INDEX

SUBJECT CODE/NAME: CS3251 – PROGRAMMING IN C

SYLLABUS

CS3251 PROGRAMMING IN C 　　L T P C

3 0 0 3

COURSE OBJECTIVES:

- To understand the constructs of C Language.
- To develop C Programs using basic programming constructs
- To develop C programs using arrays and strings
- To develop modular applications in C using functions
- To develop applications in C using pointers and structures
- To do input/output and file handling in C

UNIT I BASICS OF C PROGRAMMING 9

Introduction to programming paradigms – Applications of C Language - Structure of C program – C programming: Data Types - Constants – Enumeration Constants - Keywords – Operators: Precedence and Associativity - Expressions - Input/Output statements, Assignment statements – Decision making statements - Switch statement - Looping statements – Preprocessor directives - Compilation process.

UNIT II ARRAYS AND STRINGS 9

Introduction to Arrays: Declaration, Initialization – One dimensional array – Two dimensional arrays - String operations: length, compare, concatenate, copy – Selection sort, linear and binary search.

UNIT III FUNCTIONS AND POINTERS 9

Modular programming - Function prototype, function definition, function call, Built-in functions (string functions, math functions) – Recursion, Binary Search using recursive functions – Pointers – Pointer operators – Pointer arithmetic – Arrays and pointers – Array of pointers – Parameter passing: Pass by value, Pass by reference.

UNIT IV STRUCTURES AND UNION 9

Structure - Nested structures – Pointer and Structures – Array of structures – Self referential structures – Dynamic memory allocation - Singly linked list – typedef – Union - Storage classes and Visibility.

UNIT V FILE PROCESSING 9

Files – Types of file processing: Sequential access, Random access – Sequential access file - Random access file - Command line arguments.

COURSE OUTCOMES:

Upon completion of the course, the students will be able to

CO1: Demonstrate knowledge on C Programming constructs

CO2: Develop simple applications in C using basic constructs

CO3: Design and implement applications using arrays and strings

CO4: Develop and implement modular applications in C using functions.

CO5: Develop applications in C using structures and pointers.

CO6: Design applications using sequential and random access file processing.

TOTAL : 45 PERIODS

TEXT BOOKS:

1. ReemaThareja, "Programming in C", Oxford University Press, Second Edition, 2016.

2. Kernighan, B.W and Ritchie,D.M, "The C Programming language", Second Edition, Pearson

Education, 2015.

REFERENCES:

1. Paul Deitel and Harvey Deitel, "C How to Program with an Introduction to C++", Eighth

edition, Pearson Education, 2018.

2. Yashwant Kanetkar, Let us C, 17th Edition, BPB Publications, 2020.

3. Byron S. Gottfried, "Schaum's Outline of Theory and Problems of Programming with C",

McGraw-Hill Education, 1996.

4. Pradip Dey, Manas Ghosh, "Computer Fundamentals and Programming in C", Second

5. Edition, Oxford University Press, 2013.

6. Anita Goel and Ajay Mittal, "Computer Fundamentals and Programming in C", 1st

Edition, Pearson Education, 2013

UNIT-I BASICS OF C PROGRAMMING

Introduction to programming paradigms - Application of C language - Structure of C Program - Data types - Constants - Enumeration Constants - keywords - operators: precedence and Associativity - Expressions Input / output Statements, Assignment Statements - Decision making Statement - Switch statement - looping statement - preprocessor directives - Compilation process.

PART-A

1) What is the importance of keyword in C? (AU. Apr/may 2015)

keywords are reserved words whose meaning has already been explained to the compiler. The keywords are also called as reserved words.

eg:- auto, else, long, for, float, int, char, continue, goto, etc.

2) What do you mean by C tokens? (AU May/June

The smallest individual units of C

program are known as C tokens.

eg:- keywords, identifier, constants, string, operators, special character.

3) What are variable? Give example. (AU May 2016)

- A variable is an identifier that is used to represent some specified type of information within a designated portion of the program.

eg:- int sum;

4) Write a for loop statement to print number 10 to 1. (AU Jan 2019)

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

5) Give the use of pre processor. (AU Apr/May 2015)

— pre processor is a translator that convert a program written in one high level language into an equivalent program written in another high level language.

6) Difference between while and do...while (AU-2018)

while	do... while.
This is the top tested loop	This is bottom tested loop
The condition is first tested, if the condition is true then the block is executed until the condition becomes false.	It executes the body once, after it checks the condition, if it is true the body is executed until the condition becomes false.
Loop will not be executed if the condition is false.	Loop is executed atleast once, even the condition is false.

7) What is the use of 'break' statement? (Jan 2009)

The break statement is used to terminate the loop. when the keyword break is used inside any 'c' loop, control automatically transferred to the first statement after the loop.

eg
```
while ( condition)
{ ---
   if ( condition)
   break;
}
```

8) Define delimiters in C. (Jan 2010)

Hash – Pre-Processor directive

, Comma – Separate list of variable

: colon - Lable delimiters.

; Semicolon - statment delimeter.

() paranthesis - used in expressions.

{} Curly braces - Used blocking c Structure.

[] square braces - used along with arrays.

9) What is the use of #define processor (AU-2012)

The #define preprocessor directive is used to define a

Global Constant	#define PI 3.14
Macro	#define big (a,b)

When a Compiler read the macro name, it replaces the macro name by its expression. When it reads the constant lable, it replaces the label by its value.

10) List out various Input & output statments in C (AU 2018)

The input & output statment are classified into formatted & unformatted I/O

Formatted I/O : User can able to design / format the output. scanf(), printf()

Unformatted I/O : doesn't allow the users to design the output.
getch(), gets(), putch(), puts()

PART- B

1) Explain the decision making statement in C with example programs. (AU 2012 / AU 2018)

Decision making:-

- Decision making ~~ma~~ is about deciding the order of execution of statements based on certain conditions.

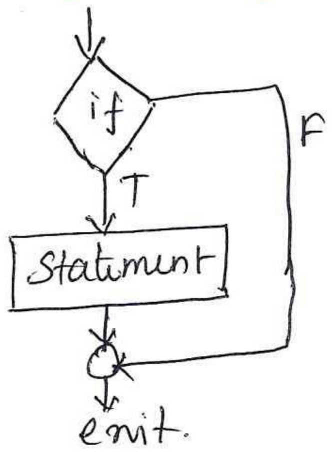
- C language handles decision-making by supporting the following statements.

Simple-if statement:-

- The if statement is simplest form of selection statement, that is frequently used in decision making.

Syntax:-

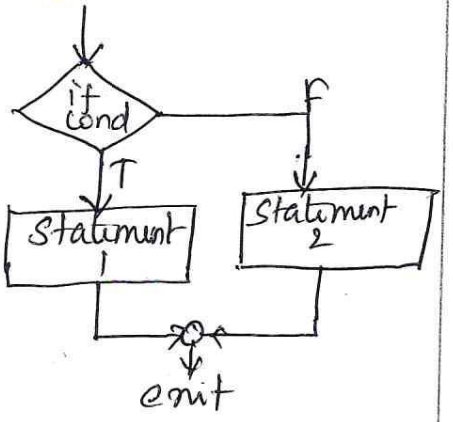
```
if (condition)
{ statement
}
```



eg:-
```
#include<stdio.h>
void main()
{ int n;
  Printf("Enter the number");
  scanf("%d",&n);
   if(n>0)
     Printf("No is positive");
}
```

If-else statement :-

if else statement evaluates the condition, if it is true, the true block will be executed, otherwise, the false block will be executed.



Syntax :-

```
if (condition){
    statement 1;
else
    statement 2;
```

eg

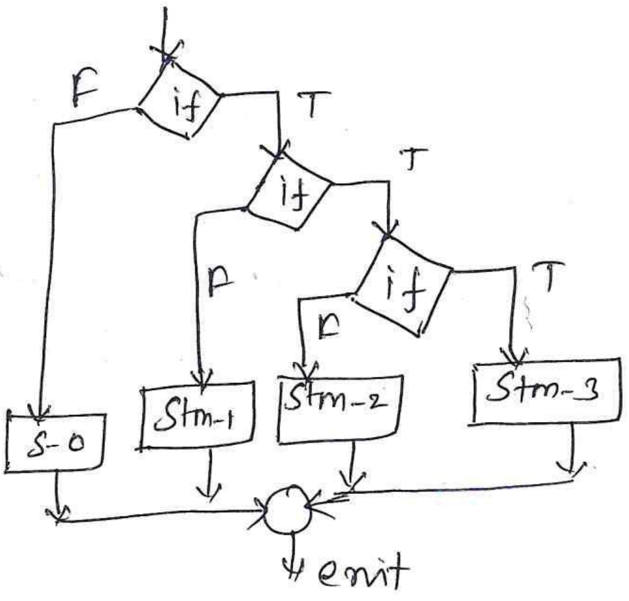
```
#include<stdio.h>
#include<conio.h>
void main()
{   int n;
  printf(" Enter number");
  scanf("%d",&n);
   if(n>0)
    printf(" No is positive");
    else
     printf(" No is negative");
}
```

Nested-if :-

If one or more if statements are embedded within the if statement is called nested if statement.

Syntax:-

```
if ( condition1)
  { if (condition2)
{ if (condition 3)
    {
        stmtn 3;
    }
    else
        stm-2;
  }
  else
      stm 1;
}
else
    stm 0;
}
```



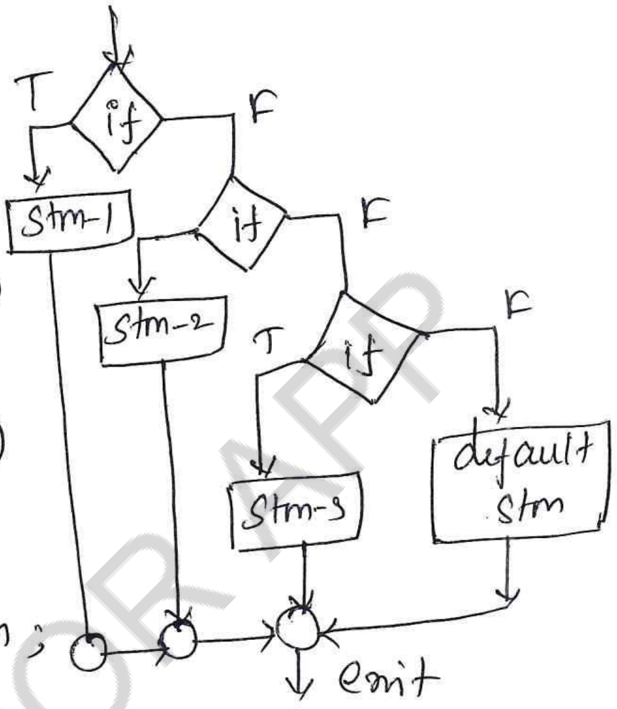
eg:-
```
#include<stdio.h>
void main()
{ int a,b,c;
  printf(" Enter three numbers");
  Scanf("%d%d%d",&a,&b,&c);
  if (a>b)
   { if(a>c)
      { printf(" A is big");
      }
      else
       printf(" c is big");
   }
   else{if(b>c)
          printf(" B is big");
         else
      }  printf(" c is big");
   }
```

if_else_if ladder:-

- The nested if statement becomes complex when there are more than three condition. In that situation, it can be represented in if-else-if ladder control structure.

Syntax:-

```
if (cond1)
   stm-1;
else if ( cond 2)
   stm-2 ;
else if (cond 3)
       stm-3 ;
else
     default stm;
```



eg:-

```
#include< stdio.h>
void main()
{ int avg;
  printf(" Enter the avg marks");
  if (avg > 90)
  printf(" O Grade");
   else if ( avg > 80)
   printf(" A+ Grade");
   elseif (avg > 70)
    printf(" A Grade");
    elseif ( avg>60)
     printf (" B+ Grade");
```

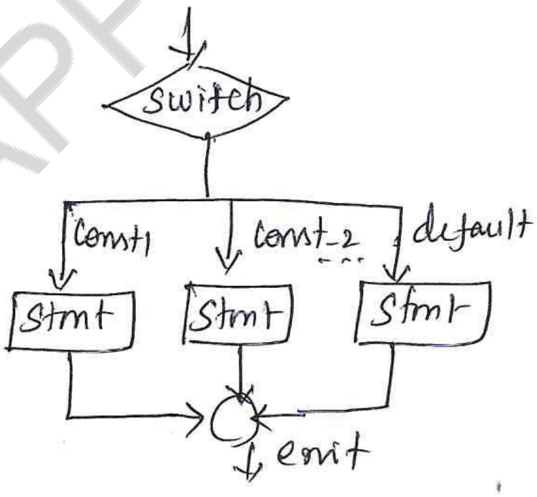
```
        else if (avg >= 50)
          printf(" B Grade");
        else
          printf(" Fail");
   }
```

Switch case statment :-

- The switch case Statment is a multiple branching (multiple decision making) statment.
- The switch statment select a particular group of statment from several available group

Syntax:-

```
   switch (exp)
   { case  const1 : stmt;
                    break;
       case Const 2: Stmt;
                    break;
              :
         default : Stmt;
   }
```



eg:-
```
   #include< stdio.h>
   void main()
   { int a,b,c;
     char op;
     printf(" + Add \n - Sub \n * mul \n");
     Printf(" Enter the case value");
     scanf("%c", op);
     Printf(" Enter the values");
```

```
scanf("%d %d", &a, &b);
switch(op)
{
    case '+' : c=a+b;
               break;
    case '-' : c=a-b;
               break;
    case '*' : c=a*b;
               break;
}
printf("Result = %d", c);
}
```

2) Explain the looping statement in c with example program. (AU 2012 / AU-2011 / AU-2007)

<u>Looping statements / Iterative statement :-</u>

– Looping is the process of repeating the same set of statements again and again until the specified condition hold true.

$$\begin{tikzcd}
 & \text{loop} \arrow[dl] \arrow[dr] & \\
\text{Counter Controlled loops} & & \text{sentinal Controlled loop}
\end{tikzcd}$$

<u>Counter Controlled loop :-</u>

– In counter Controlled loop, the number of iterations to be performed is known in advance. It is also called as definite repetition loop.

Sentinal Controlled Loop:-

- In sentinal loop, the number of times the iteration is to be performed is not known in advance. It is also called as indefinite repetition loop.

- Three types of looping statements.

* while loop * do.. while loop * for loop.

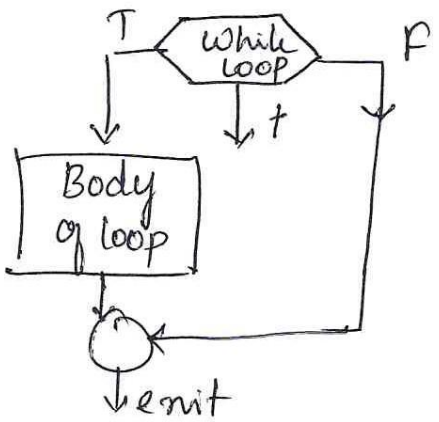
while Loop:-

- The while loop is entry controlled loop, because the control condition is placed at the first lone of the code.

- If the control condition evaluates to false, then the statements enclosed in the loop are never executed.

Syntax:-

```
intialization;
while (condition)
{ body of the loop;
  inc/dec;
}
```



eg:-

```
#include<stdio.h>
void main()
{ int n=0;
  while(n<=5)
  { printf("%d", n);
  } n++;
}
```

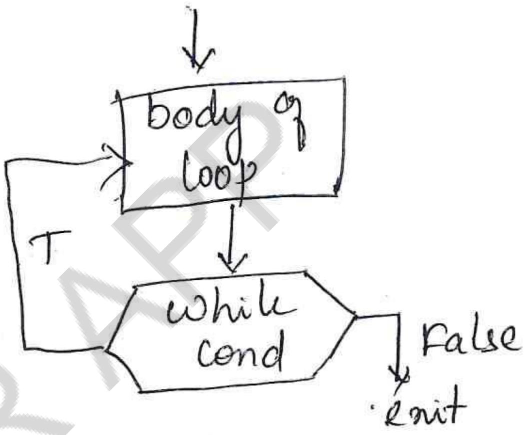
do... while loop :-

- The do.. while loop is an exit controlled loop because the test condition is evaluated at the end of the loop.

- The body of the loop gets executed atleast one time. the test condition must be terminated by (;) semi colon in do-while loop.

Syntax:-

```
    initialization;
    do
    { body of loop;
       inc/dec;
    } while(cond);
```



eg

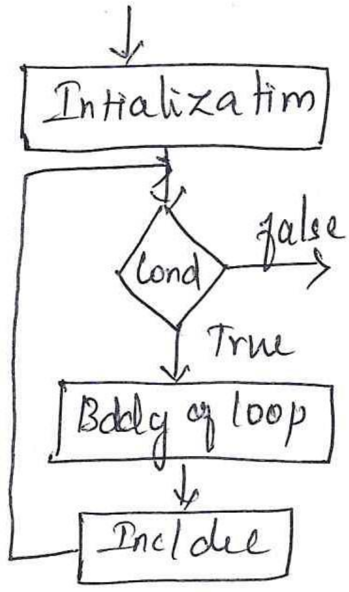
```
  #include <stdio.h>
   void main()
    { int n=0;
      do
        { printf("%d",n);
           n++;
        } while(n<=5);
      }
```

For loop :-

It is definite loop. It is used to execute a set of instructions repeatedly, until the condition becomes false.

Syntax:-

```
for( intialization ; condition ; increment/decrement);
  { body of the loop;
  }
```



eg:-

```
#include <stdio.h>
void main()
 { int i ;
  for(i=1; i<=5; i++)
   { printf("%d", i);
   }
 }
```

Nested for loop:-

- The loop within the loop is called nested loop. The number of iterations ~~is~~ in this type of structure is equal to the number of iterations in the outer loop multiplied by the number of iterations in inner loop.

eg:-

```
#include <stdio.h>
void main()
 { int i, j ;
   for(i=1 ; i<=3 ; i++)
    { printf("\n");
      for (j=1; j<=3; j++)
      Printf("%d", j);
 }  }
```

3) Explain different types of operators in C with example program. (AU 2017 / AU 2010 / AU 2008)

Operator:-

– An operator is a symbol that specifies an operation to be performed on the operand. An operand is defined as a variable (data items).

Types of operator:-

* Arithmetic Operator.
* Relational operator
* Logical operator.
* Assignment operator.
* Increment / decrement operator.
* Conditional operator.
* Bitwise operator.
* Special operator.

Arithmetic operator:-

C allows us to carryout basic arithmetic operations like addition (+), subtraction (-), multiplication (*) and division (/).

eg:-
```
#include <stdio.h>
void main()
{ int a=5, b=3;
   printf("Sum=%d", a+b);
   printf("Diff=%d", a-b);
```

```
    printf("Mul =%d", a*b);
    printf("Div = %d", a/b);
    printf("Modular Div= %d", a%b);
}
```

Relational Operator :- (<, <=, >, >=, ==, !=)

Relational operator are used to compare two or more operands. operands may be var, constant, Expressions. The result of relational expression is either one (or) zero.

eg:-
```
#include<stdio.h>
void main()
{ int a=5, b=10;
  printf("a>b :%d", a>b);
  printf("a!=b :%d", a!=b);
  printf("a<=b :%d", a<=b);
}
```
O/p :- a>b : 0   a!=b : 1   a<=b : 1

Logical operator :- (&&, ||, !)

– Logical operators are used to compare the result of two (or) more conditions.

operator	Meaning	Eg	Relation
&&	logical AND	(9>2)&&(5<2)	0
||	logical OR	(5<2)||(5==4)	0
!	logical NOT	!(5<=2)	1

eg
```
#include<stdio.h>
void main()
{ int a=9, b=2, c=5;
  Printf("%d", (a>2)&&(c<b
  Printf("%d", (a>2)||(b>c));
  Printf("%d", !(a>2));
}
```
O/p 1 1 0

Assignment operator:-

Assignment operators are used to assign a value of an expression (or) a value of a variable to another variable.

Syntax:-

var = exp;

i) Compound assignment:-

Apart from assignment operator, c provides compound assignment operator to assign a value to the variable in order to assign a new value to variable after performing a specified operation.

operation	Example	Meaning
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

eg

```
#include<stdio.h>
void main()
{ int a=50, b=5;
  b+=a;
 printf("%d", b);
}
```

(ii) Nested (or) Multiple Assignment:-

- We can assign a single value (or) an expression to multiple variable.

Syntax:- var1 = var2 = var3 = exp (or) var;

eg:- a=b=c=10;

Increment and Decrement Operator (Unary oprtr):-

- The increment operator ++ adds one to the variable and decrement operator -- subtract one from the variable.

operator	Meaning
++a	Pre_Increment
a++	post_Increment
--a	pre-decrement
a--	post_decrement

```
void main()
{ int a=5;
  printf("%d", a++);
  Printf("%d", ++a);
  printf("%d", --a);
}
```

O/P

```
a++ = 5
++ a = 7
--a = 5
```

Conditional operator (or) Ternary operator:-

Conditional operator itself check the Condition and execute the statement depanding on the Condition

Syntax:- Condition ? exp1 : exp2;

eg:-
```
#include <stdio.h>
void main()
{
  int a=5, b=3, big;
  Printf("Big%d",
    big = a>b ? a : b;
  Printf("Big %d", big);
}
```

- The ?: act as a ternary operator, It check the condition, If it is true, exp1 is evaluated. If it is false exp2 is evaluated.

Bitwise Operator:-

- Bitwise operator is used to manipulate the data at bit level. It operates on integer only.

operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift left
>>	Shift Right
~	One's Compliment

eg:-
```
#include <stdio.h>
void main()
{ int x=7, y=8;
printf("x&y :%d", x&y);
Printf("x|y :%d", x|y);
Printf("x^y :%d", x^y);
}
```

Special operator:-

member relation operator (·, →) :- used to access elements from a structure.

Comma operator:- (,) used to separate the statement elements such as var, constants, exp.

Sizeof () operator:- Return the length of the specified variable.

pointer operator:- (&, *) & specify address of variable. * specify value of the variable.

4) Explain about I/O statements in C with example program (AU-2008)

— The input/output function permit the transfer of information between the computer and the standard input/output devices. — It classified into two types.

* Formatted I/O statements
* Unformatted I/O Statements.

Formatted I/O statements:-

formatted input :-	formatted output
scanf ();	printf () ;
fscanf();	fprintf();

scanf ():-

— It is used to read formatted data from keyboard. It takes text stream from keyboard extract and format data from the stream according to a format control string and then store the data in specified program variable.

Syntax:-

scanf ("Control String", &arg1, &arg2, ... &argn);

control string begin with percent sign (%) followed by a Conversion character.

%c - single character %d - decimal integer

%f - float value %s - String.

Printf ():-

It is used to display information required by the user and also print the value of the variable.

Syntax:-

printf ("Control String", arg1, arg2, ... argn);

eg:-

```
#include<stdio.h>
#include< conio.h>
void main()
{ int rollno;
    float avg;
    char name[10];
printf ("Enter name, roll_no, avg");
scanf (" %s %d %f"&name, &roll_no, &avg);
Printf (" Name =%s roll_no =%d avg= %d", name, rollno, avg);
getch();
}
```

Unformatted I/O statements :-

Input statements	output statements
gets()	putc()
getchar()	putchar()
getch()	puts()
getche()	
getc()	

- It is simplest form of all input/output operations are reading a character from the standard input unit and writing it to the standard output unit.

getchar() :-

- It reads a single character from a standard input device (keyboard). till the user press the key enter.

Syntax :- var_name = getchar();

getche() :-

- It is also read a single character from a keyboard.

Syntax :- var_name = getche();

- When this statement is executed the entered character is displayed on the output screen without waiting for the press ~~the~~ enter key.

putchar() :-

- Single character can be displayed

using the library function putchar.

Syntax:- putchar(var-name);

Input/output of String data:-

gets() :-

- It is accept the name of the String as a parameter, till a newline character is encountered.

Syntax:- gets(str);

puts() :-

- It is used to display the string to the standard output device.

Syntax:- puts(str);

eg:-
```
#include<stdio.h>
#include<conio.h>
void main()
{
    char str[30];
    puts("Enter name");
    gets(str);
    puts("programming in c");
    puts(str);
    getch();
}
```

O/P
```
Enter name
  Abi
Programming in c
   Abi
```

5) Write short note on following statements i) goto ii) break iii) continue (AU-2010) / Explain about unconditional statements in c (AU 2018 / AU 2012)

- The unconditional statement transfer the control from one point to another without checking any condition.

Goto statement :-

- The goto statement is used to branch unconditionally from one point to another point in the program.

- The goto statement require lable inorder to identify the place where the control to be transferred.

syntax:-

```
goto label;
 - - - -
 - - -
label;
```

eg:-

```
#include<stdio.h>
void main()
{ int a=50, b=20;
   if (a>b)
      goto big;
   else
      printf("B is big");
  big : printf("A is big");
}
```

Break Statement:-

- It is used to terminate the loop, when the keyword 'break' is used inside any c loop.

Syntax:-

```
break;
```

eg:-

```
#include <stdio.h>
void main()
{ int i;
  for(i=1; i<=5; i++)
  { if (i==3)
      break;
    printf("%d", i);
} }
```

O/P :- 1, 2

Continue Statement:-

- Continue statement is used to continue next iteration of loop statement when it occurs in the loop.

- It doesnot terminate but it skip the statement after this statement.

Syntax:-

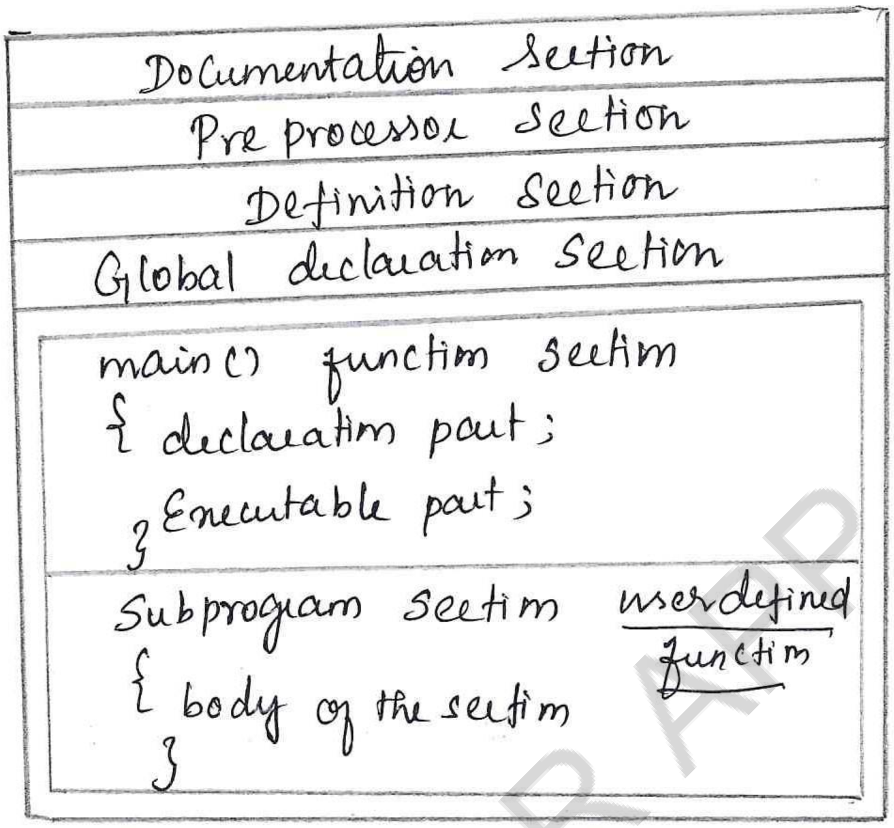
```
Continue;
```

eg:-

```
#include<stdio.h>
void main()
{ int i;
  for(i=1; i<=5; i++)
  {  if(i==3)
       Continue;
     Printf("%d", i);
  }
}
```

O/P 1 2 4 5

6) Explain the structure of c program. Write Applications of C. (AU-2014/AU 2008)



Documentation Section:-

- The documentation section consist of a set of comment lines giving the name of program and other detail.

- Comment line is a non-executable statment. It is placed between /* and */

Pre processor Section:-

- pre processor section which direct the compile to link functions from the system library.

eg:- #include <stdio.h>

#define PI 3.14

Global declaration section :-

- There are some variable that are used in more than one function. such variables are called global variable and are declared in the global declaration section.
- It is declared outside of all the functions.

Function main () :-

- Every program written in c language must contain main() function. this section contains two parts.
  * Declaration part
  * Executable part.
- Declaration part contains the statements following main() statements. It declares all the variable used in the executable part.
- Executable part contain the statements following declaration part. Here logic of the program can be implemented.
- they provide a instruction to the Computer to perform a Specific task.
- These two parts appear between the opening and close braces ({ })

Sub program section :-

- It contains all the defined fun that are called in the main function.

- User defined functions are generally placed immediatly after the main function., although they may appear in any order.

eg:-

```
/* Sum of two numbers */        Document section
#include <stdio.h>              Preprocessor section.
#define A 5                     Definition section.
int c ;                         Global variable declaration.
int add ( int);
main ()                         main function.
{ int b;
   printf(" Enter the value of b");
   scanf(" %d", &b);             } Execute part
   c = add (b)
   printf ( " Sum = %d", c);
}
int add (int b)
  { c = A+b ;                    } Sub program.
       return c ;
  }
```

Applications of C :-

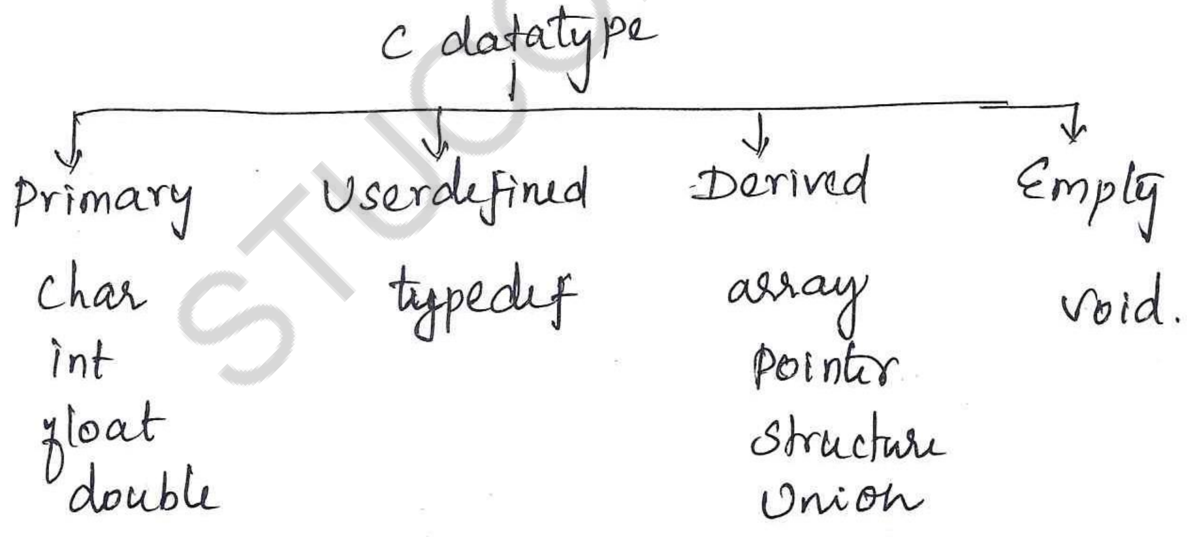
* Used in operating system
* Used in GUI

* Used in New Programming platform
* Used in Embedded systems.
* Used in Compiler design
* used in Gaming and Animation.

7) Explain the different types of data available in C? (AU-2008)

- Datatypes are the type of data, that are going to access within the program.

- C supports different data types, each data type may have predefined memory requirement and Storage representation.



Primary datatype:-

Primary datatype or fundamental datatype

* int * float * double * char.

Integer:-

Integers are the numbers with supported

range. Integers occupied one word of storage. Sizes of integers vary from 16 bit or 32 bits.

eg:- int a ;

<u>float</u> :-

- floating point number are stored in 32 bit, with 6 digits precision. It is used to store numeric values with decimal point.

eg:- float x ;

<u>Double</u> :-

- Double datatype number uses 64 bit giving a precision of 14 digits, there are known as double precision number.

eg:- double y ;

<u>Character</u> :-

- A single character can be defined as char datatype character are usually stored in 8 bit. eg:- char a;

<u>Type qualifier</u> :-

- If we use 16 bit word length size of data type is limited to specific range. Inorder to provide the some control over the range of number and storage space of the datatype, we use typequalifier.

* Sort
* long
* signed
* Unsigned

- The declaration of long and unsigned integer permits us to increase the range of values.

__User defined datatype:-__

- User defined datatype is used to create new data types. 'type declaration' allows user to define an identifier that would represent an existing datatype.

__Syntax:-__ typedef type identifier.

type refers to the existing datatype, identifier refers to new name given to the datatype.

eg:- typedef int mark;

- Another user defined datatype is enumerated datatype.

__Syntax:-__ enum identifier {var1, var2, ...varn}

eg:- enum day {monday, Tuesday, wednesday}

enum color {blue=2, black=0, green=1}

__Derived datatype:-__

- Datatype that are derived from

fundamental datatype are called derived datatype. Derived datatype add some functionality to the basic datatypes.

— Array is a collection of variables of sametype.

— pointer is a special variable that hold a memory address of another variable.

— structure is a multiple values of same or different datatypes under a single name.

— Union is also same as structure but in memory, union variable stored in common memory location.

— Void datatype is used to represent an empty or null value. It used as return type if function doesnot return any value.

———x———

C program to find square root of value

```
#include < stdio.h>
void main()
{ float x;
  printf(" Enter x value");
  scanf ("%f", &x);
  Printf(" Square root is %f",
                        sqrt(x));
}
```

Armstrong Number

```
#include<stdio.h>
void main()
{ int a,b,c,d,e;
  Print(" Enter number ");
  Scanf(" %d", &a);
  e=a;
  while(a>0)
  {b=a%10)
   c=b*b*b;
   a=a/10;
   d=c+d;
  }if (e==d)
   Pf(" Armstrong");
  else
  } Pf(" Not Armstrong");
```

UNIT - II ARRAYS AND STRINGS

Introduction to Arrays; Declaration, initialization - One Dimensional array - Two dimensional array - String Operations, length, Compare, Concatenate, Copy - Selection Sort, linear and binary Search.

PART - A

1) What is an array? Write down the Syntax for array declaration (AU. Apr/May 2022).

Array is a collection of Similar type of Values. All Values are stored in Continuous memory Locations. All Values Share a Common name. The elements are Organized in sequential area.

Syntax : datatype array-name [size];

Example : int a[10];

2) Write a C function to compare two Strings (AU. Apr/May 2019).

strcmp() function is Used to Compares two strings. The two strings are Compared Character by character. If the function returns 0 Value means that both the Strings are same. otherwise it returns the numerical difference between the first non-matching characters.

Syntax : strcmp (string 1, string 2);

3) Write the Syntax for two-dimensional array? (AU. APr/ May 2018).

Two dimensional array is an array with two subscript Values. First subscript Specifies the row & second Specifies the column.

Syntax : datatype array-name [rows][columns];

Example : int Matrix A [10][10];

4) Name any two library functions Used for string handling? (AU. Nov/Dec. 2019).

Strlen() : Finds the length of a string. It returns an integer Value. It Counts the number of characters. except null character and return the Count.

Syntax : Var = Strlen (String);

strcpy() : Copies the Content of One String to another ant it almost works like String assignment operator.

Syntax : Strcpy (String 1 , String 2);

5) How a character array is declared ? (AU. Apr / May 2014).

Declaration : Char name [n];

This array can store n-1 characters.

Initialization : char name [10] = "India";

char name[10] = {'I', 'n', 'd', 'i', 'a'};

The Char array is terminated by '\0'.

6) Define Searching ? What are the two types of searching ? (AU. Apr/May 2018)

Searching is a process of finding the Position of a given elements in a list.

The searching is successful if the

element is found. There are two types of searching.

1. Linear search
2. Binary search.

7) Sort the following elements using Selection Sort method. 23, 55, 16, 78, 2.

Step 1 : Find smallest element in the list and exchange the element with first element of the list. 2, 55, 16, 78, 23.

Step 2 : Find Second Smallest element Value and exchange it with the second element.

2, 16, 55, 78, 23.

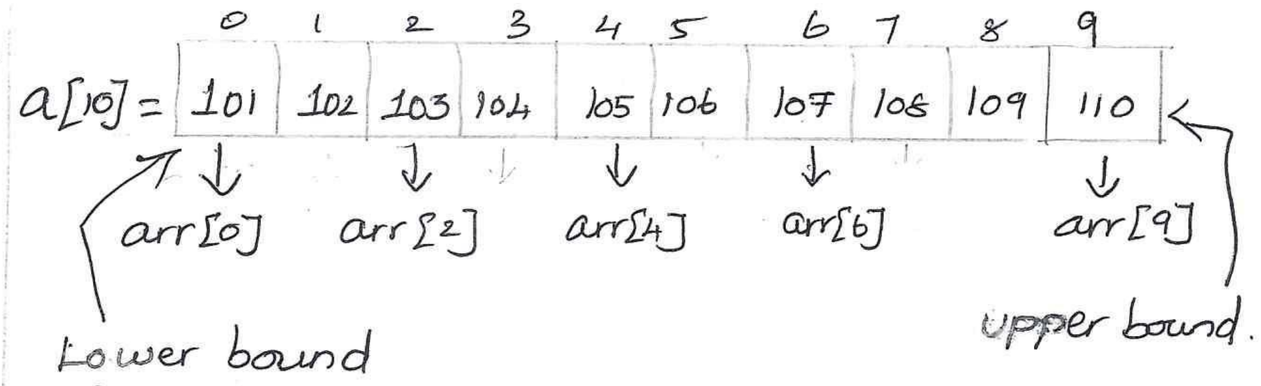
Step 3 : Continue the process Until all the elements are arranged in Order. 2, 16, 23, 78, 55.

Step 4 : 2, 16, 23, 55, 78.

8) Given an array int a[10] = { 101, 102, 103, 104, 105, 106, 107, 108, 109, 110 }. Show the Memory representation and calculate length.

(AU Nov/Dec. 2019).

Memory representation : Array elements are always stored in Consecutive Memory Location.



Length of an array : Upper bound - lower bound +1

= 9 - 0 + 1 = $\underline{9}$ + 1
= 10

9) Write any four features of arrays. (AU. Jan 2013).

* Array is a derived data type, Used to represent a collection of element in same data type

* Elements can be accessed with index and subscript define the position.

* Elements stored in continuous Memory Location.

* Array elements increment Values by Subscripts.

10) Define Sorting ? (AU. Jan 2013).

Sorting is a process of arranging the elements either in ascending Order or decending Order.

PART-B

1) What is an array? Explain about Various types of arrays in detail. (AU. APr/MAY 2022)

Array is a Collection of similar data items, that are stored Under a Common name. A Value in an array identified by index, enclosed in square brackets with array name.

Syntax : datatype array_name [size];

: int a[10];

Types of an array:

There are two types of an array.

1. One-dimensional array.
2. Two-dimensional array (Multi-dimensional array).

One-dimensional array:

The collection of data items Can be Stored Under a One Variable name Using Only One subscript, Such a Variable is called One-Dimensional array.

Array Declaration:

Arrays are declared in the Same manner as an Ordinary Variables except that each array name must have the size of the array.

Syntax : datatype array_Variable [size];

datatype → What kind of Values it can Store (int, char, float, double).

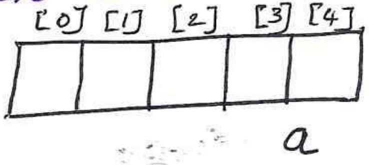
Variable / Name → To identify the array.

Size → the Maximum number of Values that the array can hold.

Example : int a[5];

c array starts from index 0 and ends with n-1.

int a[5];



Calculating the address of array elements:

An array stores all its data elements in Consecutive memory Locations. storing Just the base address. Address of first element is sufficient (Eg). Base address-1000.

int mark[ ] = {89, 87, 98, 56, 88, 80, 79}

Base address = 1000.

Calculate the address of marks [5] } = 1000 + 2(5-0)

= 1000 + 10

= 1010.

Storing an integer Value requires 2 bytes.

__Calculating the length of an array:__

Length of the array is given by the number of elements stored in it

Length = Upper bound - lower bound + 1.

Storing Values in the array based on

* Initialize the array element at the time of declaration.

Syn: type arrayname [size] = {list of Values}.

Ex: int marks[5] = {80, 85, 90, 95, 98}.

int marks [ ] = {90, 95};

80 85 90 95 98

90 95 0 0 0

Initialize the Value of an array, Programmer may Omit the size of the array.

* Inputting Values from the keyboard

while or do while / for loop is executed to input the Value for each element.

Eg:
```
int i, a[5];
for (i=0; i<5; i++)
    Scanf("%d", &a[i]);
```

* Assigning Values for individual elements

Eg: Marks[3] = 98..

We cannot assign One array to another.

Two - Dimensional arrays.

It is specified by two subscripts where one subscript denotes row and the Other denotes column.

Declaring two-dimensional arrays:

* Specifies the name of the array, the array datatype of each element, and size of each dimension.

Syntax: datatype arrayname[rowsize]
[column size];

Ex: a[3][3];

```
int  a[3][3] = { {1,2,3}, {4,5,6},
                                {7,8,9}};
```

Accessing the elements of 2D array:

Using two subscripts. The first for loop will scan each row and second for loop will scan individual column for everyrow in the array.

eg:
```
#include <stdio.h>
int main()
{
  int a[2][2] = {75,80,85,98};
  int i,j;
  for (i=0; i<2; i++)
  {
     Printf("\n");
   for (j=0; j<2; j++)
     Printf("%d", a[i][j]);
} }
```

2) Write a C program to find the addition of two matrices (AU. Jan 2014).

* Matrices are an important aspect in C program.

* It is Used in wide range of application like Computer graphics, image Processing etc.

* It is allow for efficient Manipulation and storage of Large sets of data.

* The elements of a matrix Can be accessed Using their row and column.

Program:

```
#include < stdio.h>
Void main ()
{
int a[5][5], b[5][5], c[5][5];
int i,j,m,n;
Printf(" Enter the row and column size");
Scanf ("%d-%d", &m &n);
Printf (" Enter Matrix A Elements\n");
for (i=0; i<m; i++)
{
For (j=0; j<n; j++)
{   scanf ("%d", &a[i][j]);
}}
```

```
Printf(" Enter the B Matrix elements");
for (i=0; i<m; i++)
 {
for (j=0; j<n; j++)
{
 Scanf ("%d", &b[i][j]);
  }
}
 Printf(" Matrix addition");
 for (i=0; i<m; i++)
 {
  for (j=0; j<n; j++)
 {
  c [i][j] = a[i][j] + b[i][j] ;
  }
 }.
 Printf (" Resultant Matrix");
 for (i=0; i<m; i++)
 {   Printf("\n");
for(j=0; j<n; j++)
{
  Printf("%d\t", c[i][j])
  }
 }
}.
```

Output:

```
Enter the row and column size.

2   2

Enter Matrix A Elements

3   5
2   7

Enter the B Matrix elements

4
1
5
2

Matrix addition.

Resultant Matrix

 7    6
 7    9.
```

3) Write a c program to find the transpose of a matrix. (AU. NOV/DEC 2019).

* The transpose of a matrix is a new matrix that is obtained by exchanging the rows and columns. $A = m \times n$ matrix

* Interchanging rows into columns or columns into rows. It is denoted by T. in the superscript of matrix.

A^T = Transpose of Matrix A.

Program:

```
#include <stdio.h>
int main()
{
  int a[10][10], transpose[10][10],
  int r,c,i,j;
  Printf("Enter rows and columns:");
  Scanf("%d %d", &r,&c);
  Printf(" Enter Matrix elements:\n");
  for (i=0; i<r; ++i)
     for (j=0; j<c; ++j)
{
```

```
Printf ("Enter element a %d %d :",
                          i+1, j+1);
 Scanf ("%d", &a [i][j]);
 }
Printf ("\n Entered the Matrix is:\n");
for (i=0; i<r; ++i)
   for (j=0; j<c; ++j)
   {   Printf ("%d", a [i][j]);
       if (j == c-1)
             Printf ("\n");
   }
for (i=0; i<r; ++i)
  for (j=0; j<c; ++j)
   {
      transpose [j][i] = a[i][j];
   }
 Printf ("\n transpose of Matrix:\n");
 for (i=0; i<c; ++i)
for (j=0; j<r; ++j)
{
     Printf ("%d", transpose [i][j]);
```

```
if (j == r-1)
        Printf("\n");
}
```

Output:

```
Enter rows and columns : 2
Enter Matrix elements :
Enter element a11 : 1
Enter element a12 : 2
Enter element a21 : 3
Enter element a22 : 4
Entered the Matrix is :
2
4
Transpose of Matrix :
3
4.
```

4) Explain the function (i) strlen() (ii) strcpy() (iii) strcat() (iv) strrev() with example. (AU. Nov/Dec 2011).

A String in C programming is a sequence of characters terminated with a null character '\0'. A string is a Collection of character.

* Group of character, digits and symbols enclosed within quotation(" ") marks are called as string. Null character('\0') is Used to mark the end of the string.

(i) strlen()

This function is Used to Count and return the number of characters Present in a String.

Syntax : Var = strlen(String);

Description : Var → integer Variable.
string → String Constant / Variable.

Example:

```
#include <stdio.h>
#include <conio.h>
main()
{
    char name[] = "Python";
    int len1, len2;
    len1 = strlen(name);
    len2 = strlen(" program");
    Printf("\n string length of %s is %d",
                            name, len1);
    Printf("\n String length of %s is %d",
                  "Program", len2);
}
```

Output:

```
  String length of python is 6
  String length of Program is 7
```

(ii) strcpy()

* Copy the contents of one String to another and it almost works like string assignment operator.

Syntax : strcpy(string1, String 2);

Description : String 1 → destination string
String 2 → Source string.

Example :

```
#include < stdio.h>
main ()
{
    char source = "MUNI";
    char target [10];
strcpy (target, source);
Printf ("\n Source string is %s", Source);
Printf ("\n Target string is %s", target);
}
```

Output :

```
Source String is MUNI
Target string is MUNI.
```

(iii) Strcat ()

The strcat () function is Used to Concatenate or Combine, two strings together and forms a new Concatenated String.

Syntax : Strcat (string1, string2);

Description : String1, String2 are char type arrays or constants.

Example :

```
#include <stdio.h>
main ()
{
    char Source [] = "Ramesh";
    char target [10] = "Babu";
    Strcat( Source, target );
Printf ("\n Source string is %d", source);
Printf ("\n Target string is %s", target);
}
```

Output:

```
Source string is Ramesh.
Target string is RameshBabu.
```

iv) Strrev()

* This function is Used to reverse a String. This function takes Only One argument and return one argument.

Syntax : Strrev (string);

Description : String → Character type array or string constants.

Example:

```
#include<stdio.h>
main()
{ char y[50];
    Printf(" Enter the String:");
    gets(y);
    Printf (" The string reverse is:%s",
                              strrev(y));
}
```

Output :

```
Enter the String : book
The string reverse is : koob.
```

5) Write a c program to find determinant of two matrices (2D array) which will be entered by a User. (AU. Nov/dec 2018).

Determinants of a matrix of order two can be evaluated for a square matrix of dimensions 2x2. We have to find the difference of cross multiplication of the elements.

Program:

```
#include<stdio.h>
Void main()
{
 int arr[10][10], i, j, n;
  int det =0;
  Printf ("\n\n Calculate the determinant
                of 3x3 matrix :\n");
  Printf(" Input elements in the first
                         matrix :\n");
  for(i=0; i<3; i++)
  {
      for(j=0; j<3; j++)
{
 Printf("element -[%d],[%d]:",i,j);
 Scanf("%d", &arr[i][j]);
 }
}
  Printf(" The matrix is: \n");
  for (i=0;i<3; i++)
  {
     for (j=0; j<3; j++)
        Print f ("%4d", arr[i][j]);
        Printf("\n");
  }
for (i=0; i<3; i++)
  det = det + (arr1[0][i] * (arr1[1]
          [(i+1) % 3] * arr1[2] [(i+2)%3]
        - arr1 [1][(i+2)%3] *
          arr1 [2][(i+1)%3]));
Printf ("\n The determinant of the matrix
          is: %d"\n\n", det);
}
```

Output:

```
Calculate the determinant of 3x3 matrix:
Input elements in the first Matrix:
element - [0],[0]  : 1
element - [0],[1]  : 0
element - [0],[2]  : -1
element - [1],[0]  : 0
element - [1],[1]  : 0
element - [1],[2]  : 1
element - [2],[0]  : -1
element - [2],[1]  : -1
element - [2],[2]  : 0
The matrix is :

 1    0    -1
 0    0     1
-1   -1     0

The determinant of the matrix is : 1
```

b) Explain binary search procedure. Write a c program to perform binary search and explain. (AU. Apr/May 2019).

Binary Search:

* Seach a Sorted array by repeatidly dividing the search interval is half.

* If the Value of the search key is less than the item is the middle of the interval, narrow the interval to the lower half.

* otherwise narrow it to the upper half. Repeatedly check the Value Until the Value is found or the interval is empty.

* It reduce the Complexity to $O(\log n)$

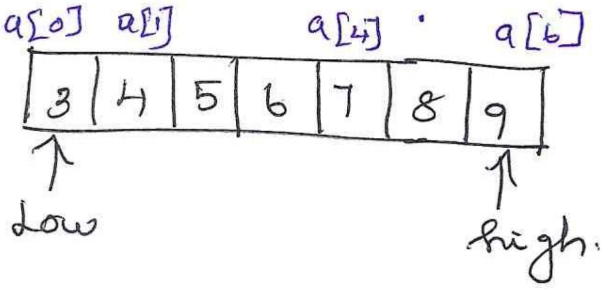
The recursive method follows the divide and conquer approach.

1. The array in which searching is to be performed is

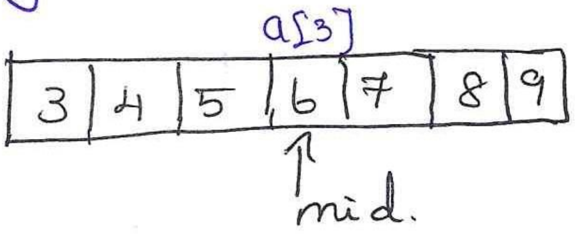
3	4	5	6	7	8	9

Let $\boxed{x = 4}$ be the element to be searched.

2. Set two pointers low and high at the lowest and highest respectively.



3. Find the middle element mid of the array. arr[(low+ high)/2]=6



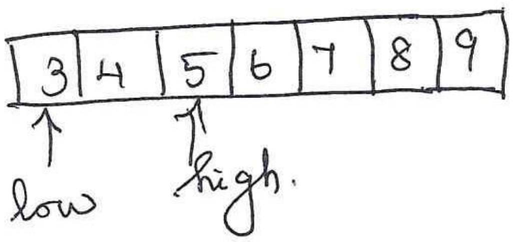
$\frac{0+6}{2} = 3$

a[3].

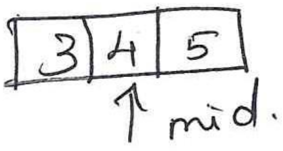
4. If $\boxed{x == mid}$ then return mid. else, Compare the element to be searched m.

5. If $\boxed{x > mid}$ compare x with mid element of right side. This is done by low = mid+1.

6. Else compare x with middle element on left side. This is done by high = mid -1.



7. Repeat step 3 to 6 Until low meets high.



8. x =4 is found.

Program:

```
#include <stdio.h>
#include <conio.h>
Void main()
{
   int a[100], i, n, low, high, mid,
                   x, f=1;
   clrscr();
   Printf("Enter the size of the array\n");
   Scanf("%d", &n);
   Printf("Enter the elements in
             ascending order");
   for(i=0; i<n; i++)
       Scanf("%d", &a[i]);
   Printf("Enter the elements to be
                           Searched");
   Scanf("%d", &x);
   low=0;
   high = n-1;
   while (low <= high)
   {
        mid= (low + high)/2;
        if (x < a[mid])
        high = mid -1;
else if (low > a[mid])
        low = mid+1;
        else
       {
         Printf("\n Entered %d is in
            Position %d", x, mid+1);
         f=2;
         break;
       }
  }
  if (f==1)
  Printf("\n Entered element is not
                in the array");
  getch();
}
```

<u>Output</u>:

```
Enter the size of the array : 7
Enter the elements in ascending order
 3  4  5  6  7  8  9
Enter the elements to be searched.
4
Entered 4 is in position 2.
```

7) Write a c program to sort the given set of numbers in ascending order using selection sort. (AU. NOV/DEC 19)

Selection Sort:

* Selection Sort is Used for Sorting. It iterates through the array and finds the smallest number in the array and swaps it with first element. if it is smaller than the first element.

* Next, it goes on to the second element and so on until all elements are sorted.

Example:

array [10, 5, 2, 1]

→ First element is 10. find the smallest number from the remaining array. So, We replace 10 by 1.

→ The new array is [1, 5, 2, 10].

→ The process is repeated and finally we get the sorted array as [1, 2, 5, 10]

Algorithm.

step 1 : Set min to the first location.

Step 2 : Search the Minimum element in the array.

Step 3 : Swap the first location with the minimum Value in the array.

Step 4 : assign the second element as min.

step 5 : Repeat the process Until we get a Sorted array.

Program:

```
#include <stdio.h>
Void main()
{
 int a[50], i, n, j, temp;
 Printf(" Enter the no. of elements");
 Scanf("%d", &n);
 for (i=0; i<n; i++)
      scanf("%d", &a[i]);
 for (i=0; i<n-1; i++)
{
    min = i;
for(j = i+1 ; j<n; j++)
{
if (a [min] > a [j])
        min = j
}
 if (min != i)
 {
   temp = a[i] ;
   a[i]  = a[min];
   a [min] = temp ;
  }
}
Printf (" Sorted Elements \n");
for (i=0; i<n; i++)
 Printf ("%d", a[i]);
}
```

Output:

```
Enter the no/. of elements
6  3  9  4  1
Sorted Elements
 1  3  4  6  9.
```

UNIT - III FUNCTIONS AND POINTERS

Modular programming - Function Prototype, function definition, function call, Built-in - functions (string functions, math functions) - Recursion, Binary Search Using recursive functions - Pointers - Pointer Operators - Pointer arithmetic - Arrays and pointers - Array of pointers - Parameter Passing: Pass by Value, Pass by reference.

PART-A

1) Define the term recursion in Language C? (AU. APR/MAY 2022, 2010, 19, 18).

Recursion is the process of calling the same function itself again and again until some condition is satisfied.

Example:
```
void recursion()
{
recursion();
}
int main()
{
recursion();
}
```

2) What is the need for functions?
(AU. NOV/Dec 2014).

* To reduce the Complexity of large Programs.
* To increase the readability.
* To achieve reusability.
* To avoid redundancy of code.
* To save money.

3) Differentiate Call by Value and call by reference. (AU. NOV/Dec. 2011).

Call by Value. The Values of the Variables are passed by the Calling function to the Called function.

Call by reference: The address of the Variables are passed by the Calling function to the Called function.

Call by Value	Call by reference.
1. Different Memory Locations are occupied by formal and actual arguments.	Same memory Location Occupied.
2. new Location is created, it is very slow.	The existing memory Location is Used thro' address is Very fast

4) What is pointer? What is the Uses of Pointers? (AU. NOV/DEC 2014, 2011, 2019)

Pointer is a Variable, contain the memory address of another Variable. It is denoted by '*' operator.

* Saves Memory Space.
* Used for dynamic memory allocation
* Faster execution.
* Used to pass array of Values to a function as a Single argument.

5) State the advantages of User defined functions Over Pre-defined functions. (AU. NOV. DEC 2011)

* 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 gives flexibility to the Programmer to Use optimal Programming instructions, not Possible in Predefined function.

6) Compare actual parameter and formal Parameter argument. (AU. Apr/May 2010)

Actual argument: Specified in the function Call statement. Used to Supply the input Values to the function either by Copy or reference.

Formal argument: Specified in the function definition statement. It takes either Copy or address of actual arguments.

7) What is the relation between the operator '&' and '*' in C pointers? (AU. APR/MAY ~~2022~~ 2005, 2014).

The '&' is the Unary operator that returns the memory address of its operand. This is also known as address of operator.

The * is a Unary operator which returns the Value of object pointed by a pointer Variable. It is known as Value of operator.

* It is also Used for declaring Pointer Variable.

8) Define typedef? (AU. Nov/Dec 2013).

The typedef keyword enables the Programmer to create a new data type name by Using an existing data type. no new data is Created rather an alternate name is given to a known data type.

Syntax : typedef existing-data type new-datatype;

9) List out any four math functions.

Pow(x,y) : Used to find power of value of x^y. Returns a double Value log10(x)

log() : Used to find natural Log Value of x.

sqrt(x) : Used to find square root Value of x.

sin(x) : returns sin Value of x.

10) When null pointer is Used? (AU. Apr/MAY 2018).

A null pointer Can never Point to a Valid data, if it is assigned to 0, for checking pointer, then it is a null Pointer, It is not pain to a Vaid memory address.

PART-B

1) Explain about the different Parameter Passing methods with ex? (AU. APR/MAY, 2022, 21, 19, 09)

When a function is called, the calling function may have to pass some Values to the called function. There are two ways in which arguments or parameters can be passed to the called function

* Call by Value * Call by reference

Call by Value:

In which Values of Variables are Passed by the calling function to the called function. When a function is called in program the Values to the arguments in the function are taken by the calling program, the Value taken can be Used inside the function.

Program:

```
#include <stdio.h>
Void swap(int, int);
Void main()
{
    int a, b;
    Printf("Enter the Values of a and b \n");
    Scanf("%d %d", &a, &b);
    Printf("Before Swap a = %d, b = %d \n",
                                a, b);
    Swap (a, b);
}
Void swap(int a, int b)
{   int temp;
    temp = a;
    a = b;
    b = temp;
    Printf ("After swap a = %d, b = %d
                  \n", a, b);
}
```

Output:

```
Enter the Values of a and b
 28   42
Before Swap   a = 28, b = 42
After swap    a = 42, b = 28.
```

Call by reference:

The process of Calling a function Using Pointers to pass the address of Variable is known as call by reference.

A function can be declared with Pointers as its arguments such functions are called by calling program with the address of a Variable as argument from it.

<u>Program:</u>

* program to interchange or swap value call by reference *

```
#include <stdio.h>
Void main()
{ int a,b;
  Void swap(int *, int *);
  Printf("Enter the Values of a and b");
  scanf("%d %d", &a, &b);
  Printf("Before swap a=%d, b=%d\n",
                                 a,b);
  Swap(&a,&b);
}
Void swap(int *a, int *b)
{
      int temp;
      temp = *a;
       *a = *b;
       *b = temp;
```

```
    Printf("After swap a=%d, b=%d",
                           *a, *b);
}
```

Output:

```
Enter the Values of a and b
   75    36
Before swap a=75, b=36
After swap a=36, b=75.
```

2) What is recursion? Write a c program to find the sum of digits and find the factorial of a number using recursion? (AU. NOV/DEC 2019)

The process is which a function call itself directly or indirectly. The function call by itself again and again to satisfied some condition is called recursion and curresponing function is called as recursive function.

Factorial Program:

```
#include <stdio.h>
Void main()
{
  int fact(int);
  int n,f;
   Printf(" Enter the number");
   Scanf("%d",&n);
   f=fact(n);
    Printf(" factorial of %d is %d", n,f);
}
int fact (int n)
{
   if (n==1)
      return 1;
   else
      return (n * fact(n-1));
}
```

Output :

```
Enter the number 5
factorial of 5 is 120.
```

factorial of n is the product of all positive descending integers.

$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.

Sum of Digits :

Reading the integer number Using the 'num' Variable. The function is Used to find sum of digits of a number(sum()).

Program:

```
#include <stdio.h>
int sum(int a);
int main()
{
 int num, result;
 Printf(" Enter the number: ");
 Scanf("%d", & num);
 result = sum(num);
 Printf(" Sum of digits is %d is %d \n",
                    num, result):
 return 0;
}
 int sum (int num)
 {
      if (num != 0)
     {
        return (num % 10 + sum(num/10));
     }
else
{
  return 0;
 }
}
```

Output :

```
Enter the number :2345
Sum of digits is 2345 is 14.
```

3) Write a C programs design the Scientific calculator Using built-in-functions (A.U 2014).

Program:

```
#include <stdio.h>
#include <math.h>
Void main ( )
{
 float x, y, result;
 int opt;
 do
 {
  Printf ("Menu \n");
  Printf ("1. sin(x) \n  2. cos (x)\n");
  Printf (" 3. tan(x) \n  4. sinh(x) \n");
```

```
Printf("5. Cosh(x)\n    6. tanh(x)\n");
Printf("7. log(x)\n    8. sqrt(x)\n");
Printf("9. exp(x)\n    10. Pow(x)\n");
Printf(" 11. exit \n");
Printf(" Enter your choice");
Scanf(" %d", &opt);
Switch (opt)
{
  Case 1 :  Printf ("Enter x \n");
            Scanf("%f", &x);
            result = sin(x);
            Printf(" sin(%f) = %f", x,
                                  result);
            break;
  Case 2 :
          Printf (" Enter x \n");
          Scanf (" %f", &x);
          result = cos(x);
          Printf(" Cos (%f)= %f ", x,
                                  result);
          break;
  Case 3 :
          Printf(" Enter x \n");
```

```
        Scanf("%f", &x);
        result= tan(x);
        Printf("tan(%f) = %f", x, result);
        break;
case 4:
        Printf("Enter x\n");
        Scanf("%f", &x);
        result = sinh(x);
        Printf(" Sinh(%f) = %f", x, result);
        break;
Case 5:
        Printf(" Enter x\n");
        Scanf("%f", &x);
        result = cosh(x);
        Printf("Cosh (%f) = %f", x, result);
        break;
case 6:
        Printf(" Enter x \n");
        Scanf("%f", &x);
        result = tanh(x);
        Printf(" tanh (%f) = %f", x, result);
        break;
```

```
Case 7 :
        Printf("Enter x \n");
        Scanf("%f", &x);
        result = log(x);
        Printf(" log (%f)=%f", x, result);
Case 8 :
        Printf("Enter x \n");
        Scanf("%f", &x);
        result = sqrt(x);
        Printf("sqrt (%f) = %f", x, result);
        break;
Case 9:
        Printf(" Enter x \n");
        Scanf ("%f", &x);
        result = exp(x);
        Printf(" exp(%f) = %f", x, result);
        break;
Case 10 :
        Printf(" Enter  x  and  y \n");
        Scanf("%f %f", &x, &y);
        result = Pow(x,y);
        Printf (" Pow(x,y) = %f", result);
}       break;
} while (option != 11);
if (option == 11)
    Printf("Exit \n");
}
```

Output:

```
Menu
1. Sin(x)     2. Cos(x)     3. tan(x)
~~4~~. Sinh(x)    5. Cosh(x)    6. tanh(x)
7. log(x)     8. Sqrt(x)    9. exp(x)
10. Pow(x)    11. Exit
Enter your option: 1
Enter x : 90.0
Sin(90.0) = 1.000000
Menu
1. Sin(x)     2. Cos(x)     3. tan(x)
4. Sinh(x)    5. Cosh(x)    6. tanh(x)
7. log(x)     8. sqrt(x)    9. exp(x)
10. Pow(x)    11. Exit
Enter your option: 11
Exit.
```

4) What is Modular programming? How does the Language C support modular programming? Explain in detail? (AU. APR/MAY 2022).

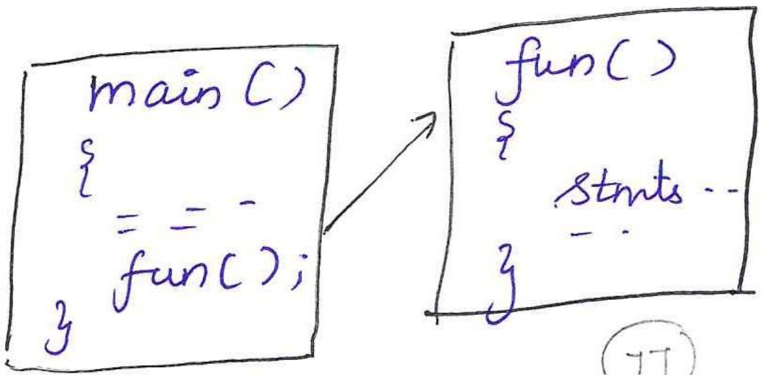
Modular programming is defined as a Software design technique that focuses on separating the program functionality into independent, interchangeable modules.

Function:

Function is a set of instructions that are Used to perform Specified tasks ~~with~~ which repeatedly Occurs in main Program.

Modules are Created and merged with compilers, in which each module Performs an operation within the Program. functions are classified into two types.

1. User Define functions.
2. Built-in functions



```
syntax: return_type function_name (argument
                                        list)
        {
            set of statements - Block of code
        }
```

* C programming Language divides the Problem into smaller modules called function.
* Each function defined in C by default is globally accessible.

__User defined function__

The function defined by the Users according to their requirements are called User defined functions. User can modify the function according to their requirements.

Elements of User defined function

* function declaration
* function definition
* function call.

__function declaration__

Declaration statement that identifies a function with its name, a list of argu-ments that it accepts, and type of data it returns.

Syntax:

return datatype functionname (datatype Var1,
datatype Var2--);

function name → is a Valid name for the function, Used to call it for execution in a program.

return datatype → data type of Value return to the Calling function as a result of processing performed by the Called function.

datatype Var1, Var2 → List of Variables of Specified datatypes. These Variables are Passed from Calling function to Called function. They are also known as arguments (or) Parameters.

<u>Function definition:</u>

When a function is defined, Space is allocated for that function in the memory.

~~the~~ Syntax:

```
returndatatype function name (datatype Var1,
                              datatype Var2...)
{
    statements
    = = .
    return(var);
}.
```

It is not followed by a semicolon.
This function header is known as the formal Paratmeter list.

Function call:

The function call statement invokes the function. when a function is invoke the compiler jumps to called function.

Syntax: functionname (Var1, Var2, ---);

Example:

```
#include <stdio.h>
int sum( int a, int b)
{
    return a+b;
}
```

function definition

```
int main ()
{
int add = sum (10,30);
Printf(" sum is %d",
            add);
return 0;
}
```

function call

Function return value.

Built-in functions:

C Language provides built-in functions called library functions. The compiler itself evaluates these functions.

The Library provides a basic set of mathematical functions, String Manipul-ation, type conversions, a file and Console base I/o.

5) Classify the function Prototypes with example C program for each (AU. 2014).

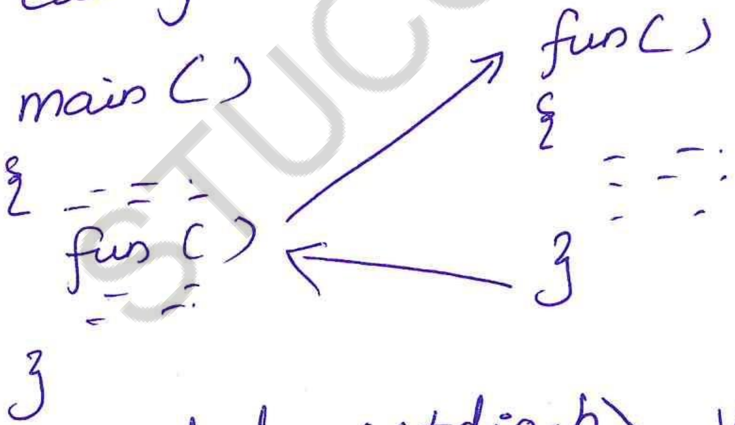
A function prototype is a function declaration that specifies the data types of its arguments in the parameter list. The functions are classified into the following types depending on whether the arguments are Present or not.

A function prototype declaration Consists of the function's return type, name and arguments itself and terminated with Semicolon.

* Function with no arguments and no return value.
* Function with argument and no return value.
* Function with arguments and with return value.
* function with no arguments and with return value.

Function with no arguments and no return value

No data transfer take place b/w the calling function and called function.



(eg):
```
# include <stdio.h>
Void sum();
Void main()
{
    Void sum();
    sum();
}
```

```
Void sum()
{ int a,b,c;
  Printf("Enter 2 values \n");
  Scanf("%d%d",&a,&b);
```

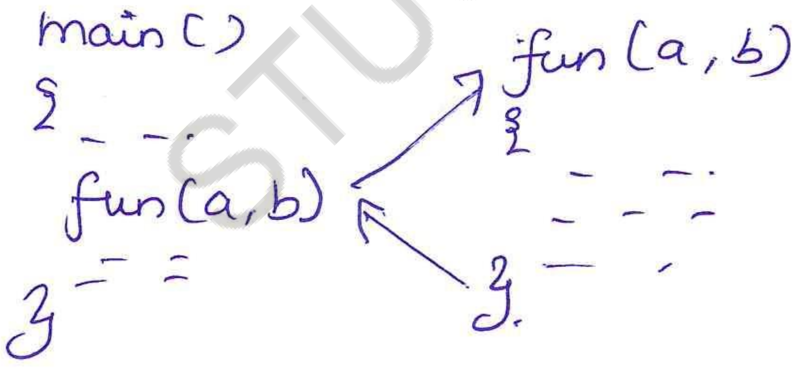
```
    c=a+b;
    Printf("Sum = %d", c);
}
```

Output:

```
Enter 2 Values 50 60
Sum = 110.
```

ii) Function with arguments and no return Value.

* Data is transferred from Calling function to Called function. The called Program receives some data from the Calling Pgm. and does not send back any values. to the Calling program.



Program:

```
#include<stdio.h>
Void sum(int, int);
Void main ()
{
```

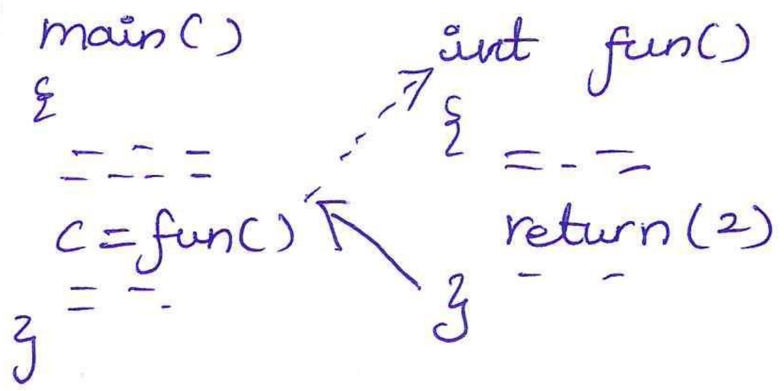
```
Void sum (int, int);
int a, b;
 Printf(" Enter 2 Values \n");
 Scanf(" %d %d", &a, &b);
 Sum(a, b);
}
void sum ( int a, int b)
{
    int c;
    c=a+b;
    Printf(" sum = %d", c);
}
```

Output:

```
Enter 2 Values   26   3
Sum = 29
```

iii) Function with no arguments and with return Value.

The calling program cannot pass any arguments to the called function Program, but the called program may send some return Value to the calling program. (One way Communication).



Program:

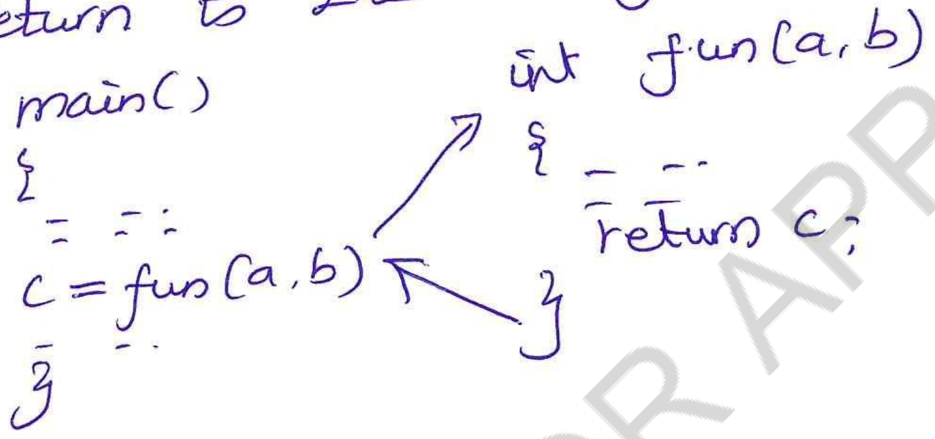
```
#include <stdio.h>
int sum();
Void main()
{
  int sum();
  int c;
  C=sum();
  Printf("sum=%d",c);
}
int sum()
{
  int a,b,c;
  Printf("Enter 2 Values\n");
  Scanf("%d %d", &a,&b);
  c=a+b;
  return c;
}
```

Output:

```
Enter 2 Values 10 20
Sum =30.
```

iv) Function with arguments and with return value.

* The data is transferred between the Calling function and Called function. The Calling function receive some data from Called program and send back a Value Called program return to the Calling Program.



```
main()                    int fun(a,b)
{                         {  - - -
  - - :                      return c;
  c = fun(a,b)            }
  - --
}
```

Program:

```
#include <stdio.h>
int sum(int, int);
Void main()
{
  int sum(int,int);
  int a,b,c;
  Printf("Enter 2 Values");
  Scanf("%d %d",&a,&b);
  c = Sum(a,b);
  Printf("Sum =%d", c);
}

int sum(int a,
        int b)
{
    int c;
    c=a+b;
    return c;
}
```

Output: Enter 2 Values 10 25   Sum = 35

6) Explain about pointers and Write the Use of pointers in arrays with Ex? (A.V. 2014)

* Pointer is a Variable that stores / Points the address of another Variable.

* It is Used to allocate memory dynamically (ie). at run time.

Syntax:

datatype *Var_name ;

Example: int *p ; char *p ;

*p → Pointer Variable.

Program:

```
#include<stdio.h>
Void main()
{
  int num, *ptr;
  Ptr = & num;
  Printf(" Enter the number\n");
  Scanf(" %d ", &num);
  Printf(" The number that was
             entered is %d ", *ptr);
}
```

Output:

```
Enter the number 50
The number that was entered is 50
```

In this program, get the value of num = 50, then, ptr value is the address of num variable.

An array of pointers can be declared as

```
int *ptr[10];
```

It is a collection of addresses.

→ All pointers in an array must be of the same type.

→ To access elements of the array, we have used pointers.

Program:

```
#include <stdio.h>
int main()
{
    int i;
    int a[5] = { 1, 2, 3, 4, 5};
    int *p = a;          // same as
                            = &a[0]
    for (i=0; i<5; i++)
    {
        Printf("%d", *p);
        P++;
    }
  return 0;
}
```

Output:

```
1, 2  3  4  5.
```

7) Write a c program to find binary search elements, and fibonacci Series Using recursion? (AV. APR/MAY 2014, 2015)

Binary Search:

Binary Search is defined as a searching algorithm Used in a stored array by repeatedly dividing the Search interval in half. It is based on divide and Conquer approach.

Program:

```
#include <stdio.h>
#include <stdlib.h>
int binary(int[], int, int, int);
Void main()
{
  int n, i, x, Pos;
  int low, high, a[50];
  Printf("Enter the total number of
              elements");
  Scanf("%d", &n);
  Printf("Enter the array elements
              in sorted order");
  for (i=0; i<n; i++)
    scanf("%d", &a[i]);
  low=0;
  high=n-1;
  Printf("Enter element to be search");
  scanf("%d", &x);
  Pos= binary(a, x, low, high);
  if (Pos != -1)
      Printf(" Number found at Position
                              %d", Pos+1);
  else
      Printf(" The number is not Present
                         in the list");
}
int binary(int a[], int x, int low,
                              int high)
{
  int mid;
  if (low > high)
       return -1;
  mid = (low + high)/2;
  if (x == a[mid])
       return(mid);
  elseif (x < a[mid])
       binary(a, x, low, mid-1);
  else
       binary(a, x, mid+1, high);
}
```

<u>Output:</u>

```
Enter number of elements 5
Enter the array elements in sorted
                              order
     5   8   12   16   23
Enter elements to be search 16
Number found at position 4.
```

Fibonacci Series - Recursion.

```
#include <stdio.h>
int fib(int );
int main()
{
 int n,i=0,res;
 Printf("Enter the
   number of terms\n");
scanf("%d",&n);
Printf("fibonacci
        Series \n");
for(i=0; i<n; i++)
{
   res = fib(i);
   Printf("%d \t", res);
  }
}
int fib(int n)
{
   if(n==0)
       return 0;
   else if (n==1)
       return 1;
else
  return(fib(n-1)+
        fib(n-2));
}
```

Output:

```
Enter the number
of terms. 6
0 1 1 2 3 5
```

fibonacci series, defined the next number is the sum of previous two numbers.

The first two number of fibonacci series are 0 and 1.

same type. for example single linked list is a self referential data structure

eg:-
```
typedef struct node
{
    int data;
    struct node * next;
} linked list;
```

3. In language c can we allocate memory dynamically? How? (AU-2021)

- In c language, we can allocate the memory at runtime.

- Dynamic memory allocation in c language is possible by 4 functions of stdlib.h header file.

1. malloc()
2. calloc()
3. realloc()
4. free()

4. Differentiate structure and union (AU-2021)

Structure	Union
Every member has its own memory.	All members used in same memory.
The keyword used is structure	The keyword used is union.

UNIT-IV STRUCTURES AND UNION

Structure - Nested structures - pointer and structure Array of structure - self referential structure - Dynamic memory allocation - Singly linked list - typedef - union - storage classes and visibility.

PART-A

1. What is meant by structure definition (AU 2022)

- A structure definition is usually defined near to the start of a file using typedef statment.

- typedef defines the names a newtype allowing its use throughout the program. It occur just after the # define and #include statments in a file.

eg:-
```
typedef struct
{
    char name[60];
    char course[100];
    int age, year;
} student;
```

2. Define self referential structure. (AU-2022)

- A self referential structure is one of the data structure which refers to the pointer to (points) to the another structure of

members occupy Separate memory location consumes more spaces compared to union.	All members are ocupy in same memory. Conservation of memory is possible.

5. what is meant by union. in c ? (AU-2012)

- A union is a special data type available in c that enables you to store different data types in the same memory location.

- Union provide an efficient way of using the same memory location for multi- purpose.

6. what are the storage classes available in c ? (AU-2020)

- A storage class define the scope (visibility) and life time of variables and /or functions with in a c program.

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

7. Define macro in C (AU - 2011)

- 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.

eg:-
```
#define tablesize 100
int tablesize1 [20];
int tablesize2 [40];
```

8. How to create node in linkedlist (AU-2008)

- We create the node using the malloc function.

```
newnode = malloc (size of (node));
```

9. Write the syntax for pointers to structure. (AU - 2012)

```
struct s
{
    char name [10];
    int age;
    float salary;
};
struct s *sptr // sptr is pointer of s.
```

10. Compare array and structure (AU-2018/AU2010

Array	Structure.
An array is a collection of data items of same data type.	A structure is a collection of data items of different datatypes.
Arrays can olnly to be declared.	Structures can be declared and defined.
There is no keyword for arrays.	The keyword for structure is struct

11. Define singly linked list. (AU-2010)/AU(2008)

- It is a linear data structure.

It is a type of list.

- In SLL, each node in the list store the content of the node and a pointer or reference to the next node in the list.

```
    struct node
     {
        int data;
        struct node *next;
     };
```

PART-B

1) What is the purpose of the concept 'structure' in language c? Explain in detail with an example program. (AU 2022)

Purpose of structure in c :-

1. structure is a user defined data in c language will allows us to combine data of different types together.

2. structure helps to construct a complex data type which is more meaningful.

3. It is similar to an array, but an array hold data of similar datatype, which structure hold the data of any type.

4. Each element of structure is called a member. It can store various information.

5. The struct keyword is used to define the structure in c language.

Uses of structure :-

1. c structure can be used to store huge data.

2. Structure can be used to send data to the pointer.

3. C structure can be used to check computer memory size.

<u>structure declaration</u>:-

- A structure is declared using the keyword struct.

<u>Syntax</u>:-

```
struct structure-name
{ datatype var1;
  datatype var2;
     :
};
```

- The structure declaration doesnot allocate memory. It gives detail of the members name. Once the structure variable is declared then the memory is allotted for the structure.

```
struct student                    struct student
{ char name[30];                  {  char name[10];
  int regno;                         int regno;
  int marks;            (or)      } s;
  float avg;
};
struct student s;
```

Typedef declarations:-

- The typedef keyword enables the Programmer to create a new datatype name from an existing datatype.
- By using typedef, no new data is created. rather than alternate name is given.

Syntax:-

```
typedef existingdatatype newdatatype.
```

eg:-

```
typedef struct student
{ int integer;
  char name[20];
  int marks, total;
} stu;
```

Intialization of Structure:-

- A structure can be intialized in the same way as other data types are intialized.

Syntax:-

```
struct structure name
{
 datatype var1;
 datatype var2;
 ;
}struct var = {value1,val2..};
```

eg:-

```
struct student
{ int regno;
  int marks;
} stu {23, 100};
```

Accessing the member of a structure :-

- A structure member variable is generally accessed using a '.' (dot) operator.

eg:- stu.regno = 523;
stu.name = "Abi";
stu.mark = 98;

_ the input value for datamember of structure variable stu,

scanf("%d", & stu.regno);

- To print the values of structure variable stu, we may write,

printf("%d", stu.regno);

To read and display the information about Student :-

```
#include <stdio.h>
struct student
 {  int regno;
    char name[10];
     float marks;
  } stud;
void main()
 { printf("Enter regno");
scanf("%d",&Stud.regno);
Printf(" Enter the student name");
scanf(" %s", Stud.name);
Printf(" Enter marks");
scanf(" %d", & Stud.marks");
printf(" Student Details");
Printf(" Regno = %d", stud.regno);
printf(" Name = %s", Stud.name);
printf(" Marks = %f", Stud.marks);
}
```

2. Write short notes on 'Array of Structures' (AU 2022)

- If we want to handle more records within one structure, we create the structure variable as an array. That kind of declaration is called array of structure.

eg:-
```
struct student
{
    int rollno;
    int s1, s2, s3;
} Stud[5];
```

Write a program to read and display the information of all the students in the class.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
struct student
{ int rollno, fees;
   char name[10];
   char dob[10];
} stud[20];
voidmain()
{
  int i,n;
 prinf ("Enter the no of
          students");
 scanf("%d",&n);
 for(i=0;i<n;i++).
 {printf("Enter rollno");
  scanf("%d",&stud[i].
           rollno);
 Printf("Enter Studentname");
 Scanf("%s",stud[i].name);
 Printf("Enter fees");
 Scanf("%d",&stud[i].fees);
 printf("Enter dob");
 gets(stu[i].dob);
 }
 printf("Student detail");
 for(i=0;i<n;i++)
 {
 printf("Name=%s",stud[i].
           name);
 Printf(" Rollno=%d",stud[i]
           rolno;
 Printf("fees=%d",stud[i].fees);
 Printf("DOB=%s",stud[i].dob);
 }
}
```

5. Write a Comparative analysis of Various storage classes in C language? (AU-2022)

The storage classes are used to define the scope end life time of the variable and function.

- There are four types of storage classes which are available.

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

auto :-

- The auto is default for function (or) variable.

- The storage is automatically allocated in function entry.

- The auto can be used only within a function. ie, local variable.

eg:-
```
void main ()
{
  int a; sum;
  auto int b;
  sum = a+b;
  printf ("%d", sum);
}
```

register :-

— The register is used to define local variable that should be stored in register instead of RAM.

— Register should only be used for variables that require quick access such as counters.

eg:-
```
void main ()
{ register int count;
}
```

static :-

— The static is the default storage class for global variables.

— The static can be defined within a function. If it defined within a function, the lifetime is depend to life time of the program.

eg:-
```
void main ()
{ int a =1
  int sum ();
  sum ();
}
int sum ()
{ static int b = 2;
  sum = a+b ;
  return sum ;
}
```

extern :-

- If a variable is declared (with global Scope) in one file but referenced in another then extern keyword is used to inform the Compiler of the variable existence.

- An extern declaration doesnot create any storage.

eg:-
```
extern int a=10;
void main()
{
 int b, sum;
 sum = a+b;
 printf("%d", sum);
}
```

4. Write a short notes on Nested structure with example program. (AU - 2018)

A structure that Contains another Structure as its member is called as Nested Structure. ie, the structure can be Placed within another Structure.

Syntax:-

```
struct structure-name1
{ data members;
        :
  };
 struct structurename 2
 {
      . . .
      data members;
      struct structurename1 var1;

   } var2;
```

Program to read and display information of students using structure within structure (or) Nested structure.

```
#include< stdio.h >
struct DOB
{  int date, month, year;
 };
struct student
 { int rollno;
   char name[50];
       float avg;
   struct DOB d;
 } stud;
```

```
void main()
{
printf("Enter rollno & name");
scanf("%d%s", &stud.rollno, stud.name);
Printf(" Enter avg marks");
scanf("%f", &stud.avg);
printf(" Enter the date of birth");
scanf(" %d%d%d", &stud.d.date, &stud.d.
                                month, &stud.d.year);

printf(" Student Details");
Printf(" Name=%s",\n Rollno=%d", stud.name,
                                stud.rollno);
Printf(" Avg=%f", stud.avg);
Printf(" DOB=%d%d%d ", stud.d.date, stud.d.
                                month, stud.d.year);
}
```

5. Explain about pointers and structure with example program. (AU-2012)

Structure can be created and accessed using pointers. when we have a pointer of structure type we use (→) to access the structure member.

Syntax:-

```
struct structure_name;
{
    data members;
        :
  } *ptr;
```

- Next we create one structure variable to assign the address.

```
struct structurename s;
 ptr = &s;
```

- Then we can access the data members by using pointing-to-operator →

Program by using array of pointer to a structure, to read and display the data of student.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct student
{
    int rollno;
    char name[20], dept[10];
 } *ptr[10];
```

```
void main()
{ int i,n;
 printf(" Enter the no of students");
 scanf(" %d", &n);
 for(i=0; i<n; i++)
  { ptr[i] = (struct student *) malloc (sizeof
                                   (struct Student));
    printf (" Enter the data");
    printf(" Enter rollno");
    scanf(" %d", &ptr[i] -> rollno);
    printf(" Enter name");
    scanf(" %s", ptr[i] -> name);
    printf(" Enter dept");
    scanf(" %s", ptr[i] -> dept);
  }
 printf(" Detail of student ");
 for(i=0; i<n; i++)
  { printf(" Rollno=%d", ptr[i] -> rollno);
    printf(" Name = %s", prt [i] -> name);
    printf(" Dept = %s", ptr[i] -> dept);
  }
}
```

6. Explain in briefly about dynamic memory allocation in c language? (AU-2011)

– The concept of dynamic memory allocation in c language enables the c programmer to allocate memory at runtime.

– Dynamic memory allocation in c language is possible by four functions of stdlib.h header file.

1. malloc()
2. calloc()
3. realloc()
4. free()

malloc() :-

– The malloc function allocates single block of requested memory.

– It doesn't intialize memory at execution time. It returns NULL if memory is not sufficient.

Syntax:-

```
ptr = (cast-type *) malloc (byte-size)
```

eg:-

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ int n, i, *ptr, sum=0;
Printf(" Enter no of elements");
scanf("%d", &n);
ptr= (int*) malloc(n*sizeof(int));
if( ptr == NULL)
 {
 printf(" Sorry! Unable to allocate ");
 exit(0);
 }
else
 printf(" Memory allocated successfully");
printf(" Enter elements of array");
for(i=0; i<n; i++)
  { scanf("%d", ptr+i);
      sum+ = *(ptr+i);
   }
   printf(" sum=%d", sum);
    free(ptr);
  return 0;
}
```

calloc() :-

- The calloc() function allocates multiple block of requested memory.
- It returns NULL if memory is not sufficient.

syntax:-

```
ptr=(cast type*)calloc
        (number, byte)
```

eg:-

```
#include <stdio.h>
#include<stdlib.h>
int main()
{
 int n, i, *ptr, sum=0;
 Printf(" Enter elements
          count");
 scanf(" %d", &n);
 Ptr = (int*) calloc (n,
          sizeof(int));
 if( ptr == NULL)
 {
 printf(" sorry! Unable
     to allocate memory");
 exit(0);
 }
 else
 Printf(" Memory allocated
              succesfully");
 Printf("Enter elements");
 for(i=0; i<n; i++)
 { scanf("%d", ptr+i);
   sum+ = *(ptr+i);
 }
 Printf(" sum=%d", sum);
 free(ptr);
 return 0;
}
```

o/p

```
Enter elements count 3
Enter elements
        10
        10
        10
 sum= 30
```

realloc() :-

- If memory is not sufficient for malloc() or calloc(), you can reallocate the memory by realloc() function.

- i.e, it changes the memory size.

Syntax :-

```
ptr = realloc(ptr, new_size)
```

free() :-

- The memory occupied by malloc() or calloc() functions must be released by calling free function. otherwise, it will consume memory until program exit.

Syntax :-

```
free(ptr);
```

7. Write c program to create mark sheet for students using self referential structure. (AU-2022)

self referential structure :-

- A self referential structure is used to create data structure like linked list, stacks etc.

Syntax:-

```
struct struct_name
{ datatype  datatypename;
   struct name * pointer name;
};
```

eg:-

```
typedef struct node
 {   int data
     struct * node;
  } linked list;
```

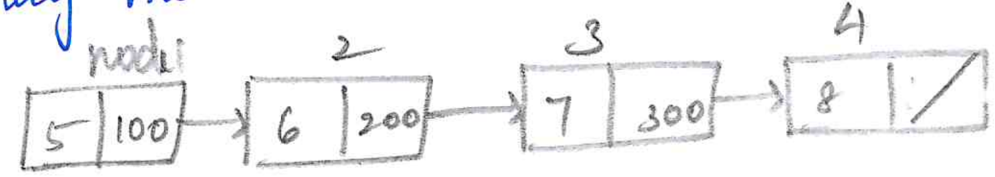
<u>Program for student mark sheet using self referential structure</u>

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
struct student
{   int rollno;
    char name[10];
     int m1, m2, m3 ;
    struct student *link;
 };
```

```
Void main ()
{
 Struct student obj1;
 Struct student obj2;
 clrscr();
 Object1. link = NULL;
 Obj1. rollno = 10;
 strcpy(obj1.name,'Abi');
 Obj1. m1 = 100;
 Obj1. m2 = 100;
 Obj1. m3 = 95;
 Obj2. link = NULL;
 Obj2. rollno = 20;
 strcpy (obj2.name, 'Bala');
 Obj2. m1 = 90;
 Obj2. m2 = 95;
 Obj2. m3 = 100;
 Struct student obj3;
 obj3. link = NULL;
 Obj3. rollno = 30;
 ~~ob~~ strcpy (obj3.name, 'magi')
 obj3. m1 = 100;
 obj3. m2 = 90;
 Obj3. m3 = 92;
 Obj1. link = &obj2;
 obj2. link = &obj3;
 printf ("%d", obj1.link->
                    rollno);
 Printf("%d", obj1.link->m1);
 printf("%d", obj1.link->m2);
 printf("%d", obj1.link->m3);
 Printf ("%d", obj2.~~link~~->rollno);
 printf("%s", obj2. link->name);
 Printf("%d %d %d", obj3.link->
        m1, obj3.link->m2, obj3.link
                        ->m3);
 getch();
}
```

8. Why is singly linked list called as self referential structure? (AU-2021)

- A linked list is a way to store a collection of elements. like an array there can be char or integers.



Syntax:-

```
struct linkedlist
{   int data;
    struct linkedlist *next;
};
```

- The above definition is used to create every node in the list. the data field stores the element and the next is pointer to store the address of next node.

- In place of datatype, struct linked list is written before next. That's because its a self referencing pointer. It means the pointer that point to whatever it is a part of. Here next is pointer, point to next node.

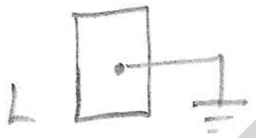
Singly linked list Implementation :-

Type declaration :-

```
struct node
{ int data;
  struct node * next;
} * head = NULL;
typedef struct node * position;
```

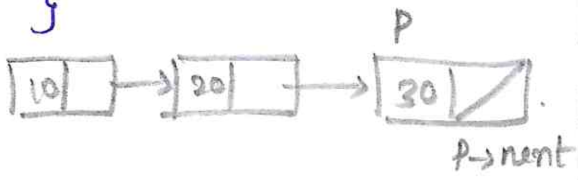
Routine to check whether the list is empty

```
int isempty (position head)
{ if (head == NULL)
    return 1;
}
```



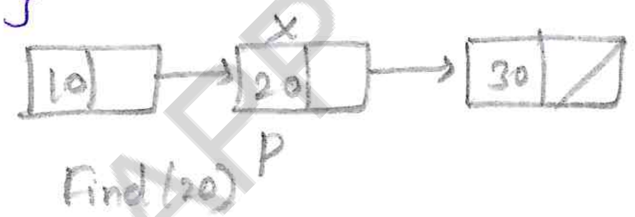
Routine to check whether the current position last

```
int islast (position p)
{
  if (p -> next == NULL)
    return 1;
}
```



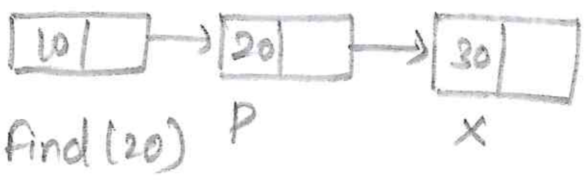
Find Routine :-

```
Position Find ( int x)
{ position p;
  P = head -> next;
  while(( P != NULL) && (P -> data != x
  { P = p -> next;
  }
  return p;
}
```



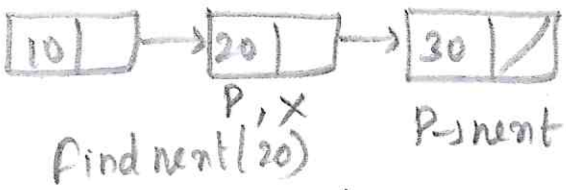
Find previous Routine :-

```
Position Findprevious ( int x)
{ position p;
  P = head;
  while(( p -> next != NULL) &&
        (p -> next -> data != x))
  { p = p -> next;
  }
  return p;
}
```



Find next Routine :-

```
Position find ( int x)
{ position p;
  P = head -> next;
while((p!=NULL)&&
      (p->data!=x))
{
    p=p->next;
}
return p->next;
}
```

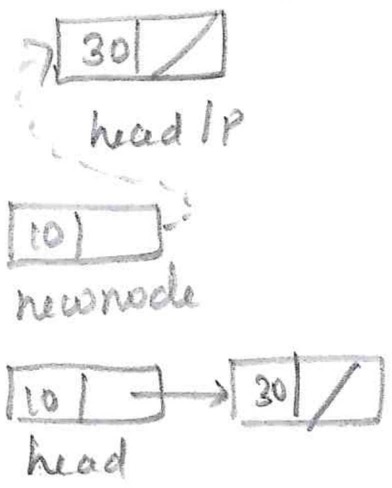


Traversal / Display

```
void traversal
{ position p;
  p=head->next;
while(p!=NULL)
{ printf(p->data);
  p=p->next;
}
}
```

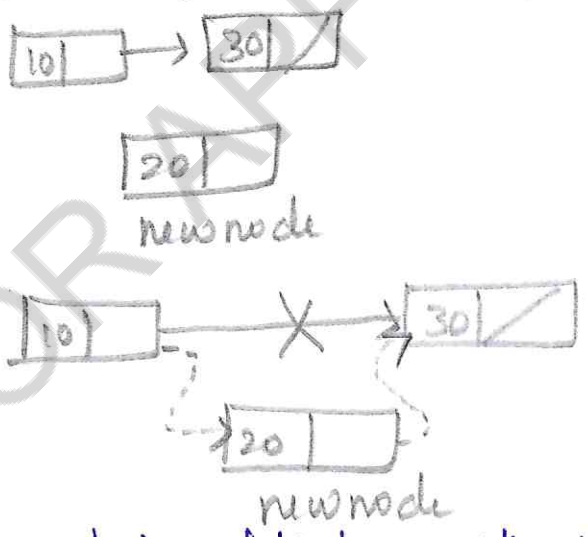
Insertion :-

Inserting node to the front of list :-



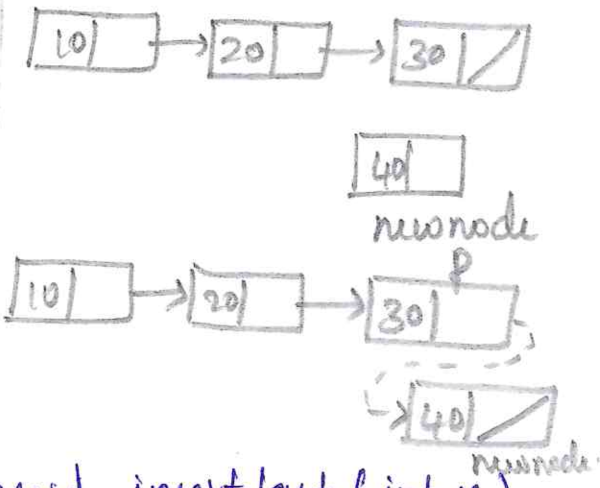
```
Void insert_beg(int x)
{ position Newnode;
Newnode=malloc(sizeof
         (Struct Node));
if(Newnode!=NULL)
{ Newnode->data=x;
  Newnode->next=L->next;
   head=Newnode;
}
}
```

Inserting node in Middle:-



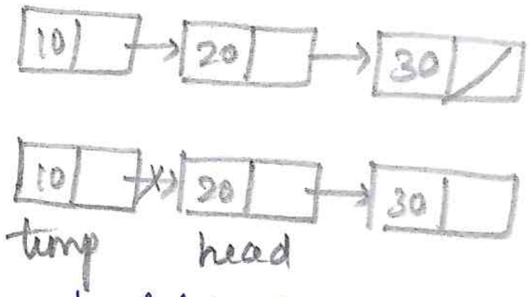
```
Void insert(int x, position p)
{ position new; int key;
new=malloc(sizeof(Structnode));
if(new!=null)
  { if (p->data==key)
  { new->next=p->next;
     p->next=new;
  }
else
   P=p->next;
 }
}
```

Inserting node to end :-



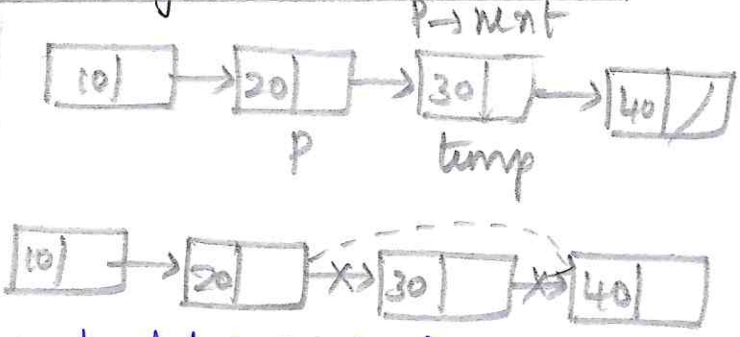
```
void insertlast (int x)
{
  position P, new;
new = malloc(sizeof (struct node));
if ( new != NULL)
{ while (p->next != NULL)
    {
      p = p->next ;
    }
  new->data = x ;
  new->next = NULL;
  p->next = new;
}
}
```

Delete at front of list :-



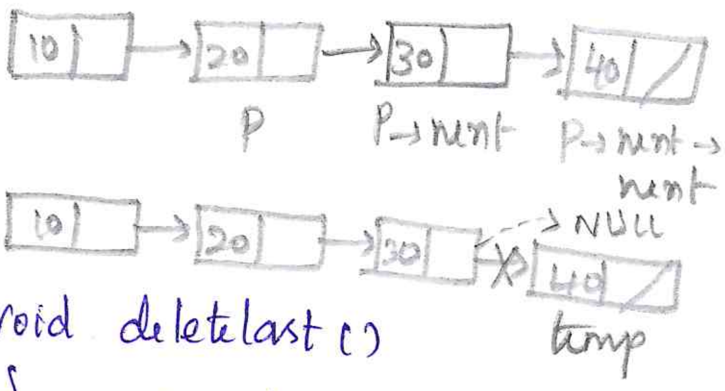
```
Void delete_beg()
{   position temp;
if ( head != NULL)
   {  temp = head;
      head = temp->next;
      free (temp);
   }
}
```

Deleting the middle node :-



```
void delete (int x)
{ position p, temp;
P = findprevious (x);
  temp = p->next ;
P->next = temp->next;
  free ( temp);
}
```

Deleting last node :-



```
void deletelast ()
{  position temp, P;
while (p->next->next != NULL)
{  P = P->next ;
}
temp = P->next ;
   P->next = NULL;
}  free (temp);
```

UNIT-V FILE PROCESSING

Files - Types of file processing: Sequential access, Random access - sequential access file - Random access file - Command line arguements.

Part-A

1. Why are files needed? (AU-2022)

- when program is terminated, the entire data is lost. storing in a file will preserve your data even if the program terminates.
- 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. Give an example for fseek(). (AU-2022)

1. fseek(p, 10L, 0)

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

2. fseek(p, -5L, 1)

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

3. What will be the impact if fclose() function is avoided in a file handling C program? (AU-2021)

– If an open file is not closed correctly, lead to data loss of the entire file content. However it varies, depending on the OS.

1. When file was not closed correctly before the program is terminate normally, the OS will try to close the file. In many cases this can prevent the data loss.

2. When file not closed, the program is terminated unexpectedly, by crash, the loss of data can hardly be prevented.

4. What are command line arguments? (AU-2021) / AU (2018)

– Command line arguments

are the values given after the name of the program in the command-line shell of operating system.

– Command-line arguements are handled by the main() function of C

Syntax:- int main(int argc, char *argv[])

5. Give an example of rewind() (AU-2010)

rewind() function is used to move the file pointer to the beginning of the given file.

syntax:- rewind(fptr);

6. write short note on fread() and fwrite (AU-2012)

– fwrite() is used to writing an entire block to a given class.

syntax:- fwrite(ptr, size, nst, fptr);

– fread() is used to reading an entire block from a given file.

syntax:- fread(ptr, size, nst, fptr);

7. How can you restored a redirected standard stream? (AU-2018)

– By using the standard

c library functions named dup()
and fdopen(), you can restore a
standard stream such as stdout to
its original state.

– The dup() function duplicates
a file handle. The fdopen() function
opens a stream that has been duplicated
with the dup() function.

8. List the file operations in C paradigm?
(AU – 2019)

– In c, you can perform four
major operations on the file, either
text or binary.

1. creating a new file.
2. opening a existing file.
3. closing an file.
4. Reading from and writing information
to a file

9. What are two main ways a file can
be organized? (AU – 2019)

– In c, the ~~two~~ File can be
accessed in two ways.

* Random access
* Sequential access.

10. write the functions for random access file processing. (AU-2015)

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

11. Write a short note on fseek() (AU-2016)

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

syntax:- fseek(filepointer, displacement, pointer position.

filepointer - It is pointer which point to file.

displacement - It is positive or negative.

pointer position-	0	Begining of file
	1	Current position
	2	End of file.

PART-B

1. What is file? what are facilities available in language C to handle files? Explain. (AU-2022)

File:-

File is a collection of data stored on a secondary storage device.

Streams in C:-

Stream is a logical interface to the devices that are connected to the computer.

* Standard input (stdin)
* Standard output (stdout)
* Standard error (stderr)

Standard input:- is the stream from which the program receives its data.

Standard output:- is the stream where a program writes its output data.

Standard error:- is basically an output stream used by programs to report error messages.

Types of files:-

- In c, the types of files used can

be broadly classified into two catigories.

- Text files
- Binary files.

Text file :-

- A text file stream of character that can be sequentially processed by a compiler in a forward direction.

- A text file usually opened for only one kind of operation at any given time (creating, reading, writting, appending).

- In a text file, each line contains zero or more characters and ends with one or more characters that specify end of line.

Binary file :-

- A binary file may contain any type of data, encoded in binary form for computer storage. and processing purposes.

- while text file can be processed sequentially, binary files on other hand, can be either processed sequentially (or) randomly depending on the needs of application.

– To use files in c, we must follow the steps.

* Declare a file pointer variable.
* open a file
* process the file
* close the file.

Declaring a file pointer variable:-

– In order to access a file, we must specify the name of file, that has to be used.

– This is accomplished by using file pointer variable that point to a structure FILE.

Syntax:-

```
FILE *fpvar;
```

eg:- `FILE *fp;`

Opening a file:-

– A file must first be opened before data can be read from it or written to it.

- Inorder to open a file and associate it with a stream, the fopen() function is used.

Syntax:-

```
FILE *fopen(const char* filename, const
                              char * mode);
```

File modes:-

Mode	Description
r	open text file for reading
w	open text file for writting.
a	Append to a text file
rb	open binary file for reading
wb	open binary file for writting
ab	Append to binary file.
r+	open text file for both reading and writting
w+	open text file for both reading and writing.
a+	open text file for both reading and writting.

eg:-

```
  File *fp;
 fp=fopen ("student.txt", "r");
  if( fp = = NULL)
  {
printf(" the file couldnot opened");
 exit(1);
}
```

<u>closing a file:-</u>

— The fclose() function not only closes the file but also flushes all the buffers that are maintained for that file.

<u>Syntax:-</u> fclose (fp);

---

2. Explain the functions of following
 i) Reading data from a file
 ii) writting data to a file (AU-2012)

<u>Read data from files:-</u>

C provide the following set of function to read the data from a file.

* fscanf * fgetc()
* fgets() * fread()

fscanf :-

- It is used to read formatted data from the stream.

Syntax :-

fscanf (FILE *stream, const char * format, ..);

eg :-

```
#include <stdio.h>
void main()
{
  FILE *fp;
   char name[30];
    int rollno;
 fp= fopen ("student.txt","r");
 if ( fp== NULL)
 {
 printf(" The file couldnot be
          opened");
  exit(0);
 }
 printf (" Enter the name & rollno);
 fscanf (stdin, "%s %d", name, &rollno);
  printf ("name = %s \t Rollno=%d", name, rollno);
  fclose (fp);
  }
```

fgets() :-

- The function fgets() is used to get a string from a file.

Syntax :-

```
Char *fgets (char *str, int size, FILE *stream);
```

- The fgets() function read atmost one less than the number of characters specified by size. from the given stream.

- The fgets() terminate as soon as it encounter either a newline character, EOF or anyother errors.

eg:-

```
#include <stdio.h>
void main()
{ FILE *fp;
  char str[80];
  int i, ch;
fp = fopen("add.c", "r");
if ( fp == NULL)
{
printf("The file couldnot
              opened");
exit(1);
}
while (fgets(str, 80, fp) != 
                    NULL)
{
Printf("\n %s", str);
Printf(" close the file");
fclose(fp);
}
```

O/p

```
abcdef....
close the file.
```

fgetc():-

- fgetc() is takes the character from the file and displays it. Only one character is read at a time.

Syntax:-
```
int fgetc ( FILE *pointer) ;
```

eg:-
```
#include <stdio.h>
#include<stdlib.h>
int main ()
 { int ch;
   FILE *fp;
   fp= fopen (" data.txt","r");
   if( fp== NULL)
   { printf(" Error opening file");
     exit(1);
   }
   printf("Reading content of data.txt");
  while(( ch =fgetc (fP)! = EOF)
   { printf("%c", ch, ch);
   }
   fclose( fp);
   return 0;
 }
```

fread() :-

- The fread() function is used to read data from a file.

Syntax :-

```
int fread(void *str, size_t_size, size_t_num,
                                   FILE *stream);
```

eg:-

```
#include<stdio.h>
void main()
{
    FILE *fp;
    char str[10];
fp=fopen("sample.txt","r+");
if(fp==NULL)
{ printf("The file not opened");
  exit(1);
}
fread(str,1,10,fp);
str[10]='\0';
Printf("First 9 characters
of the file are %s",str);
fclose(fp);
}
```

O/p:-

```
The First 9 characters
of the file are
        hello how
```

Writing data to file :-

C provides the following set of functions to read data from a file.

* fprintf()
* fputs()
* fputc()
* fwrite()

fprintf :-

- The fprintf() function is used to write set of character into file.

- It sends formatted output to stream.

Syntax :-

```
int fprintf ( FILE * stream , const char * format
                                        [arguement])
```

eg :-

```
#include <stdio.h>
void main()
{ FILE *fp;
fp= fopen( "file.txt", "w");
fprintf (fp, "Hello How are you");
   fclose(fp);
}
```

~~fputs~~ fputc() :-

- The fputc() is used to write a line to a file.

Syntax :-

```
int fputs ( const char *str, FILE
                              *Stream);
```

eg :-

```
#include < stdio.h>
int main ()
```

```
{ FILE *fp;
fp = fopen("file.txt", "wt");
fputs("This is c pgmg", fp);
fputs("This is easy one", fp);
 fclose(fp);
 return (0);
}
```

fputc() :-

The fputc() is just opposite to the fgetc(). fputc() is used to write a single character at a time to a given file.

Syntax :-

```
    int fputc(int char, FILE *pointer);
```

eg:-

```
  # include <stdio.h>
  void main()
  { FILE *fp;
   fp = fopen("file.txt", "w");
    fputc('z', fp);
   printf("Data successfully written to file");
    fclose(fp);
  }
```

fwrite() :-

The fwrite() is used to write data to a file.

Syntax :-

```
fwrite( const void *ptr, size_tsize,
                    size_t nmemb, FILE *stream)
```

eg:-

```
# include < stdio.h>
 int main ()
 { FILE *fp;
  char str[] = " This is c pgmg";
 fp = fopen(" file.txt", "w");
 fwrite (str, 1, sizeof (str), fp);
   fclose (fp);
   return (0);
 }
```

3. Write a C program to read name and marks of n number of students from user and store them in a file. (AU-2021)

```
# include < stdio.h>
# include < conio.h>
 int main ()
 { char name[50];
    int marks, i, n;
clrscr();
Printf(" Entr number of students");
Scanf("%d", &n);
FILE *fptr;
fptr = fopen("c:\\student.txt", "w");
 if (fptr == NULL)
   { printf("Error!");
     exit(1);
   }
for (i=0; i<n; i++)
{ printf(" For students %d \n Entrname", i+1);
  scanf("%s", name);
  printf(" Entr marks");
  scanf("%d", &marks);
 fprintf(fptr, "\n Name: %s \n Marks = %d \n",
                              name, marks);
}
 fclose(fptr);
  getch();
return 0;
}
```

4. Write c program to read name and marks of n number of students from user and store them in a file. If the file previously exists, add the information of n students.

(AU-2021)

```
#include<stdio.h>
#include<conio.h>
int main()
{ char name[50];
 int marks, i,n;
 Printf(" Enter no of
  students");
 scanf("%d",&n);
 FILE *fptr;
 fptr= fopen("c:\\
    student.txt","a");
 if(fptr==NULL)
 {printf("Error!");
  exit(1);
 }
 for(i=0;i<n;i++)
 { printf(" For students
  %d\n Enter name" i+1);
```

```
scanf("%s", name);
printf(" Enter marks");
scanf("%d",&marks);
fprintf(fptr,"\n name:%s
 \n marks=%d",name,marks);
}
fclose(fptr);
 getch();
 returno;
}
```

5. Write c program to write all the member of an array of structure to a file using fwrite(). Read array from the file and display on the screen. (AU-2021)

```
#include< stdio.h>
#include< conio.h>
struct s
{ char name[20];
  int height;
};
int main()
{
  struct s a[5], b[5];
   FILE *fptr;
   int i;
 clrscr();
fptr = fopen("file.txt","wb");
for(i=0; i<5 ; i++)
{
   fflush(stdin);
printf(" Enter name");
gets(a[i].name);
Printf(" Enter height");
scanf("%d", & a[i].height);
}
fwrite(a, sizeof(a), 1, fptr);
fclose(fptr);
fptr = fopen("file.txt","rb");
fread(b, sizeof(b), 1, fptr);
for(i=0; i<5; i++)
{
printf(" Name : %s \n
Height: %d ", b[i].name,
                  b[i].height);
}
fclose(fptr);
getch();
}
```

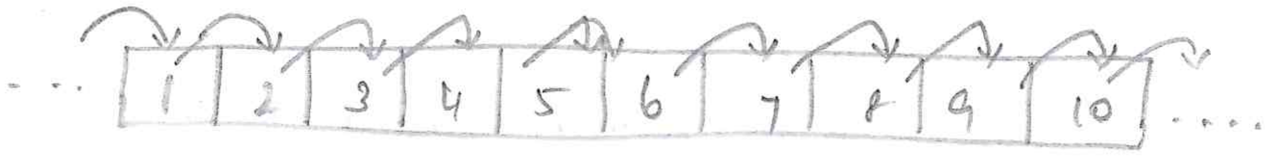
```
printf("\n Name of my program %s", argv[0]);
 if (age==2)
  {
   printf("\n Value given by user:%s", argv[1]);
  }
 else if (argc>2)
  {
  printf("\n Many values given by user");
  }
  else
  {
   printf("\n single value expected\n");
   }
 }
```

7. Explain the various file accessing policies available in language c with example program (AU 2021)

– In computer programming, the two main types of file handling are,

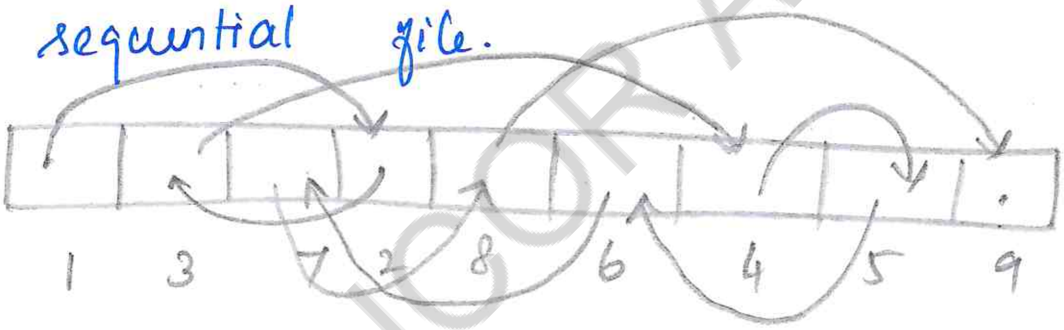
sequential access :-

– In this type of files data is kept in sequential order if we want to read the last record of the file. we need to read all records before that record so it takes more time.



Random access :-

- In this type of file data can be read and modified randomly. If we want to read the last record we can read it directly. It takes less time when compared to sequential file.



- There is no need to read each record sequentially, If we want to access a particular record.

- C support these function for random access file processing.

* fseek() * ftell() * rewind()

fseek() :-

- It is used to move the reading control to different positions using fseek()

function.

Syntax:-

fseek (file pointer, displacement, pointer position);

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

displacement - It is positive or negative file 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	Begining of file
1	Current position
2	End of file.

Example:-

1) fseek (P, 10L, 0)

0 means pointer is on begining

of the file. from this statement pointer position is skipped 10 bytes from the begining 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.

ftell():- It is tell the byte location of current position of cursor in file pointer.

Rewind():- It moves the control to the begining of the file.

Write a program to read last 'n' charalters of the file using file functions (fseek() and fgetc().

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

```
void main()
{ FILE *fp;
 char ch;
 clrscr();
fp= fopen("file.c","r");
 if( fp== NULL)
 {pf( file canot be opened");
 }
else
printf("Entr value of n to read last n character");
scanf("%d", &n);
 fseek( fp, -n, 2);
 while(( ch= fgetc(fp))!= EOF)
   {
    printf("%c", ch);
   }
  }
 fclose(fp);
}
```

Program to find average of numbers stored in sequential file:-

```
# include<stdio.h>
void main()
 { FILE *fp;
```

```
      int n=0;
    float  num, sum, avg;
      sum=0.0;
 fp= fopen("data.txt", "r");
 while(!feof(fp))
 {
   fscanf(fp,"%f",&num);
     sum= sum+num;
        n=n+1;
   }
  avg= sum/n;
    fclose(fp);
  printf("Avg= %f", avg);
 }
```

O/P   input : data.txt

```
      1   2   3   4   5
   avg = 3.000000
```

Reg. No. : E N G G T R E E . C O M

Question Paper Code : 70069

B.E./B.Tech. DEGREE EXAMINATIONS, NOVEMBER/DECEMBER 2022.

Second Semester

Computer Science and Engineering

CS 3251 — PROGRAMMING IN C

(Common to Computer and Communication Engineering/Information Technology)

(Regulations 2021)

Time : Three hours

Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What are the various types of operators?
2. What is the use of preprocessor directive?
3. Declare a float array of size 5 and assign 5 values to it.
4. Sort the following elements using selection sort method 23,55,16,78,2.
5. How is pointer arithmetic done?
6. Which is better to use? Macro or function. Justify your answer.
7. What is meant by structure definition?
8. Define self-referential data structure.
9. Why are files needed?
10. Give an example for fseek().

PART B — (5 × 16 = 80 marks)

11. (a) Explain the storage classes in 'C' with suitable examples.

Or

(b) Explain the looping statement in C with suitable examples.

12. (a) (i) Write a C program to multiply two matrices (2D array) by getting input from the user. (8)

(ii) Write a C program to find scaling of two matrices (2D array) which will be entered by a user. (8)

Or

(b) (i) Write a C program to find determinant of a matrix (2D array) which will be entered by a user. (8)

(ii) Write a C program for matrix transpose. (8)

13. (a) (i) Classify the function prototypes with suitable examples. (8)

(ii) Write a C program to design the scientific calculator using built-in functions. (8)

Or

(b) (i) 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. (8)

(ii) Explain about pointers and write the use of pointers in arrays with suitable example. (8)

14. (a) 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.

Or

(b) Write a C program to create mark sheet for students using self-referential structure.

15. (a) (i) Write a C program to get name and marks of 'n' number of students from user and store them in a file. (8)

(ii) Write a C program to read name and marks of 'n' number of students from user and store them in a file. If the file previously exits then append the information into the existing file. (8)

Or

(b) (i) Write a C program to write all the members of an array of structures to a file using fwrite(). Read the array from the file and display on the screen. (8)

(ii) Describe command line arguments with example C program. (8)

Reg. No. : | | | | | | | | | | | | |

Question Paper Code : 60025

B.E./B.Tech. DEGREE EXAMINATIONS, APRIL/MAY 2022. CSE1

Second Semester

Computer Science and Engineering

CS 3251 — PROGRAMMING IN C

(Common to : Computer and Communication Engineering/Information Technology)

(Regulations 2021)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Write short notes on Keywords in C language.
2. What is difference between the statements a = 5 and a == 5 in language C?
3. Write down the syntax for array declaration.
4. What is the purpose and prototype of the function 'strcpy'?
5. Define the term recursion in language C.
6. What is the relation between the operators '&' and ' * ' in C pointers?
7. In language C can we allocate memory dynamically? How?
8. What are the key differences between structure and union?
9. What will be the impact if 'fclose( )' function is avoided in a file handling C program?
10. What are command line arguments?

PART B — (5 × 16 = 80 marks)

11. (a) Draw the structure of a C program and explain each part in detail.

Or

(b) Enumerate the difference between 'else-if ladder' and 'switch – case' statements with appropriate C programs.

12. (a) (i) What is an array? Explain about various types of arrays in detail. (8)

(ii) Explain the usage of 'strcat( )' with an C program. (8)

Or

(b) (i) Differentiate binary search from linear search. (8)

(ii) Write a C program to compare two strings without using the function 'strcmp( )'. (8)

13. (a) What is modular Programming? How does the Language C support modular programming? Explain in detail.

Or

(b) What is the necessity of parameter passing in C Programs? What are the two types of doing that? Explain any one in detail.

14. (a) (i) What is the purpose of the concept 'structure' in Language C? Explain in detail with an example program. (8)

(ii) Why is singly linked list called as self-referential structure? Explain. (8)

Or

(b) (i) Write short notes on 'Array of structures'. (8)

(ii) Write a comparative analysis on various storage classes of language C. (8)

15. (a) What is file? What are facilities available in language C to handle files? Explain.

Or

(b) Explain the various file accessing policies available in language C with appropriate programs.