);	Returns the length of string s1.
strcmp(s1, s2);	Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
strchr(s1, ch);	Returns a pointer to the first occurrence of character ch in string s1.
strstr(s1, s2);	Returns a pointer to the first occurrence of string s2 in string s1.

10. What is the use of atoi() function?

C allows us to manipulate characters the same way we do with numbers. Whenever a character constant or character variable is used in an expression, it is automatically converted into integer value by the system.

For eg, if the machine uses the ASCII representation, then,

x = 'a'; printf("%d \n",x);

will display the number 97 on the screen.

The C library supports a function that converts a string of digits into their integer values. The function takes the form

11. Write syntax and example for two dimensional array in c?

Synatx: data-type arr_name[num_of_rows][num_of_col];

Array declaration, initialization and accessing	Example

Array declaration syntax:
data_type arr_name
[num_of_rows][num_of_column];
Array initialization syntax:
data_type arr_name[2][2] =
{{0,0},{0,1},{1,0},{1,1}};
Array accessing syntax:
arr_name[index];
	Integer array example:
	int arr[2][2];
	int arr[2][2] = {1,2, 3, 4};
	
	arr [0] [0] = 1;
	arr [0]]1] = 2;
	arr [1][0] = 3;
	arr [1] [1] = 4;

12. What is meant by Sorting?

Sorting refers to ordering data in an increasing or decreasing fashion according to some linear relationship among the data items. Sorting can be done on names, numbers and records.

13. What are the types of sorting available in C?

a. Insertion sort.
b. Merge Sort.
c. Quick Sort.
d. Radix Sort**.**
e. Heap Sort
f. Selection sort
g. Bubble sort

14. Define Heap Sort.

A sorting algorithm that works by first organizing the data to be sorted into a special type of binary tree called a heap. The heap itself has, by definition, the largest value at the top of the tree, so the heap sort algorithm must also reverse the order. It does this with the following steps:

1. Remove the topmost item (the largest) and replace it with the rightmost leaf. The topmost item is stored in an array.
2. Re-establish the heap.
3. Repeat steps 1 and 2 until there are no more items left in the heap.

15. Define Bubble sort.

A simple but popular sorting algorithm. Bubble sorting is used frequently as a programming exercise because it is relatively easy to understand. It is not, however, particularly efficient. Other sorting algorithms, such as heap sorts, merge sorts and quick sorts, are used more often in real applications.

16. Define Searching.

Searching for data is one of the fundamental fields of computing. Often, the difference between a fast program and a slow one is the use of a good algorithm for the data set. Naturally, the use of a hash table or binary search tree will result in more efficient searching, but more often than not an array or linked list will be used. It is necessary to understand good ways of searching data structures not designed to support efficient search.

17. Mention the various types of searching techniques in C

1. Linear search
2. Binary search

18. What is linear search?

In Linear Search the list is searched sequentially and the position is returned if the key element to be searched is available in the list, otherwise -1 is returned. The search in Linear Search starts at the beginning of an array and move to the end, testing for a match at each item.

19. What is Binary search?

A binary search, also called a dichotomizing search, is a digital scheme for locating a specific object in a large set. Each object in the set is given a key. The number of keys is always a power of 2. If there are 32 items in a list, for example, they might be numbered 0 through 31 (binary 00000 through 11111). If there are, say, only 29 items, they can be numbered 0 through 28 (binary 00000 through 11100), with the numbers 29 through31 (binary 11101, 11110, and 11111) as dummy keys.

20. Write syntax and example for one dimensional array in c?

Synatx: data-type arr_name[array_size];

<table>
<tr><th>Array declaration, initialization and accessing</th><th>Example</th></tr>
<tr><td rowspan="2">Array declaration syntax:
data_type arr_name [arr_size];
Array initialization syntax:
data_type arr_name [arr_size]=(value1, value2, value3,….);
Array accessing syntax:
arr_name[index];</td><td>Integer array example:
int age [5];
int age[5]={0, 1, 2, 3, 4};</td></tr>
<tr><td>Character array example:
char str[10];
char str[10]={‘H’,‘a’,‘i’};
(or)
char str[0] = ‘H’;
char str[1] = ‘a’;
char str[2] = ‘i;</td></tr>
</table>

UNIT III
FUNCTIONS AND POINTERS

1. Write definition of function. Indicate types of functions available in C.

A function is a self-contained block or a sub-program of one or more statements that performs a special task when called.

- Without arguments or return values. Eg. abc()
- With arguments but without return values. Eg. abc (int x)
- With arguments and return values. Eg. int abc(int x)
- Without argument but with return values. Eg. int abc().

2. Write the difference between pre-defined (library) function and user defined function.

Library(Pre-defined) function	User defined function
Contains Pre-defined set of functions	The user defined the functions
User cannot understand the internal working	User can understand internal working
Source code is not visible	Source code is visible
User cannot modify the function	User can modify the function

3. Write the Syntax of function call.
Syntax of function call

functionName(argument1, argument2, ...);

function call is made using addNumbers(n1,n2); statement inside the main().

4. What are the advantages of unions over structures?

- Union save memory space as the size of a union variable is equal to its largest sized member. In contrast, the size of a structure variable is equal to the sum of the sizes of all its individual member elements.
- Unions are particularly useful in situation where there is a need to use only one of its member elements at any given point of time.

5. What is a pointer?

A pointer is a memory variable that stores a memory address. It can have any name that is legal for another variable and it is declared in the same fashion like other variables but it is always denoted by pre fixing '*' operator.

Pointer Declaration: datatype * variable-name;

Example: int *x, c=5; x=&a;

6. Write the features of a pointer.

Features:

- Pointers save memory space.
- Execution time with the pointer is faster because data is manipulated with the address, direct access to memory location.
- The memory is accessed efficiently with the pointers.
- Pointers are used with data structures.
- We can access elements of any type of array irrespective if its subscript range.
- Pointers are used in file handling.

7. What are the uses of Pointers?

- Pointers are used to return more than one value to the function
- Pointers are more efficient in handling the data in arrays
- Pointers reduce the length and complexity of the program
- They increase the execution speed
- The pointers save data storage space in memory.

8. Write the syntax for function definition.

```
returnType functionName(type1 argument1, type2 argument2, ...)

{

body of the function

}
```

9. Differentiate call by value and call by reference.

Call by value	Call by reference
Only values of the variables are passed as parameters	Pointer or reference to the variable is passed as parameter.
Any change made to the value will not reflected in the variable storage location	Any change made to the variables gets reflected at the original location of the variable

10. What is the difference between Array and Pointer?

Array	Pointer
Array allocates space automatically	Pointer is explicitly assigned to point to an allocated space
It cannot be resized	It can be resized using realloc()
It cannot be reassigned	It can be reassigned
Size of array name gives the number of bytes occupied by the array	Size of pointer name returns the number of pointer used to store the pointer variable.

11. What are the various dynamic memory allocation functions?

- malloc() - Used to allocate blocks of memory in required size of bytes.
- free () - Used to release previously allocated memory space.
- calloc() - Used to allocate memory space for an array of elements.
- realloac() - Used to modify the size of the previously allocated memory space.

12. Write a C program for factorial using recursion function.

Following is an example which calculates factorial for a given number using a recursive function:

```
#include <stdio.h>

int factorial(unsigned int i)
{
if(i <= 1)

{
return 1;

}
return i * factorial(i - 1);

}

int       main()
{

int i = 15;

printf("Factorial of %d is %d\n", i, factorial(i)); return 0;

}
```

When the above code is compiled and executed, it produces the following result:
Factorial of 15 is 2004310016

13. State the advantages of user defined functions over pre-defined functions.

- A user defined function allows the programmer to define the exact function of the module as per requirement. This may not be the case with predefined function. It may or may not serve the desired purpose completely.

- A user defined function gives flexibility to the programmer to use optimal programming instructions, which is not possible in predefined function.

14. Write the syntax for pointers to structure.

```
Struct S

{

char datatype1;

int datatype2;

float datatype3;

};

Struct S *sptr //sptr ia pointer to structure S
```

15. Write the advantages and disadvantages of recursion.

Recursion makes program elegant and cleaner. All algorithms can be defined recursively which makes it easier to visualize and prove.

If the speed of the program is vital then, you should avoid using recursion. Recursions use more memory and are generally slow. Instead, you can use loop.

Check out these examples to learn more:

- Find the Sum of Natural Numbers using Recursion
- Find Factorial of a Number Using Recursion
- Find G.C.D Using Recursion

16. What is meant by Recursive function?

If a function calls itself again and again, then that function is called Recursive function.

Example:

```
void recursion()
{

recursion(); /* function calls itself */
```

```
}

int main()
{

recursion();
}
```

17. Is it better to use a macro or a function?

Macros are more efficient (and faster) than function, because their corresponding code is inserted directly at the point where the macro is called. There is no overhead involved in using a macro like there is in placing a call to a function. However, macros are generally small and cannot handle large, complex coding constructs. In cases where large, complex constructs are to handled, functions are more suited, additionally; macros are expanded inline, which means that the code is replicated for each occurrence of a macro.

18. List any five-library functions.

- ceil(x)
- sqrt(x)
- log(x)
- pow(x,y)
- sin(x)

19. List the header files in 'C' language.

- <stdio.h> contains standard I/O functions
- <ctype.h> contains character handling functions
- <stdlib.h> contains general utility functions
- <string.h> contains string manipulation functions
- <math.h> contains mathematical functions
- <time.h> contains time manipulation functions

20. What are the steps in writing a function in a program?

- **Function Declaration (Prototype declaration):**
  Every user-defined functions has to be declared before the main().

- **Function Callings:**
  The user-defined functions can be called inside any functions like main(), user-defined function, etc.

- **Function Definition:**
  The function definition block is used to define the user-defined functions with statements.

UNIT IV
STRUCTURES

1. Compare arrays and structures.

Comparison of arrays and structures is as follows.

Arrays	**Structures**
An array is a collection of data items of same data type. Arrays can only be declared.	A structure is a collection of data items of different data types. Structures can be declared and defined.
There is no keyword for arrays.	The keyword for structures is struct.
An array cannot have bit fields.	A structure may contain bit fields.
An array name represents the address of the starting element.	A structure name is known as tag. It is a Shorthand notation of the declaration.

2. Compare structures and unions.

Structure	**Union**
Every member has its own memory.	All members use the same memory.
The keyword used is struct.	The keyword used is union.
All members occupy separate memory location, hence different interpretations of the same memory location are not possible. Consumes more space compared to union.	Different interpretations for the same memory location are possible. Conservation of memory is possible

3. Define Structure in C.

C Structure is a collection of different data types which are grouped together and each element in a C structure is called member.

If you want to access structure members in C, structure variable should be declared.

Many structure variables can be declared for same structure and memory will be allocated for each separately.

It is a best practice to initialize a structure to null while declaring, if we don't assign any values to structure members.

4. What you meant by structure definition?

A structure type is usually defined near to the start of a file using a typedef statement. typedef defines and names a new type, allowing its use throughout the program. typedefs usually occur just after the #define and #include statements in a file.

Here is an example structure definition.

```
typedef struct { char
    name[64];
    char course[128];
    int age;
```

```
    int year;
} student;
```

This defines a new type student variables of type student can be declared as follows.

```
student st_rec;
```

5. How to Declare a members in Structure?

A **struct** in C programming language is a structured (record) type[1] that aggregates a fixed set of labeled objects, possibly of different types, into a single object. The syntax for a struct declaration in C is:

```
struct tag_name
{
  type
  attribute;
  type
  attribute2;
  /* ... */
};
```

6. What is meant by Union in C?

A **union** is a special data type available in C that enables you to store different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multi-purpose.

7. How to define a union in C.

To define a union, you must use the **union** statement in very similar was as you did while defining structure. The union statement defines a new data type, with more than one member for your program. The format of the union statement is as follows:

```
union [union tag]

{
  member
  definition;
  member
  definition;
  ...
  member definition;
} [one or more union variables];
```

8. What are storage classes?

A storage class defines the scope (visibility) and life time of variables and/or functions within a C Program.

9. What are the storage classes available in C?

There are following storage classes which can be used in a C Program

1. auto
2. register
3. static
4. extern

10. What is register storage in storage class?

Register is used to define local variables that should be stored in a register instead of

RAM. This means that the variable has a maximum size equal to the register size (usually one word) and cant have the unary '&' operator applied to it (as it does not have a memory location).

```
    {
register int Miles;
    }
```

11.What is static storage class?

Static is the default storage class for global variables. The two variables below (count and road) both have a static storage class.

```
static int Count; int Road;
  {
    printf("%d\n", Road);
  }
```

12. Define Auto storage class in C.

auto is the default storage class for all local variables.

```
    {
int Count;
auto   int
Month;

    }
```

The example above defines two variables with the same storage class. auto can only be used within functions, i.e. local variables.

13. Define Macro in C.

A macro definition is independent of block structure, and is in effect from the #define directive that defines it until either a corresponding #undef directive or the end of the compilation unit is encountered.

Its format is: #define identifier replacement

Example:

```
#define
TABLE_SIZE    100
int
table1[TABLE_SIZ
E];                int
table2[TABLE_SIZ
E];
```

14. What is Line control?

Line control (#line)

When we compile a program and some error happens during the compiling process, the compiler shows an error message with references to the name of the file where the error happened and a line number, so it is easier to find the code generating the error.

The #line directive allows us to control both things, the line numbers within the

code files as well as the file name that we want that appears when an error takes place. Its format is:
#line number "filename"

Where number is the new line number that will be assigned to the next code line. The line numbers of successive lines will be increased one by one from this point on.

15. What is the use of 'typedef''?

It is used to create a new data using the existing type.
Syntax: typedef data type name;
Example:
typedef int hours: hours hrs;/* Now, hours can be used as new datatype */
16. Write the syntax for pointers to structure.

```
Struct  S

{

char datatype1;

int datatype2;

float datatype3;

};

Struct S *sptr; //sptr ia pointer to structure S
```

17. Define self referential data structure

A self referential data structure is essentially a structure definition which includes at least one member that is a pointer to the structure of its own kind. A chain of such structures can thus be expressed as follows.

```
struct name {
        member 1;
        member 2;
        . . .
        struct name *pointer;
};
```

The above illustrated structure prototype describes one node that comprises of two logical segments. One of them stores data/information and the other one is a pointer indicating where the next component can be found. .Several such inter-connected nodes create a chain of structures.

18. What is singly linked list?

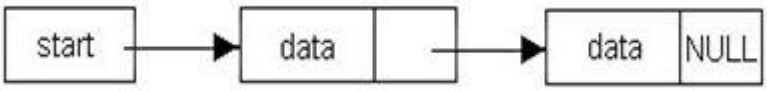
A linear linked list is a chain of structures where each node points to the next node to create a list. To keep track of the starting node's address a dedicated pointer (referred as *start*

pointer) is used. The end of the list is indicated by a *NULL*pointer. In order to create a linked list of integers, we define each of its element (referred as *node*) using the following declaration.

```
struct node_type {
        int data;
        struct node_type *next;
};
struct node_type *start = NULL;
```

Note: The second member points to a node of same type.

A linear linked list illustration:



19. What are the various dynamic memory allocation functions?

- malloc() - Used to allocate blocks of memory in required size of bytes.
- free () - Used to release previously allocated memory space.
- calloc() - Used to allocate memory space for an array of elements.
- realloac() - Used to modify the size of the previously allocated memory space.

20. How to create a node in linked list?

The basic thing is to create a node. It can be done by using the malloc function.

start = (node*) malloc(sizeof(node))

This statement creates the starting node of list. A block of memory whose size is equal to the sizeof(node) is allocated to the node pointer start. Typecasting is used because otherwise malloc will return pointer to character.

UNIT V
FILE PROCESSING

1. Why files are needed?

- When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.
- If you have to enter a large number of data, it will take a lot of time to enter them all. However, if you have a file containing all the data, you can easily access the contents of the file using few commands in C.
- You can easily move your data from one computer to another without any changes.

2. Types of Files

When dealing with files, there are two types of files you should know about:

1. Text files
2. Binary files

Text files

Text files are the normal .txt files that you can easily create using Notepad or any simple text editors.

When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.

They take minimum effort to maintain, are easily readable, and provide least security and takes bigger storage space.

Binary files

Binary files are mostly the .bin files in your computer.

Instead of storing data in plain text, they store it in the binary form (0's and 1's).

They can hold higher amount of data, are not readable easily and provides a better security than text files.

3. Enlist the File Operations.

In C, you can perform four major operations on the file, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

Working with files

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and program.

FILE *fptr;

4. How to open a file?

Opening a file is performed using the library function in the **"stdio.h"** header file: fopen().

The syntax for opening a file in standard I/O is:

ptr = fopen("fileopen","mode")

5. List the opening modes in standard I/O

File Mode	Meaning of Mode	During Inexistence of file

File Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	If the file does not exist, fopen() returns NULL.
rb	Open for reading in binary mode.	If the file does not exist, fopen() returns NULL.
w	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. i.e, Data is added to end of file.	If the file does not exists, it will be created.
ab	Open for append in binary mode. i.e, Data is added to end of file.	If the file does not exists, it will be created.
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
rb+	Open for both reading and writing in binary mode.	If the file does not exist, fopen() returns NULL.
w+	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb+	Open for both reading and writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exists, it will be created.
ab+	Open for both reading and appending in binary mode.	If the file does not exists, it will be created.

6. How to close a file?

The file (both text and binary) should be closed after reading/writing.

Closing a file is performed using library function fclose().

fclose(fptr); //fptr is the file pointer associated with file to be closed.

Reading and writing to a text file

For reading and writing to a text file, we use the functions fprintf() and fscanf().

They are just the file versions of printf() and scanf(). The only difference is that, fprint and fscanf expects a pointer to the structure FILE.

7. What are two main ways a file can be organized?

1. **Sequential Access** — The data are placed in the file in a sequence like beads on a string. Data are processed in sequence, one after another. To reach a particular item of data, all the data that proceeds it first must be read.

2. **Random Access** — The data are placed into the file by going directly to the location in the file assigned to each data item. Data are processed in any order. A particular item of data can be reached by going directly to it, without looking at any other data.

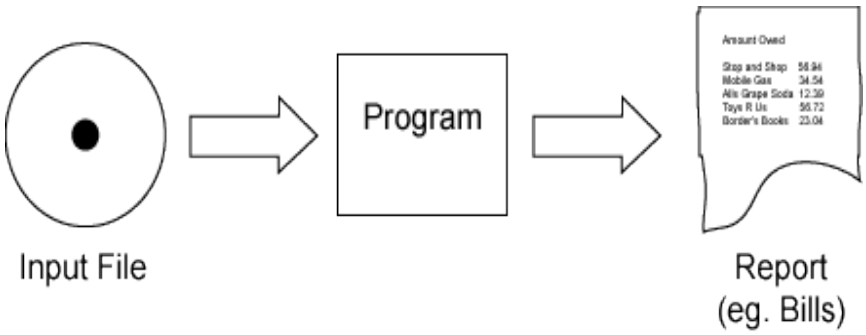
8. What is file?

A file is a semi-permanent, named collection of data. A File is usually stored on magnetic media, such as a hard disk or magnetic tape.

Semi-permanent means that data saved in files stays safe until it is deleted or modified.

Named means that a particular collection of data on a disk has a name, like mydata.dat and access to the collection is done by using its name.

9. State Report Generation.

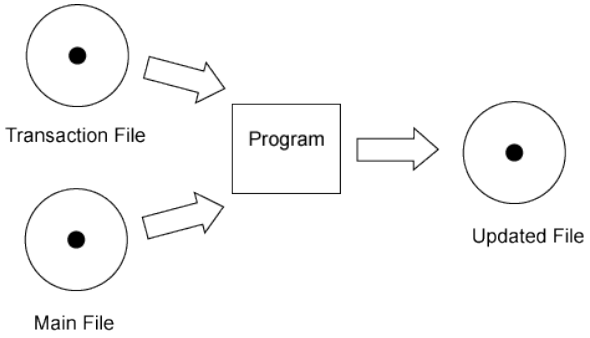


A very common data processing operation is **report generation**. This is when the data in a file (or files) is processed by a program to generate a report of some kind. The report might be a summary of the financial state of a corporation, or a series of bills for its customers, or a series of checks to be mailed to its employees.

Large corporations have very large databases kept on many high capacity disks and processed by programs running on large computers called **mainframe**

10. State Transaction Processing.

As data flows into an organization, the files that keep track of the data must be updated. For example, data flows into a bank from other banks and ATM machines. Often this data is gathered as it occurs into a **transaction file**. Periodically (often overnight) the data in the transaction file is used to update the main file of data.



The picture shows the main file and the transaction file as input to a program. The output is a new, updated main file. The previous version of the file can be kept as backup.

Management information science is the field that studies how to organize and manage the information of a corporation using computers and files. Usually the business school of a university has a management information science department.

11. List the Types of Files.

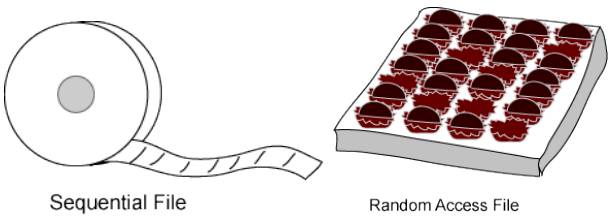
There are two main ways a file can be organized:

1. **Sequential Access** — The data are placed in the file in a sequence like beads on a string. Data are processed in sequence, one after another. To reach a particular item of data, all the data that proceeds it first must be read.

2. **Random Access** — The data are placed into the file by going directly to the location in the file assigned to each data item. Data are processed in any order. A particular item of data can be reached by going directly to it, without looking at any other data.

A sequential file works like a reel of tape. (In fact, sequential files are often stored on reels of magnetic tape.) Data in a sequential file are processed in order, starting with the first item, the processing the second, then the third and so on. To reach data in the middle of the file you must go through all the data that preceeds it.

A random access file works like a sampler box of chocolate. You can access a particular item by going directly to it. Data in a random acess file may be accessed in any order. To read data in the middle of the file you can go directly to it. Random access files are sometimes called **direct access** files.



You might think that all files should be random access. But random access files are much more difficult to imp.

12. What is command line arguments?

It is possible to pass some values from the command line to your C programs when they are executed. These values are called command line arguments and many times they are

important for your program especially when you want to control your program from outside instead of hard coding those values inside the code.

The command line arguments are handled using main() function arguments where argc refers to the number of arguments passed, and argv[] is a pointer array which points to each argument passed to the program.

13. Write an example program for command line arguments.

```
#include <stdio.h>
int main( int argc, char *argv[] )
{
if( argc == 2 ) {
                printf("The argument supplied is %s\n", argv[1]);
                }
else if( argc > 2 )
                {
                printf("Too many arguments supplied.\n");
                }
  else
                {
    printf("One argument expected.\n");
                }
}
```

14. Write the functions for random access file processing.

1. fseek()
2. ftell()
3. rewind()

15. Write short notes on fseek().

fseek():

This function is used for seeking the pointer position in the file at the specified byte.

Syntax: fseek(file pointer, displacement, pointer position);

Where

file pointer ---- It is the pointer which points to the file.

displacement ---- It is positive or negative.This is the number of bytes which are skipped backward (if negative) or forward(if positive) from the current position.This is attached with L because this is a long integer.

Pointer position:

This sets the pointer position in the file.

Value	pointer position
0	Beginning o
1	Current position
2	End of file

16. Give an example for fseek().

1) fseek(p,10L,0)

0 means pointer position is on beginning of the file,from this statement pointer position is skipped 10 bytes from the beginning of the file.

2) fseek(p,5L,1)

1 means current position of the pointer position.From this statement pointer position

is skipped 5 bytes forward from the current position.

3) fseek(p,-5L,1)

From this statement pointer position is skipped 5 bytes backward from the current position.

17. Give an example for ftell().

ftell()

This function returns the value of the current pointer position in the file.The value is count from the beginning of the file.

Syntax: ftell(fptr);

Where fptr is a file pointer.

18. Give an example for rewind().

rewind()

This function is used to move the file pointer to the beginning of the given file.

Syntax: rewind(fptr);

Where fptr is a file pointer.

19. State Block read/write.

It is useful to store the block of data into the file rather than individual elements. Each block has some fixed size, it may be of structure or of an array. It is possible that a data file has one or more structures or arrays, So it is easy to read the entire block from file or write the entire block to the file. There are two useful functions for this purpose

20. Write short notes on fwrite().

This function is used for writing an entire block to a given file.

Syntax: fwrite(ptr, size, nst, fptr);

Where ptr is a pointer which points to the arrayof struture in which data is written.

Size is the size of the structure

nst is the number of the structure

fptr is a filepointer.

Example program for fwrite():

21. Write short notes on fread().

This function is used to read an entire block from a given file.

Syntax: fread (ptr , size , nst , fptr);

Where **ptr** is a pointer which points to the array which receives structure.

Size is the size of the structure

nst is the number of the structure

fptr is a filepointer.

22. Write short notes on fprintf ().

This function is same as the printf() function but it writes the data into the file, so it has one more parameter that is the file pointer.

Syntax: fprintf(fptr, "controlcharacter",variable-names);

Where **fptr** is a file pointer

Control character specifies the type of data to be printed into file.

Variable-names hold the data to be printed into the file.

23. Write short notes on fscanf ().

This function is same as the scanf() function but this reads the data from the file, so this has one more parameter that is the file pointer.
Syntax: fscanf(fptr, "control character", &variable-names);
Where **fptr** is a file pointer
Control character specifies the type of data to be read from the file.
Address of Variable names are those that hold the data read from the file.

24. Write short notes on fscanf ().

The macro feof() is used for detecting whether the file pointer is at the end of file or not.It returns nonzero if the file pointer is at the end of the file otherwise it returns zero.

Syntax: feof(fptr);

Where fptr is a file pointer .

25. Write short notes on ferror().

ferror()

The macro ferror() is used for detecting whether an error occur in the file on filepointer or not.It returns the value nonzero if an error,otherwise it returns zero.

Syntax: ferror(fptr);

Where fptr is a file pointer.

PART B

UNIT I
BASICS OF C PROGRAMMING

1. Describe the structure of a C Program. [**UN**]
2. List the different data types available in C. [**RE**]
3. What are constants? Explain the various types of constants in C. [**UN**]
4. Explain the different types of operators available in C. [**UN**]
5. Describe the various looping statements used in C with suitable examples. [**UN**][**AP**]
6. Explain about various decision making statements available in C with illustrative programs. [**UN**][**AP**]
7. Write the operations of compilation process. [**AP**]
8. Write a C program to print the Fibonacci series of a given number. [**AP**]
9. Write a C program to solve the quadratic equation and to find a Factorial of a given number. [**AP**]
10. Write a C program to check whether a given number is prime number or not. [**AP**]

UNIT II
ARRAYS AND STRINGS

1. What is an array? Discuss how to initialize a one dimensional and two dimensional arrays with suitable example? **[RE][UN]**
2. Write a C program to search an element in a given array using linear and binary search. [**AP**]
3. Write a C program for sorting an array of numbers using selection sort. [**AP**]
4. Write a C program to addition, subtract and multiply two matrices. [**AP**]
5. Write a C program to scaling transformations of a matrix. [**AP**]
6. Write a C program to determinant a matrix. [**AP**]
7. Write a C program to transpose a matrix. [**AP**]
8. Write a C program to find mean, median and mode. [**AP**]
9. Explain in detail about string and list the various string operations with example. **[RE][UN]**
10. Write a C program to find out the length and reverse of the string without using built-in function. **[AP][AN]**

UNIT III
FUNCTIONS AND POINTERS

2. What is a function in C? Explain the steps in writing a function in C program with example. **[RE][UN]**
3. Classify the function prototypes with example C program for each. **[UN]**
4. What is recursion? Write a C program to find the sum of the digits, to find the factorial of a number and binary search using recursion. **[RE][AP]**
5. Write a C program to design the scientific calculator using built-in functions.**[AP]**
6. Explain about pointers and write the use of pointers in arrays with suitable example. **[RE][UN]**
7. Explain the concept of pass by value and pass by reference. Write a C program to swap the content of two variables using pass by reference. **[RE][AP]**

UNIT IV
STRUCTURES

1. What is a structure? Create a structure with data members of various types and declare two structure variables. Write a program to read data into these and print the same. Justify the need for structured data type. **[RE][AP]**
2. Write a C program to store the employee information using structure and search a particular employee using Employee number. **[AP]**
3. Define and declare a nested structure to store date, which including day, month and year. **[RE][UN]**
4. Explain about array of structures and pointers in structures with example program. **[RE]**
5. Write a C program to create mark sheet for students using self referential structure. **[AP]**
6. Discuss about dynamic memory allocation with suitable example C program. **[RE]**

7. Explain about singly linked list with suitable example C program. **[RE]**

UNIT V
FILE PROCESSING

1. Explain about files and with it types of file processing. [**RE]**
2. Compare sequential access and random access. **[AN]**
3. Write a C program to finding average of numbers stored in sequential access file and random access file. **[AP]**
4. Write a C program for transaction processing using random access files. **[AP]**
5. Describe command line arguments with example C program. [**RE]**