C PROGRAMMING

Denick Retchie 1972
Bell laboratries

- The C language is derived from B language which was in turn derived from BCPL(Basic Combined Programming Language).
- C is a middle level language and generally also considered as high level language.
- C is highly portable.
- C is format free language.
- Programs written in C are fast and efficient.
- C is a structured programming language.
- C is extensible as it can have user defined functions.
- C is case sensitive. The keywords in the compiler are defined in lower case.
- The standards for C language is given by ANSI(American National Standard Institute)
- The popular C compilers used are Turbo and Borland.
- Every C program should definitely have one function called main function.
- The character set in C is made up of letters, digits, special characters and white spaces.
- The total number of printable characters are 94.
- C tokens are made up of keywords, identifiers, constants, operators and separators.
- There are 32 keywords in C.
- Keywords have special meaning to the compiler.
- The various keywords in C are

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

- Identifiers are the names of the variables, functions and arrays.
- The rules for naming the identifiers is
  - i. it should not be a keyword
  - ii. it should not have white spaces
  - iii. it should not have any special characters except underscore
  - iv. it should not start with a digit
  - v. its length should not exceed 31 characters.
- Some invalid variable names are 4xyz, abc def, cdf*xz.
- Constants have fixed values and do not change during the execution of the program
- Numeric constants are made up of integer constants and real constants.
- Integers can be represented in decimal, octal and hexadecimal.
- The octal integer starts with 0.
- The value of octal 023 is 19 in decimal.
- The hexadecimal integer starts with 0x.
- The value of hexadecimal 0x15 is 21.
- Real numbers are also called floating point numbers.
- A real number may also be represented in exponential or scientific notation.
- Single character constants are represented within single quotations.
  Eg. 'a', '4'
- A string constant is a sequence of characters enclosed within double quotes.
  Eg. "hello", "Rama"

Interpreter- It is a mediator which con[illegible] dis- More time, memory spa..
Adv- 100% original data will appear,

- C supports some special character constants which are supported in output functions like printf. They are called escape sequences.

Constant	meaning
'\a'	audible bell
'\n'	new line
'\b'	back space
'\r'	carriage return
'\t'	horizontal tab
'\0'	null character
'\\'	back slash

we decide datatype based on ↓ Size, Qualifier ← short, long, signed (+ve), Unsigned (+, Range

- The four fundamental data types in C are int, char, float, double.
- The memory required for each data type is given by format specifier

1 byte=8bits char - 8bits 1 byte %c
1 bit=0/1 int - (16bits) 2 bytes/4 %d
float - (32bits) 4 bytes %f
double - 64bits 8 bytes %lf

Range: signed char -125 to +125, unsigned char 0 to 255, signed integer -32,765 - +32765, unsigned integer 0 to 65,535

- A number can be **signed** or **unsigned**. In an unsigned number all the bits are used for magnitude where as in signed number 1 bit is used for sign.
- An unsigned number is always positive.
- The qualifiers like **short, long** are used to specify the special sizes.
- The sizes of some data types with qualifiers is given below

short int -	1 byte
long int -	4 bytes
long double-	10bytes

- float is single precision and has 6 digits after decimal point .
- double is double precision and has 14 digits after decimal point.
- User defined data types can be defined by using the keyword **typedef.** and enum
- They are used to improve the meaning of the programs.
- **enum** is another user defined data type.
- Eg. enum color { red, blue, green};
- If a declaraction is given in this way red is assigned 0,blue is assigned 1, green is assigned 2.
- The syntax for declaring a variable is

  Storageclass datatype variablename;
- There are 4 storage classes in C.
- They are

Automatically allocate memory, optional

auto	- local variables with scope and life within a function
static	- local variables with scope within the function and life through out the program → updated value will be taken
register	- local variable stored in register
extern	- global variable, after header file

- The scope of the variable is the area of program within which it is accessible.
- The life of the variable is the time till the variable dies.
- If a variable is declared as volatile then that variable's value may be changed at any time by some external sources.
- The largest value a variable can hold depends on the machine.
- Symbolic constants are represented in capital letters.

~~Right~~ Left to ~~Left~~ Right approach

- They are defined by using #define.
- # must be the first character in the line.
- There should not be space between # and define.
- The lines starting with # are called preprocessor directives.
- The preprocessor directives do not end with semi colon.
- C operators are classified into the following
  1. arithmetic operators
  2. relational operators
  3. logical operators
  4. assignment operator
  5. increment and decrement operators
  6. conditional operator
  7. bit-wise operators
  8. special operators
- The arithmetic operators defined in C are

+	addition	*	multiplication
-	subtraction	/	division
%	modulo		

- Op1% op2 gives the remainder of op1 divided by op2. The sign of it will be the sign of op1.
  Eg. 3%2 = 1 3%-2 =1 -3%2 =-1 -3%-2 = -1
- If both the operands are integers then it is called integer arithmetic.
- If both the operands are real then it is real arithmetic.
- If one operand is integer and other operand is real then it is called mixed arithmetic.
- Relational operators are used to compare two quantities.
- The various relational operators are

<	less than	>	greater than
<=	less than equal to	>=	greater than equal to
==	equal to	!=	not equal to

If true prints 1
False prints 0

- The logical operators are

&&	logical and
||	logical or
!	logical not

- Assignment operator is =. The left hand side of assignment operator is *lvalue*. The right hand side of assignment operator is called *rvalue*.
- The short hand assignment operators are
  += -= *= /= %=
  Eg. a+=4 is a=a+4 d*= c +f is d=d*c +f
- The increment and decrement operators are ++ and --.
- These are unary operators.
- An unary operator is an operator with only one operand.
- A binary operator is an operator with two operands.
- A ternary operator is an operator with three operands.
- If ++ is used before the variable then it is called pre-incrementation and if it is used after the variable then it is called post-incrementation.
- Pre-incrementation and post-incrementation are the same if they are used as single statements. They differ only when used in expressions.

- The ternary operator is ? : operator. It is also called conditional operator.
- Syntax exp1 ? exp2 : exp3
- If exp1 is true then exp2 is evaluated if not exp3 is evaluated. smallest element in memory
- Bit-wise operators are used for manipulation at bit level. change bit to bit in a memory bit position
- & bit-wise and -+=T ++=F bit-wise exclusive or

  | bit-wise or << $n\times2^1$ ← left shift $\frac{n}{2^2}$ → shifting position

  ~ n=5 =-5-6 one's complement >> right shift
- A comma linked list of expressions are evaluated from left to right and the value of the right most expression is the value of the combined expression.

  Value =(x=5,y=15,y-x)

  The value will be assigned the value 10.
- The sizeof operator returns the number of bytes the operand occupies.

  Eg. sizeof(int) results a value 2.
- An arithmetic expression will be evaluated from left to right using rules of precedence.
- C has the facility of automatic type conversion.
- Explicitly converting the data type is called type casting.
- The input and output statements are scanf and printf functions.
- The format specifiers used are

%d	integer
%f	float
%lf	double
%c	character
%s	string

- The **if** statement is used as a conditional statement.
- The syntax of using if statement is

```
if(expr)
{    ......
  }
```

- The { and } is used to group some set of statements. This is called a compound block.
- If the expr evaluates to non-zero value then it is true otherwise it is false.
- The **switch** statement is used in the place of if-else ladder when there are more number of conditions to be tested.
- The various **case** statements are used within the switch statement.
- Each case statement should end with break statement.
- The **goto** statement is an unconditional loop.
- If a loop does not terminate it is called infinite loop.
- There are 3 loop statements

  1. while 2. do … while3. for
- The while loop is entry controlled loop and do.. while loop is exit controlled loop.
- Syntax of while loop:
- while(condition)

```
{
      set of statements
 }
```

- Syntax of do.. while is
- do

- {
  set of statements
  } while(cond);
- Every statement in C is terminated by semicolon which is called statement terminator.
- The syntax of for loop is
- for(initialization; test condition ; increment)
  {
  set of statements
  }
- The initialization is done first, then the condition is tested and after executing the set of statements updation is done.
- **Arrays :** A list of items can be given one variable name using one subscript . Such a variable is called one-dimensional array.
- If an array is not initialized it will have garbage values.
- An array is initialized using { and }.
- An integer array can store only integer type of data.
- Memory is allocated at the compilation time. It is static memory allocation.
  Eg. int a[10];
- An array **a** will be declared for 10 integers.
- A memory of 20 bytes will be allocated for it.
- The array declaration int c[ ]; is invalid because the size of the array is not given.
- The array initialization can be given in this way
  Char z[3]={'a','c','d'};
- If some are initialized and others are not initialized they are set to zero.
- An array is given indices to refer various items in the array. The array index start with 0.
- The definition of arrays reduces the number of variables of the same type to be defined.
- A two dimensional array can be taken as an array of single dimensional arrays.
- A two dimensional array is declared as follows
  type arrayname[row_size][col_size];
- A two dimensional is initialized similar to a single dimensional array.
- C allows multi-dimensional arrays.
- There is no limit on the number of dimensions.
- It becomes difficult for the programmer to visualize arrays of more than 4 dimensions.

Strings:

- A string is an array of characters.
- A string is terminated by a special character called null character(\0).
- A string is given within double quotations.
- The size of the string should be equal to the maximum number of characters in the string plus one.
- C permits to initialize a character array without specifying the number of elements.
- In such cases the size will be determined automatically.
  Eg: char a[ ] = "hello";
- Some of the important string functions defined in string.h header file are
  strcat strcpy strcmp strlen
- The function strcat() takes two strings s1 and s2 as arguments and it appends str2 at the end of str1.

- The function strcpy() takes two strings s1 and s2 as arguments and copies the second string s2 into first string s1.
- The function strcmp() takes two strings s1 and s2 as arguments and compares them as which one precedes the other. It returns an integer. It returns 0 if both the strings are equal. It return a negative value if str1 comes first in the dictionary. It returns a positive value if str2 comes first in the dictionary.
- The function strlen() takes a string as argument and returns the length of the string excluding \0.

Functions

- C functions can be classified into two categories
  - i. library functions
  - ii. user-defined functions
- Library functions are not required to be written by programmers. They are pre-defined and placed into number of header files.
- To use the library functions we have to include the corresponding header file in which the function is declared.
- Examples of header files: stdio.h dos.h conio.h stdlib.h math.h string.h
- Examples of library functions : printf() scanf() getchar() cos()
- **User-defined Functions:**
- C functions are easy to define and use.
- main() is a specially recognized user-defined function in C.
- The use of having user defined functions is that it makes coding, debugging and testing easier. A function when defined can be re-used number of times.
- Using functions facilitates top-down modular programming.
- A function is a self contained block of code that performs a particular task.
- The inner details of the function are not visible to the rest of the program.
- Any function can call any other function.
- There are two methods of declaring the function arguments.
- In the classic method or the older method the function arguments are placed separately after the definition of the function name.
- In the newer method or ANSI method the function arguments are declared along with the function name.
- Modern compilers support both the versions of function declaration.
- The function declaration or function prototype tells the compiler the name of the function , the return type , the number and type of arguments.
- The function definition has function declarator followed by function body.
- Syntax of function declarator
- `returntype  functionname( argument-list)`
- If this is followed by semi-colon it is called function declaration.
- `returntype functionname(argument-list);`
- Function declaration is used before the function definition to tell the compiler about the function so that it can be used before definition.
- If the function declarator is followed by function body it is called function definition.

```
    returntype  functionname(argument-list)
    {
            declaration of local variables
```

```
        executable statements
        return statement
   }
```

- The local variables are defined only if necessary.
- The functionname must follow the same rules of C variable names.
- The argument-list contains valid variable names separated by commas. The list is surrounded by parentheses.
- A function may or may not return any value.
- A function should return one and only one value.
- There can be more than one return statement in a function.
- There cannot be two return statements successively.
- A function can be called by using the name of the function.
- There are 3 categories of functions depending on whether arguments are present or not and whether the value is returned or not.
  1. Functions with no arguments and no return value
  2. Functions with arguments and no return value
  3. Functions with arguments and return value
- When a function has no arguments , it does not receive any data from the calling function.
- When a function does not return any value , there is no data transfer between calling function land the called function.
- If there nothing to be returned, return statement is optional.
- The arguments used in the function definition are called formal arguments or dummy arguments.
- The arguments used in the calling function are called actual arguments.
- The actual and formal arguments should match in number, type and order.
- The default return type is integer.
- C permits nesting of functions.
- A function can call itself. This is called recursion.
- A stack is used internally for recursion.
- Recursive functions can be effectively used to solve problems where the problem is divided into small problems of same sort.
- Recursive functions take more time when compared to iterative functions.
- Arrays can also be passed as arguments to functions.
- To pass an array to a function, it is sufficient to list the name of the array without subscripts.
- When an entire array is passed as an argument , the contents of the array are not copied into formal array, but the information about the addresses of array elements are passed.
- The names of the arguments in the prototype declaration need not be the same names given in prototype definition.
- There can be functions with variable number of arguments. It is indicated by ellipsis(...).
- Examples for the functions with variable number of arguments are printf(), scanf().
- A function that returns a value can be used in arithmetic expressions.
- A function cannot be present on the left hand side of the assignment statement.
- A function can be placed either before or after the main function.
- When more functions are used they can appear in any order.
- The variables may be broadly classified depending on the place of their declaration, as internal(local) and external(global) variables.

- Internal variables are those declared within a function.
- External variables are declared outside of any function.
- There are 4 storage classes in C . They are
  - i. Automatic variables
  - ii. External variables
  - iii. Static variables
  - iv. Register variables.
- Automatic variables created in the function when it is called and they are destroyed automatically when the function is exited.
- If the storage class is not given explicitly then by default it is taken as automatic variable.
- Variables that are both alive and active throughout the entire program are known as external variables. They are known as global variables.
- Global variables can be accessed by any function in the program.
- Global variables are initialized to zero by default.
- The variable declared as extern can be accessed from any function.
- The extern declaration does not allocate storage space for variables.
- Multiple files can share a variable if it is defined as external variable.
- If we declare a variable as global in two different files used by a single program, then the linker will have a conflict as to which variable to use.
- The scope of static variables is within the function .
- The life of the static variables is through out the program.
- A static variable is initialized only once when the program is compiled. It is never initialized again.
- A static external variable is available only within the file where it is defined while the simple external variable can be accessed by other files.
- The variables defined as **register** are stored in the register so that it is accessed very fast. Only very frequently accessed variables are defined as register.
- As the number of registers in the microprocessor are limited the number of register variables should be very less.

Structures and Unions:

- Structures and unions are composite data types.
- A structure is used to pack items of different data types.
- The concept of structures is similar to that of a record.
- The structure declaration tells the format of the various items in it.
- When a structure is declared memory is not allocated. Memory is allocated only When structure variables are created.
- Each member within a structure can be of different type.
- The total memory allocated for a structure variable is the sum of the various data Items within the structure.
- The items of the structure variable can be accessed by using '.' (dot)operator.
- Initialization of individual structure members within structure definition is not valid.
- An array of structure variables is possible.
- An array can be defined within a structure.
- A structure can be defined within a structure.
- A structure can be passed to a function as argument.
- The syntax of union is similar to that of a structure.

- In structures, each member has its own storage location, whereas all the members of a union use the same location.
- A bit field is a set of adjacent bits whose size can be from 1 to 16 bits in length.
- The internal representation of bit fields is machine dependent.
- Bit fields cannot be arrayed.

Pointers:

- The main necessity of pointer variables is dynamic memory allocation.
- A pointer enables us to access a variable that is defined outside the function.
- Pointers reduce the length and complexity of a program.
- They increase the execution speed.
- A pointer is nothing but a variable that contains an address which is a location of another variable in memory.
- The & operator is called address of operator.
- A pointer variable is declared in this way by using special character "*".
- Eg. int *ptr;
- An integer pointer points only integer type of data.
- A character pointer points only character type of data.
- A float pointer points only float type of data.
- A double pointer points only double type of data.
- Pointer variables can be used in expressions.
- Pointer arithmetic takes place using scale factor.
- The scale factor in character, integer, float and double pointers is respectively 1,2, 4,8.
- If p1 is a pointer variable pointing to integers then p1++ will increment p1 by 2.
- Pointer multiplication and division are invalid.
- The process of calling a function using pointers to pass the addresses of variable is known as call by reference.
- There can be pointers to functions.
- A pointer can be defined to a structure variable.
- To access the data item within a structure using a pointer the -> operator is used.
- A pointer contains garbage until it is initialized.
- When an array is passed as an argument to a function, a pointer is actually passed.

File Management

- Files are used because it is very cumbersome and time consuming to handle large volumes of data through terminals.
- The basic operations to be performed on files are

  i naming a file ii. opening a file

  iii reading the file iv. writing into the file v. closing the file
- All files should be declared of the type FILE before they are used. Data structure of a file is defined as FILE in the header file.
- The various functions used for file operations are

fopen()	creates a new file or opens existing file
fclose()	closes a file which has been opened for use
getc()	reads a character from a file
putc()	writes a character to a file
fprintf()	writes s set of data values to a file
fscanf()	reads a set of data values from a file

getw()	reads an integer to a file
fseek()	sets the position to the desired position in the file
ftell()	gives the current position in the file
rewind()	sets the position to the beginning of the file

- The various modes of opening a file are

R	open the file for reading only
W	open the file for writing only
A	open the file for appending data
R+	the existing file is opened to the beginning for both reading and writing
W+	same as w except for reading and writing
A+	same as a except for reading and writing

- A file must be closed as soon as all operations on it have completed.
- EOF is End-Of-File marker. It is a symbolic constant defined as –1 in library files.
- The feof() function is used to test for an end of file condition.
- Even main() function has arguments. They are passed as arguments to the command line. Therefore they are called command line arguments.
- The two arguments of main function are **argc** and **argv .**
- The variable argc is argument counter that counts the number of arguments on the command line.
- The variable argv is an argument vector that represents an array of character pointers that point to the command line arguments.
- The argument vector contains the entire command line in memory.
- The process of allocating memory at runtime is called dynamic memory allocation.
- Dynamic memory allocation techniques permit us to allocate additional memory or to release unwanted space in memory.
- The 4 functions that support dynamic memory allocation are

malloc()	allocates requested size of bytes and returns a pointer to the first byte of the allocated space.
calloc()	allocates space for an array of elements initializes them to zero and then returns a pointer to memory
free()	frees previously allocated space
realloc()	modifies the size of previously allocated space

- When memory allocation fails then NULL is returned.
- The malloc() function returns a pointer of type void.
- malloc() function allocate a block of contiguous memory.
- calloc() allocates multiple blocks of storage, each of the same size and then sets all bytes to zero.
- Reallocation or modification of already allocated memory is done by realloc() function.
- Preprocessor directives start with a special symbol #.
- The preprocessor directives are used for 3 purposes.
  - i. macro substitution directives
  - ii. file inclusion directives
  - iii. compiler control directives
- Macro substitution is a process where an identifier in a program is replaced by a predefined string composed of one or more tokens.
- This task is performed by using #define.

- Macros can have arguments.
- Macros can be nested.
- Macros are open subroutines whereas ordinary functions are closed subroutines.
- Recursion is also possible in macros.
- An external file containing functions or macro definitions can be included as a part of a program so that we need not rewrite those functions or macro definitions.
- Debugging and testing are done to detect errors in the program. The compiler can detect syntactic and semantic errors where as it cannot detect errors in the algorithm.

DATA STRUCTURES

- A program is generally given by a combination of data structures and algorithms.
  Program = data structures + algorithms
- Data is a collection of raw facts.
- It can be of different types namely numeric ,text, Boolean, date etc.
- The type of data is given by the data type.
- The structures which logically relate various data items in some manner are called data structures.
- The representation of a particular data structure in the memory of a computer is called storage structure.
- The representation of storage structure in auxiliary memory is called file structure.
- An algorithm is a sequence of steps to be followed to complete a particular task.
- Time and space complexities are determined to compare algorithms.
- Time complexity is the amount of time required to execute the algorithm.
- Space complexity is the amount of space required to execute the algorithm.
- Both time complexity and space complexity are given as a function of the input size 'n'.
- There are 3 ways of representing time complexity
  i. Big O notation ii. Omega notation iii. Theta notation.
- Big O notation represents the upper bounds.
- Omega notation represents the lower bounds.
- Theta notation represents tight bounds.
- Non-primitive data structures can be classified as arrays, lists and files.

Arrays

- An array is an ordered set which consists of a fixed number of objects.
- No deletion and insertion operations are performed on arrays.
- A number of locations will be sequentially allocated for an array.
- An array is also called a vector.
- To access any element within an array the address of that element is computed by using the starting address of the array
- Address of A[I] = address of A + c* (I)
  where **c** is the size of each object of the array. The index of the array is taken to start with 0.
- Memory allocation of arrays is done at compile time.

- Two dimensional arrays are stored row by row or column by column. If they are stored row by row it is called row-major order and if they are stored column by column it is called column-major order.

Stacks

- It is a linear data structure of variable size.
- Stack permits insertion and deletion of an element at only one end.
- The insertion operation is referred to as push operation.
- The deletion operation is referred to as pop operation.
- The most accessible element of the stack is called the top of the stack.
- The elements are popped in the opposite order of pushing.
- This phenomenon is called LIFO(Last In First Out). This is used in recursive functions.
- A pointer TOP keeps track of the top element in the stack. Initially when the stack is empty TOP value is zero and if the stack contains only one element TOP value is one.
- Each time when a new element is pushed the TOP value is incremented by one.
- Each time when a element is popped the TOP value is decremented by one.
- When the stack is full, it is referred as 'stack overflow' and no element can be pushed then.
- When the stack is empty it is referred as 'stack underflow' and no element can be popped then.
- The important applications of stacks are recursion , compilation of infix expressions and stack machines.
- There are two types of recursive functions : primitive recursive functions and non-primitive recursive functions.
- All primitive recursive functions can also be represented by iterative procedure.
- The depth of recursion is the number of times the procedure is called recursively.
- The notation in which the operator is written between the operands in an expression is called infix notation.
- Repeated scanning is avoided if the infix expression is first converted to equivalent suffix or prefix expression.
- The suffix notation is also called postfix or reverse polish notation.
- The prefix notation is also called prefix or polish notation.

Operand1 operator operand2	---	Infix
Operand1 operand2 operator	---	Postfix(reverse polish)
Operator operand1 operand2	---	Prefix(polish)

- Stack is used in the process of conversion of infix expression into polish notation.

Queues

- It is a list which permits deletions at one end and insertions at the other end.
- The information is processed in the same order as it is received . FIFO(First In First Out) or FCFS(First Come First Served).
- A queue which operates based on priorities is called priority queue.
- The two pointers Front and Rear keeps track of the queue.
- The Front pointer points to the starting of the queue and it is where deletions are performed.
- The Rear pointer points to the end of the queue and it is where insertions are performed.
- Memory allocated for the queue is optimally utilized if it is represented as a circular queue.
- A deque (doubly-ended queue) is a linear list in which insertions and deletions are made at both the ends.

- The input-restricted deque allows insertions at only one end, while an output-restricted deque permits deletions from only one end.

Linked lists

- A list is an ordered set consisting of a variable number of elements to which insertions and deletions can be made.
- A list which displays the relationship of adjacency between elements is said to be linear.
- The insertion and deletion of elements in a list is specified by the position.
- In an array the time taken to access any item is the same whereas it differs from one time to another in a linked list.
- A linked list is made up of number of nodes of the same type.
- Each node consists of various data items and a pointer as links.
- The pointers require more amount of memory.
- Linked lists can be used to represent the polynomials.
- If each node has only one pointer then the list formed is linear linked list. The link stores the address of the next node.
- Insertions and deletions of nodes in a linked list is very easy known the position of the node where it is to be inserted or from where it is to be deleted.
- The last node has NULL in its link.
- A linear linked list can be traversed in only one direction.
- If the last node points to the first node then it is called circular linked list.
- The problem with circular linked list is that it is difficult to keep track when the traversal end. So a special node without data is placed at the starting of the list and it is called head list.
- If two pointers are placed in each node , one to store the address of successor node and other to store predecessor node then it is called a doubly linked list.
- The burden of handling the pointers is more for doubly linked lists.
- In doubly linked list the traversals can takes place in both the directions.

Trees

- Tree is a non-linear data structure.
- Any general tree can be converted into a binary tree.
- Any node in a binary tree has either 0,1 or 2 successors.
- The nodes with no successors are called terminal nodes or leaves.
- Two trees are said to be similar if they have the same structure.
- If a node has two successors, then the node is called parent and the other two nodes are called left child and right child respectively.
- The depth or height of the tree is the maximum number of nodes in any branch of the tree.
- A tree is said to be complete if all the levels except the last level are full of nodes.
- The height of a complete binary tree is log n + 1.
- A binary tree can be traversed in 3 ways.

  i. pre-order ii. in-order iii. post-order
- In pre-order traversal
  - a. process the root
  - b. traverse left sub-tree in preorder
  - c. traverse right sub-tree in preorder
- In in-order traversal
  - a. traverse the left sub-tree in in-order

b. process the root

c. traverse the right sub-tree in in-order

- In post-order traversal

a. traverse the left sub-tree in post-order

b. traverse the right sub-tree in post-order

c. process the root

- The average running time for searching an element in a binary search tree is O(log n).
- While searching an item in the binary search tree a node is compared with one node and if the value is less than the node we move to left sub-tree otherwise we move to right sub-tree.

Graphs

- A graph is made up to set of vertices and set of edges connecting them.
- The nodes are said to be adjacent or neighbors if there is an edge connecting the two vertices.
- The degree of a node u, deg(u) is the number of edges involving the node u.
- The node with degree zero is called an isolated node.
- Path from a node u to node v is given by a sequence of nodes.

  P= (v1, v2, ... vn)
- If v1=vn then it is a closed path.
- A path is said to be simple if it does not have any two nodes with repetition.
- A simple closed graph is nothing but a cycle.
- A graph is said to be connected if there is a path between any two of its nodes.
- A tree is an acyclic graph which is connected.
- A graph is said to be complete if every node is adjacent to every other node.
- A complete graph with n vertices has n(n-1)/2 edges.
- A graph is said to be labeled or weighted if a label or weight is given to each and every edge.
- A multigraph has multiple edges and loops.
- A directed graph is also called digraph and contains even direction in the edges.
- The indegree of a node is the number of edges coming into that node.
- The outdegree of a node is the number of edges going out of the node.
- Graphs are represented by using Adjacency matrix.
- A node v is reachable from node u if there is a path from u to v.
- Path matrix or Reachability matrix shows whether a particular node is reachable from vertex to another or not.
- A graph is traversed by using breadth first search and depth-first search.

Sorting and Searching

- The worst case time complexity and average time complexity of bubble sort is $O(n^2)$.
- The worst case time complexity of quick sort is $O(n^2)$ and average time complexity is $O(n \log_2 n)$
- The worst case and average case time complexity of heap sort is $O(n \log_2 n)$.
- The worst case and average case time complexity of insertion sort is $O(n^2)$.
- The sort which is highly affected by the order in which the initial set of numbers are , is insertion sort. If they are in the same order sorting is very faster and if they are in reverse order then takes more time.
- The worst case and average case time complexity of selection sort is $O(n^2)$.
- In merge sort two lists in ascending order are merged together.
- The worst case and average case time complexity of merge sort is $O(n \log_2 n)$.

- Radix sort is popularly used for sorting a list of large number of alphabetical words.
- Hashing is method in which a key when operated on a function results in the address of that record.
- The most commonly used hashing methods are
  - i. division method
  - ii. mid-square method
  - iii. folding method
- There is a possibility that two or more keys result in a same function value, it is called collision.
- Collision resolution can be done by either linear probing or quadratic probing or double hashing.

MULTIPLE CHOICE QUESTIONS

1. What is the final value of the digit ?

```
main()
{
  int digit;
  for(digit=0;digit<=9;++digit)
      printf("%d\n",digit);
  digit = 2 * digit;
   --digit;
}
```

a. 19 b. –1 c. 17 d. 16

2. What is the output of the following program

```
#include <stdio.h>
#define square(x) x*x
main()
{
         int b,c=3;
          b=square(c)++;
          printf("%d",b);
}
```

a. 3 b. 4 c. 9 d. 16

3. If x and y are variables declared as double x=0.005 , y = -0.01;. What is the value of **ceil(x+y)** where **ceil** is a function to compute the ceiling of the number

a. 1 b. 0 c. 0.005 d. 0.5

4. What will be the output of the following program

```
main()
{
    int i= 255, j=0177,k=0xa0;
    printf("%x %o %d",i,j,k);
}
```

a. 255 0177 a0 b. ff 177 160 c. 255 0177 0xa0 d. error

5. In the case of ordinary **int** variables
   a. the left most bit is reserved for sign b. the right most bit is reserved for sign
   c. no bit is reserved for sign d. none

6. #define microprocessor command is used for defining
   a. macros b. for loop c. symbolic constants d. both a and c

$\frac{n}{2^k}$

7. If i=20 and j = i << 1 then
   a. i = j [✓ b. j =2i] c. i= 2j d. j = i −1

8. The minimum number of temporary variables needed to swap the contents of two variables is
   a. 1 b. 2 c. 3 d.0

16 8 4 2 1
1 0 1 0 0

9. Coercion
   a. takes place across assignment operator
   b. takes place if an operator has operands of different data types
   c. means casting d. both a and b

10. What is the output of the following

```
main()
{
   int x,y=2,z,a;
   x=(y*=2) + (z=a=y);
   printf("%d",x);
}
```

   a. 7 b. 6 c. 8 d. error

11. Bit fields will be accommodated in a word
   a. from left to right [✓ b. from right to left]
   c. in a way that depends on the implementation d. none of the above

12. The && and | | operators are used to
   a. combine two numeric values b. combine two Boolean values
   c. combine two float values d. combine two double values

13. A C function can have variable parameters
   a. true b. false c. can't say d. none

14. Can a void pointer be used in manipulations
   a. yes b. no c. can't say d. none

15. Output of the following

```
struct abc {     };
main()
 {
     printf("%d",sizeof(struct abc));
 }
```

   a. 1 b. 0 c. error d. none

16. switch(a)

```
{
   default:  printf("A");
   case 1 :  printf("B");
}
```

If a=2 then the output is

a. AB b. A c.B d. error

17. Which keyword in C is used to avoid register storage
    a. static b. extern c. volatile d.none

18. struct ab { char a,b; };

```
union abcd
{
    int c;
    struct ab d;
}k;
k.d.a = 5;
k.d.b = 0;
```

then the value of k.c is
    a. 5 b. 7 c. 0 d. 50

19. fp = fopen("abc.dat","r+"); then the contents of "abc.dat"
    a. exists b. can be modified c. can't be appended d. all

20. int a=5;

```
f(){ extern int a;  a=7; }
g(){ extern int a;  f();  printf("%d",a);}
main() {  g();  }
```

a. 5 b. 7 c. error d. none

21. In recursion which variables are not stored on the system stack
    a. static b. local c. global d. both a and c

22. int a[] ={4,2,3};

```
int *p = a;
*p++;
```

then *p is
    a. 4 b. 2 c. error d.none

23. Can a function be passed as a parameter to another function
    a. true b. false c. can't say d. none

24. enum abc { a=-1,b=5,c,d=8}; then the value of c is
    a. 6 b. 2 c. 7 d. garbage

25. char * const p = "Good" then
    a. contents of p can be changed b. "Good" can be changed
    c. both a and b d. none

26. Which of the following is the primitive data type
    a. string b. array c. file d. none

27. What is the output of the following program

```
main()
{
        int sum=0,i;
         for(i=1;i<=10;i++ )
             if(i%2 ==0)
               sum = i + sum;
           printf("%d",sum);
}
```

a. 45 b. 40 c. 0 d. 30

28. If a= -11 and b= -3 . What is the value of a%b

a. –3 b. 3 c. 2 d. –2

29. What is the output of the program

```
main()
{
    int sum=0,i;
    for(i=-100;i< 0;i++)
        sum+=abs(i);
    printf("%d",sum);
}
```

a. 100 b. 5050 c. –5050 d. -100

30. Pick the correct statement

a. unions are like structures

b. unions contains members of different data types which share same storage area in memory

c. unions are less frequently used in program d. all the above

31. Consider the following declaration

enum colors {black,blue,green}; this represents

a. black=0,blue=1,green=2 b. color ='black' or color ='blue' or color='green'

c. color[0]='black' d. none

32. The library function exit() is used for exit from

a. a loop b. a block c. a function d. a program

33. When an array name is passed to a function, the function

a. accesses exactly the same array with the same name as the calling program

b. accesses a copy of the array passed by the program

c. both a and b

d. none

34. What is the output of the program

```
#define row 3
#define col 4
int a[row][col] = { 1,2,3,4,5,6,7,8,9,10,11,12};
main()
{
    int i,j,k=99;
    for(i=0;i<row;i++)
     for(j=0;j<col;j++)
        if(a[i][j] < k)
          k=a[i][j];
    printf("%d",k);
}
```

a. 99 b. 12 c. 1 d. none

35. What is the output of the following program

```
main()
{
  int a,*ptr,b,c;
```

```
    a=25;
    ptr=&a;
    b=a+30;
    c=*ptr;
    printf("%d,%d,%d",a,b,c);
}
```

a. 25,25,25 b. 25,55,25 c. 25,55,55 d. none

36. What is the output of the following program

```
main()
{
    char ch='A';
    while(ch<='F')
    {
        switch(ch)
        {
            case 'A':
            case 'B' :
            case 'C'  :
            case 'D':
            case 'E':      ch++;
            case 'F':      ch++;
        }
        putchar(ch);
    }
}
```

a. ABCDEF b. FG c. EFG d. CEG

37. Pick the correct one

a. while(0) { i=2; } 2 is assigned to i
b. do { i=2; } while(0); 2 is assigned to i
c. both a and b
d. none

38. Pick the correct one

a. calloc(), allocates and clears memory
b. malloc(), allocates memory but does not clear memory
c. both a and b
d. none

39. Pick the correct one

a. a char type has an equivalent integer interpretation so it is really special kind of short integer
b. the printf() is a macro which takes in a variable number of arguments
c. 12,245 is a legal string constant in C
d. none

40. Pick the correct one

a. the statement x=x<<2 is equivalent to the statement x=x*4
b. an if else statement can be replaced by a ternary operator
c. both a and b

d. none

41. What is the output of the following program

```
main()
{
        extern int a;
        printf("%d",a);
  }
  int a =50;
```

a. 50 b. 0 c. garbage value d. error

42. Suppose a program is divided into 3 files f1,f2,f3 and a variable defined in f1 is to be used in files f2 and f3 then the storage class of the variable is

a. auto b. static c. register d. extern

43. Examine the following program

```
main()
{
      display();
}
display()
{
     printf("Titanic is a ship");
}
```

Pick the correct answer

a. display() function returns void type data by default

b. display() function returns int type data by default

c. display() function displays a message "Titanic is a ship" and returns number of bytes displayed

d. both b and c

44. What is the output of the program

```
main()
 {
      int i;
      int a[5]={10,20};
       for(i=0;i<5;i++)
          printf("%d",a[i]);
 }
```

a. 10 20 b. 10 20 0 0 0

c. 10 20 and remaining 3 values are garbage d. error in initialization

45. What would be the output of the program

```
main()
{
        int i=4;
         switch(i)
          {
             default:   printf("A is a upper case letter");
             case 1 :   printf(" b is a lower case letter"); break;
             case 2:    printf("0-9 digits");  break;
```

```
            case 3:     printf("Ascii value of a is 97");
        }

    }
```

a. default keyword should not be on top of all cases

b. A is upper case letter b is a lower case letter

c. Wont print any message as no case is 4

d. None

46. Point out the error , if any in the following program

```
main()
{
            int I;
            for( ; ;  )
            {
                printf("%d",I++)
                if(I>10)
                  break;
            }
    }
```

a. the condition in the for loop is a must

b. the two semicolons should be dropped

c. the for loop should be replaced by a while loop

d. no error in the program

47. Point out the error , if any, in the following program ?

```
main()
{
            int i=4, j=1,k=2;
            switch(i)
            {
               case i:
               case j:
                case k:   printf("All are same"); break;
            }
    }
```

a. error as no statements for first two cases

b. constant expressions required in the second and third case, we can't use j and k

c. no error in the program

d. none of the above

48. Point out the error if any, in the following program

```
main()
{
            int I=1;
            switch(I)
            {
                  printf("\n Hello");
                  case 1:      printf("\n case one"); break;
                  case 2:      printf("\n case two"); break;
                  case  1+3*4: printf("\n case last"); break;

            }
}
```

a. the first printf() can never get executed and all statements in a switch have to belong to same case or the other

b. case label should not be an expression

c. the first printf should not exist, so error in program

d. none of the above

49. What would be the output of the following program

```
main()
{
        int i=-3,j=2,k=0,m;
        m= ++i && ++j || ++k;
        printf("%d %d %d",i,j,k,m);
}
```

a. –4 2 0 1   b. –4 3 0 0   c. –2 3 0 1   d. –2 3 1 1

50. What would be the output of the following program

```
main()
{
        printf("%d %d",sizeof(3.14), sizeof(3.14f));
}
```

a. 4 4   b. garbage value   c. 8 4   d. 4 8

51. By default any real number is treated as

a. float   b. double   c. long double   d. depends upon memory model used

52. What would be the output of the program

```
main()
{
     int  arr[] ={ 12,13,14,15,16};
      printf("%d %d %d",sizeof(arr), sizeof(*arr),sizeof(arr[0]);
}
```

a. 2 2 2   b. 10 2 2   c. 5 2 2   d. none

53. What would be the output of the following program

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

a. 5   b. 6   c. 7   d. 0

54. What would be the output of the following program

```
main()
{
        int near *a;
        int far *b;
        int huge *c;
        printf("%d %d",sizeof(a),sizeof(b),sizeof(c));
}
```

a. 2 2 2   b. 2 2 4   c. 2 4 4   d. 4 4 4

55. According to ANSI specifications which is the correct way of declaring main() when it receives command line arguments

a. main(int argc, char *argv[])
b. main(char *argc, int argv[])
c. main(){int argc; char *argv[];}
d. none

56. What would be the output of the following program assuming that the array begins at location 1002.

```
main()
{
      int a[3][4] = { 1,2,3,4,5,6,7,8,9,10,11,12};
      printf("%u %u %u",a[0]+1,*(a[0]+1),*(*(a+0)+1));

}
```

a. 2 1004 2   b. 1004 2 2   c. 1004 2 0   d. none

57. What would be the output of the following program

```
#include <stdio.h>
main()
{
        printf("%d %d",sizeof(NULL),sizeof(""));

}
```

a. garbage 1   b. 0   c. 1 1   d. 2 1

58. Examine the following program and comment on it

```
main()
{
      float *p1,I=25.5;
      char *p2;
      p1 = &I;
      p2 = &I;
}
```

a. executes successfully   b. simply a warning
c. an error of suspicious pointer conversion   d. none

59. What is the output of the following program

```
main()
{
    char a[]="neelesh";
    char *b="neelesh";
    printf("%d %d",sizeof(a),sizeof(b));
    printf("%d %d", sizeof(*a),sizeof(*b));
}
```

a. 11 2 1 1   b. 10 10 1 1   c. 1 2 10 10   d. none

60. Examine the following program

```
main()
{
          const int x;
          x=128;
```

```
            printf("%d",x);
    }
```

a. 128 b. error at compile time c. error at runtime d. none

61. Pick up the correct one from following

a. bool is a keyword in C b. asm is a keyword in C

c. void is a keyword in C d. none of the above

62. What would be the output of the following program

```
main()
{
        int i=10,j=12,k,l,m;
        k= i | j;
        l= i & j;
        m = i ^ j;
        printf("%d %d %d",k,l,m);
}
```

8 4 2 1

10 -> 1 0 1 0

i^j= 12 -> 1 1 0 0

0 1 1 0 -> 6 -> i^j

1 0 0 0 -> 8 -> i&j

1 1 1 0 -> 14 -> i|j

[✓ a. 14 8 6] b. 6 8 14 c. 8 14 6 d. 6 14 8

63. Which of the following bitwise operator is used for masking a particular bit in a number

a. & b. | c. ! d. ^

64. What is the maximum memory that we can allocate by a single call of malloc()

a. 64KB b. 640KB c. 256 bytes d. infinite size

65. What would be the output of the second printf() in the following program

```
#include <alloc.h>
main()
{
        int *p;
        p= (int *)malloc(20);
        printf("%u",p);  /*suppose this prints 1314*/
        free(p);
        printf("%u",p);
}
```

a. 1314 b. 1316 c. garbage d. error

66. What would be the output of the following program

```
main()
{
        float a[]= { 12.4,2.3,4.5,6.7};
        printf("%d",sizeof(a)/sizeof(a[0]));
}
```

a. 1 b. 0 c. 4 d. 2

67. What would be the output of the following program

```
main()
{
        char str[5] = "hello";
        printf("%s",str);
}
```

a. error b. hello c. can't say exactly d. none

68. Which of the following is true about 'argv'

a. it is an array of character pointers

b. it is a pointer to an array of character pointers

c. you cannot use variable name other than argv

d. none

69. A C program contains the following array declaration char text[80]; and initialized with a string "Getting a seat can be a challenging". The output resulting from the printf("% -18.7s",text);

a. Getting a seat cab be a challenging b. Getting a seat can

c. Getting d. Getting a seat can be a C

70. In the following program, how many functions are used

```
#include <stdio.h>
main()
{
        printf("hello friend");
}
```

a. 0 b. 1 c. 2 d. all

71. What is the output of the following program

```
main()
{
    char far *s1, *s2;
    printf("%d %d",sizeof(s1),sizeof(s2));
}
```

a. 2 2 b. 4 2 c. 4 4 d. 2 4

72. What would be the output of the following program

```
int x=40;
main()
 {
     int x=20;
     printf("%d",x);
 }
```

a. 40 b.20 c. 60 d. none

73. What is the output of the following program

```
main()
{
      int x=40;
      {
          int x=20;
          printf("%d",x);
       }
      printf("%d",x);
}
```

a. 40 40 b. 20 40 c. 40 20 d. 20 20

74. What would be the output of the following program

```
main()
{
        extern int I;
        I =20;
        printf("%d",sizeof(I));
}
```

a. 2 b.4 c.varies from compiler to compiler d. error,I is undefined

75. Which of the following is correct
   a. a global variable has several declarations but only one definition
   b. if we use extern it is a declaration and not definition
   c. a function may have several declarations but only one definition
   d. all

76. What is the output of the following program

```
main()
{
    display();
}
void display()
{
        printf("ECET");
}
```

a. ECET b. ecet c. either d. error

77. What is the output of the following program

```
main()
{
    extern int fun(float);
    int a;
    a=fun(3.14);
    printf("%d",a);
}
int fun(aa)
float aa;
{
    return((int)aa);
}
```

a. 3 b. 3.14 c. 0 d. error

78. What would be the output of the following program

```
main()
{
    struct emp
    {
        char name[20];
        int age;
        float sal;
    };
    struct emp e={"puli"};
    printf("%d %f",e.age,e.sal);
}
```

a. 0 0.0000000 b. garbage values c. error d.0 0

79. What is the output of the following program

```
F();
main()
{    int (*p)() = F;
     (*p)();

}
F()
{      printf("Donot waste time");

}
```

a. Donot waste time b. Donot study Enjoy c. error d. none

80. What is the output of the following program

```
main()
{
    int I=8;
    while( )
    {
        printf("%d",I++);
        if(I > 10)
            break;
    }
}
```

a. the condition in while loop is a must b. 8 9 10 11
c. infinite loop d. none

81. What is the output of the following program

```
main()
{
    int I=1;
    while(I<=3)
    {
        printf("%d",I);
        goto  america;
    }
    india:
        printf("I");
}
func()
{
    america :
            printf("h");
}
```

a. 1h2h3h b. infinite loop c. error d. the control goes to india not america

82. What is the output of the following program

```
main()
```

```
{
        int a =10;
        switch(a)
        {

        }
        printf(“Always be an Indian”);
}
```

a. Always be an Indian b. 10 c. error d. none

83. What is the output of the following program

```
main()
{
        int a=2,b;
        a>=5?b=100:b=200;
        printf(“%d”,b);
}
```

a. 100 b.200 c. error d. garbage value

84. What is the output of the following program

```
main()
{
          char str[]=”Come on dance. This is the day to celebrate”;
          int a=5;
           printf(a>10?”%40s”:”%s”,str);
}
```

a. Come on dance. This is the day to celebrate b. Come

c. Come on celebrate. d. Error

85. What is the output of the following program

```
main()
{
        static int a[20];
        int I=0;
        a[I]=I++;
        printf(“%d %d %d”,a[0],a[1],I);
}
```

a. 0 1 0 b. 0 0 1 c. 0 1 1 d. none

86. What is the output of the following program

```
main()
{
        int j=3;
        j= ++j + ++j;
        printf(“%d”,j);
}
```

a. 8 b. 9 c. 10 d. none

87. What is the output of the following program

```
main()
```

```
{
    int j=2;
    printf("%d %d",++j,++j);
}
```

a. 3 4 b. 4 3 c.2 2 d. 4 4

88. What is the output of the following program

```
main()
{
    int x=10,y=20,z=5,I;
    I = x<y<z;
    printf("%d",I);
}
```

a. 0 b.1 c. error d.none

89. In the following code in which order the functions would be called

```
a= f1(23) * f2( 36)  + f3();
```

a. f1,f2,f3 b. f3,f2,f1 c. varies from compiler to compiler d. none

90. What would be the output of the following program

```
main()
{
    int I = -3,j=2,k=0,m;
    m = ++I || ++j && ++k;
    printf("%d %d %d %d",I,j,k,m);
}
```

a. –3 2 0 1 b. –2 2 0 1 c. –2 3 0 –1 d. none

91. What would be the output of the following program

```
main()
{
    int I=-3,j=2,k=0,m;
    m = ++ I  && ++j && ++k;
    printf("%d %d %d %d",I,j,k,m);
}
```

a. –3 2 0 1 b. –2 3 1 1 c. –2 2 0 1 d. none

92. What is the output of the following program

```
main()
{
    float b=0.8;
    if( b < 0.8)
        printf("Good Morning");
    else
        printf("Good Night");
}
```

a. Good Morning b. Good Night c. Good Afternoon d. none

93. What is the output of the following program

```
main()
{
    float b=0.8;
    if( b < 0.8f)
      printf("Good Morning");
    else
      printf("Good Night");
}
```

a. Good Morning   b. Good Night   c. Good Afternoon   d. none

94. What is the output of the following program

```
main()
{
    printf("%f",sqrt(36.0));
}
```

a. 6.0   b. 6   c. 6.000000   d. some absurd result

95. If you want to round off x, a float , to an int value . The correct way to do so is

a. y=(int)(x+0.5)   b. y= int(x+0.5)   c. y=(int) x + 0.5   d. y=(int)((int)x+0.5)

96. Which of the following is wrong

a. float type takes 4 bytes and the format specifier used for it is %f
b. double type takes 8 bytes and the format specifier used for it is %lf
c. long double type takes 10 bytes the format specifier used for it is %Lf
d. none of the above

97. What would be the output of the following program

```
main()
{
        printf("%d %d %d",sizeof(2.13f),sizeof(2.13),sizeof(2.13l));
}
```

a. 4 4 4   b. 4 garbage garbage   c. 4 8 10   d. error

98. A float occupies 4 bytes. If the hexadecimal equivalent of each of these bytes is A,B,C and D , then when this float is stored in memory these bytes get stored in this order

a. ABCD   b. DCBA   c. 0xABCD   d. none

99. Which of the following statements is true

a. two return statements should never occur in a function
b. two return statements should never occur successively in a function
c. all functions except main() can be called recursively
d. recursion is faster than loops

100. Which of the following statements is true

a. two different near pointers contain two different addresses but refer to same location in memory
b. two different far pointers contain two different addresses but refer to same location in memory
c. two different huge pointers contain two different addresses but refer to same location in memory
d. NULL pointer is the same as uninitialised pointer

101. What error would the following function give on compilation

```
f(int a,int b)
{
    int a;
```

```
        a=20;
        return a;
  }
```

a. missing parenthesis in return statement
b. the function should be defined as int f(int a , int b)
c. redeclaration of a
d. none of the above

102. What would be the output of the following program

```
main()
{
       int a=10;
       void f( );
       a=f( );
       printf("%d",a);
}
void f( )
{
       printf("hi");
}
```

a. 2 b. 3 c. error d. none

103. What is the output of the following program

```
main()
{
   int b;
   b=f(20);
   printf("%d",b);
}
int f(int a)
{
     a >10 ? return(40) : return(50);
}
```

a. 40 b.50 c. error d. none

104. How many times the string will be printed

```
main()
{
     printf("I have one brother");
     main();
}
```

a. infinite number of times b. error
c. 65535 d. till the stack does not overflow

105. What would be output of the following program

```
#define SQR(x) x*x
 main()
 {
      int a,b=10;
      a=SQR(b+2);
```

```
        printf("%d",a);
   }
```

a. 144 b. b+2 * b+2 c.32 d. garbage

106. What is the output of the following program

```
   #define CUBE(x) (x*x*x)
   main()
   {
        int a,b=3;
        a=CUBE(++b);
        printf("%d %d",a,b);
   }
```

a. 27 3 b. 64 4 c. 216 6 d. none

107. What is the type of the variable *abc* according to the following definition

```
   #define FLOATPTR float *
   FLOATPTR xyz,abc;
```

a. float b. pointer to float c. both d.none

108. Which of the following is true

a. a header file should definetely have an extension .h

b. there is no error if a header file is included twice

c. a preprocessor can trap simple errors like missing declarations

d. NULL macro is defined in stdio.h

109. What is the output of the following program

```
   #define BUS Train
   main()
   {
        printf("BUS");
   }
```

a. BUS b. Train c. error d. none

110. The following program results in

```
   #define LOOP while(1)
   main()
   {
          LOOP
            Printf("Hi");
   }
```

a. error b. infinite loop c. prints Hi once d. none

111. What is the output of the following program

```
   #define MAX(a,b) (a>b?a:b)
   main()
   {
          int x;
          x=MAX(4+2,3+8);
          printf("%d",x);
   }
```

a. 6 b. 11 c. 17 d.error

112. What is the output of the following program

```
#define PRINT(int) printf("%d",int)
main()
{
      int x=2,y=3,z=4;
      PRINT(x);
      PRINT(y);
      PRINT(z);
}
```

a. 2 3 4 b. 4 4 4 c. 4 3 2 d. error

113. What is the output of the following

```
main()
{
      printf("BBC" "world");
}
```

a. BBC b. BBC world c.error d.none

114. What is the output of the following

```
#define str(x) #x
#define hstr(x) str(x)
#define proper divide
main()
{
      char *name = hstr(proper);
      printf("%s",name);
}
```

a. proper b. hstr c. divide d. error

115. What is the output of the following

```
#define sum(xy) printf(#xy " = %f",xy)
main()
{
      float a=3.5,b=5;
      sum(a+b);
}
```

a. 8.5 b. error c. a+b=8.5 d. none

116. What is the output of the following program

```
#define combine(s1,s2) s1 ## s2
main()
{
      float totalsales=2.5;
      printf("%f",combine(total,sales));
}
```

a. 2.5 b. totalsales c. error d. none

117. What is the output of the following program

```
main()
{
      printf("%c",4["neelesh"]);
}
```

a. e b. l c. neel d. error

118. Which of the following is true

a. void pointer cannot be used for arithmetic operations

b. *ptr++ and ++*ptr are the same

c. a structure cannot have pointer

d. we can have an array of bitfields

119. What would be the output of the following

```
main()
{
        char a[]="neelesh";
        printf("%d %d",sizeof(a),sizeof(*a));
}
```

a. 8 2 b. 7 1 c. 8 1 d. 8 8

120. What would be the output of the following program, if the array begins at address 5000.

```
main()
{
        int abba[] = { 6,4,7,3,3};
        printf("%u %d",abba,sizeof(abba));
}
```

a. 5000 2 b. 5000 10 c. 5002 10 d. error

121. What would be the output of the following program, if the array begins at the address 8000

```
main()
{
        float fff[20]={5,4,6};
        printf("%u %u",fff,&fff);
}
```

a. 8000 8000 b. 8000 5000 c. 8000 8004 d. negative values

122. What would be the output of the following program, if the array begins at the address 6000

```
main()
{
        int abc[]={3,4,5,4};
        printf("%u %u",abc+1,&abc+1);
}
```

a. 6001 6001 b. 6002 6002 c. 6002 7000 d. error

123. What would be the output of the following program

```
main()
{
        int a[]="hello";
        a="no hello";
        printf("%d",strlent(a));
}
```

a. 5 b. 8 c. 9 d. error

124. What would be the output of the following program

```
main()
{
    double x[]={2.3,34.4,4.5};
    printf("%d", sizeof(x)/sizeof(x[1]));
}
```

a. 2 b. 3 c. 8 d. x is a double array it cannot be printed as integer

125. What would be the output of the following program, if the array begins at the address 8000

```
main()
{
    int b[3][4] ={ 4,3,2,5,6,3,2,7,7,6,5,4};
    printf("%u %u",b+1,&b+1);
}
```

a. 8001 8001 b. 8002 8002 c. 8002 8024 d.8008 8024

126. What is the output of the following program

```
main()
{
        printf("kishore"+2);
}
```

a. kishore b. ki c. shore d. error

127. What is the output of the following program

```
main()
{
    char s1[]="right",s2[]="right";
    if(s1==s2)
        printf("yes");
    else
        printf("no");
}
```

a. yes b. no c. error d. none

128. What would be the output of the following program in C

```
main()
{
    char xyz[4]="Balu";
    printf("%s",xyz);
}
```

a. Balu b. error c. cannot predict d. none

129. What is the output of the following program

```
main()
{
    int k=5;
    printf("%d %d  %d",k==5,k<30,k=80);
}
```

a. 1 1 80 b. 0 0 80 c. 5 30 80 d. none

130. What is the output of the following

```
main()
{
        char ch='a';
        printf("%d %d %d %d",sizeof(ch),sizeof('a'),sizeof(4),sizeof("hello"));
}
```

a. 1 1 2 5   b. 1 2 2 6   c. 1 1 2 6   d. 1 2 2 5

131. Whatt would be the output of the following program

```
main()
{
      printf("%c","jayaram"[5]);
}
```

a. a   b. r   c. error   d. jayaram

132. What is the output of the following program

```
main()
{
      struct student
      {
         char  *x;
         int age;
      };
     struct student s1 = { "mahesh",21};
     struct student s2=s1;
     printf("%s",s2.x);
}
```

a. mahesh   b. garbage   c. error in assigning structure   d. none

133. What is the output of the following program

```
main()
{
      struct student
      {
         char  *x;
         int age;
      };
     struct student s1 = { "mahesh",21};
     struct student s2=s1;
     if(s1==s2)
       printf("hello");
}
```

a. hello   b. no output

c. error in assigning structure variables   d. error in comparing structure variables

134. Which of the following is false

a. a structure can have a pointer to itself

b. structures can be passed as arguments in functions

c. bit fields of structures are used to save space in structures

d. the size of all the elements in an union should be same

135. What is the output of the following program

```
main()
{
        union a
        {
                int x;
                char ch[2];
        };
        union a z1={2,3};
        printf("%d",z1.x);
}
```

a. 2 b. 3 c. initialization error d. none

136. What would be the output of the following program

```
main()
{
        struct student
        {
                int rollno : 5;
                char name[20] : 6;
        };
        printf("%d",sizeof(struct student));
}
```

a. 2 b. 22 c. 11 d. error

137. Which of the following is false about bit fields

a. there can be unnamed fields declared with size
b. we can scan the bit fields using scanf statement
c. bit fields can be arrayed
d. there can be unused bits in the words

138. What is the output of the following program

```
main()
{
        float b=6.15529;
        printf("%6.2f",b);
}
```

a. 6.16 b. 6.15 c. 000006.15 d. errror

139. What is the output of the following program

```
main()
{
        float d=8.7;
        printf("%0.0f",d);
}
```

a. 8.7 b. 8 c. 0 d. error

140. In the following code

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

```
        fp=fopen("xyz","r");
    }
```

fp points to

a. the first character of the file

b. a structure which contains a char pointer which points to the first character of file

c. the name of the file d. None

141. What is the output of the following program

```
main()
{
    printf("%%%%%%");
}
```

a. %%%%%% b. %%% c. %%%% d. error

142. What is the output of the following program

```
main()
{
  int x=4;
  printf("x=%*d",x,x);
}
```

a. x=4 4 b. x=   4 c. x=4 d.error

143. Which of the following statement is false

a. we can specify variable field width in scanf()

b. we can specify variable field width in printf()

c. a file written in text mode cannot be read in binary mode

d. we should not write a file without an intervening call to fflush(),fseek(),rewind()

144. Which of the following statement is true

a. the variable argc and argv are always local to main b. C is superset of C++

c. C is not case sensitive d. C is not free format

145. The maximum combined length of the command line arguments including the spaces between adjacent arguments is

a. 128 characters b. 256 characters

c. 67 characters d. depends on operating system

146. If the program (xyz) is run from the command line as What is the output

xyz red green blue

```
main(int argc,char *argv[])
{
     while(argc)
       printf("%s",argv[--argc]);
}
```

a. xyz red green blue b. xyz red green c. blue green red xyz d. garbage

147. If the program (xyz) is run from the command line as What is the output

xyz red green blue

```
main(int argc,char *argv[])
{
    printf("%c",*++argv[2]);
}
```

a. e b. r c. g d. error

148. What would be the output of the following program

```
main()
{
    int I=32,j=0x20,k,l,m;
    k=I|j;
    l = I &j;
    m= k^I;
    printf("%d %d %d %d %d",I,j,k,l,m);
}
```

a. 32 32 32 32 0 b. 0 0 0 0 0 c. 0 32 32 32 32 d. 32 32 32 32 32

149. What is the output of the following program

```
main()
{
    unsigned int m=32;
    printf("%x",~m);
}
```

a. ffff b. 0000 c. ffdf d. ddfd

150. Which of the following is false

a. & operator is used for checking a particular bit is on or off
b. & operator is used for turning off a particular bit
c. | operator is used for putting a particular bit on
d. left shifting rotates the left most bits to the right end

151. What is the output of the following program

```
main()
{
     unsigned int a =0xffff;
     ~a;
      printf("%x",a);
}
```

a. ffff b. 0 c. 00ff d. error

152. What would be the output of the following program

```
main()
{
      unsigned char I=0x80;
       printf("%d",I<<1);
}
```

a. 0 b. 256 c. 100 d. none

153. What is the output of the following program

```
main()
{
        printf("%x",-1>>4);
}
```

a. ffff b. 0fff c. 0000 d.depends on computer architecture

154. If the definition is given in the following way which is false

```
#define xyz char *
xyz a,b;
```

a. a is a character pointer b. b is a character pointer

c. b is character d. error

155. What would be the output of the following program

```
main()
{
        int y=128;
        const int x=y;
        printf("%d",x);
}
```

a. 128 b. garbage c. error d. 0

156. What is the output of the following program

```
int get()
{
   return(30);
}
main()
{
        const int x=get();
        printf("%d",x);
}
```

a. 30 b. garbage c. error d. 0

157. What would be the output of the following

```
main()
{
     const int x=5;
     int *ptr;
     ptr=&x;
     *ptr =10;
     printf("%d",x);
}
```

a. 5 b. 10 c. error d. garbage

158. What would be the output of the following

```
main()
{
     const int x=5;
     const int *ptr;
     ptr=&x;
     *ptr =10;
     printf("%d",x);
}
```

a. 5 b. 10 c. error d. garbage

159. What is the output of the following

```
main()
{
        const int k=9;
        int * const p=&k;
        printf("%d",*p);
```

}

a. 9 b. garbage c. error d. none

160. Choose the correct answer

a. use of goto enhances the logical clarity of a code

b. use of goto makes the debugging task easier

c. use goto when you want to jump out of a nested loop

d. we can never use goto

161. Which is true about conditional compilation

a. it is taken care by the compiler b. it is setting the compiler option conditionally

c. it is compiling based on condition d. none of the above

162. C was primarily developed as

a. systems programming language b. general purpose language

c. data processing language d. none

163. C is a

a. high level language b. low level language

c. high level language with low level features d. none

164. The variables which can be accessed by all modules in a program are known as

a. local variables b. internal variables c. external variables d. global variables

165. In what kind of storage structure for strings, one can easily insert,delete, concatenate and rearrange substrings

a. fixed length storage structure b. variable length storage

c. linked list storage d. array type storage

166. The variables which can be accessed by all modules in a program, are known as

a. local variables b. internal variables c. external variables d. global variables

167. In what kind of storage structure for strings, one can easily insert, delete, concatenate and rearrange substrings

a. fixed length storage structure b. variable length storage

c. linked list storage d. array type storage

168. What is the time complexity of linear search algorithm over an array of n elements

a. O(log n) b. O(n) c. O(nlogn) d. $O(n^2)$

169. What is the time taken by binary search algorithm to search a key in a sorted array of n elements

a. O(log n) b. O(n) c. O(nlogn) d. O(1)

170. What is the time required to search an element in a linked list of length n

a. O(log n) b. O(n) c. O(1) d. O(nlog n)

171. What is the worst case time required to search a given element in a sorted linked list length n

a. O(1) b. O(log n) c. O(n) d. O(nlog n)

172. Consider a linked list of n elements which is pointed by an external pointer. What is the time taken to delete the element which is the successor of the element pointed to by a given pointer

a. O(1) b. O(log n) c. O(n) d. O(n log n)

173. Consider a linked list of n elements. What is the time taken to insert an element after an element pointed by some pointer

a. O(1) b. O(log n) c. O(n) d. O(n log n)

174. Which of the following operations is performed more efficiently by doubly linked

list than by linear linked list

a. deleting a node whose location is given

b. searching an unsorted list for a given item

c. inserting a node after the node with a given location

d. traversing the list to process each node

175. Which data structure is needed to convert infix notations to postfix notation

a. linear list b. queue c. tree d. stack

176. Recursive procedures are implemented by

a. queues b. stacks c. linked lists d. strings

177. A linear list of elements in which deletion can be done from one end (front) and Insertion can take place only at the other end(rear) is known as

a. queue b. stack c. tree d. deque

178. A linear list in which elements can be added or removed at either end but not in the middle is known as

a. queue b. deque c. stack d. tree

179. A list of integers is read in , one at a time , and a binary search tree is constructed . next the tree is traversed and the integers are printed. Which traversal would result in a printout which duplicates the original order of the list of integers

a. preorder b. postorder c. inorder d. none

180. The five items : A,B,C,D and E are pushed in a stack , one after the other starting from A. The stack is popped four times and each element is inserted in a queue. The two elements are deleted from the queue and pushed back on the stack . Now one item is popped from the stack. The popped item is

a. A b. B c. C d. D

181. The time required to search an element in a binary search tree having n elements is

a. O(1) b. O(log n) c. O(n) d. O(n log n)

182. Consider that n elements are to be sorted . What is the worst case time complexity of bubble sort

a. O(1) b. O(log n) c. O(n) d. $O(n^2)$

183. Consider that n elements are to be sorted. What is the worst case time complexity of shell sort

a. O(n) b. O(n log n) c. $O(n^{1.2})$ d. $O(n^2)$

184. What is the worst case time complexity of straight insertion sort algorithm to sort n elements

a. O(n) b. O(n log n) c. $O(n^{1.2})$ d. $O(n^2)$

185. What is the worst case time complexity of binary insertion sort algorithm to sort n elements

a. O(n) b. O(n log n) c. $O(n^{1.2})$ d. $O(n^2)$

186. If each node in a tree has value greater than every value in its left subtree and has Value less than every value in its right subtree, the tree is known as

a. complete tree b. full binary tree c. binary search tree d. threaded tree

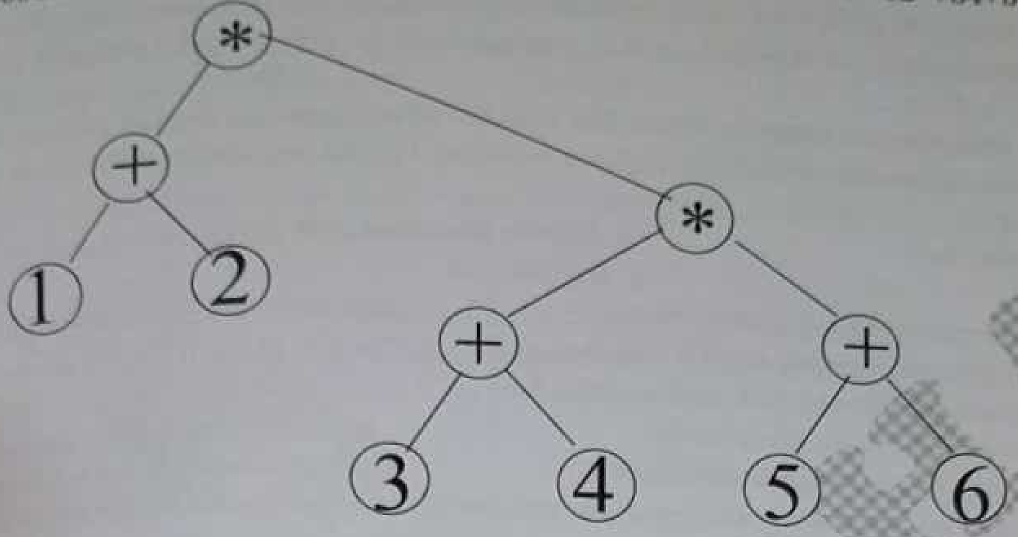
187. Which of the following procedure is the slowest

a. quick sort b. heap sort c. shell sort d. bubble sort

188. The infix expression A+(B-C)*D is correctly represented in prefix notation as

a. A+B-C*D b. +A*- BCD c. ABC-D*+ d. A+BC-D*

189. What is the postfix expression of the following tree
a. 655321++**+ b. 1+2*3+4*5+6 c. 12+34+56+** d.*+12*+34+56



190. A list of data items usually words or bytes, with the accessing restriction that elements can be added or removed at one end of the list only, is known as
a. stack b. memory c. linked list d. heap

191. In what order the elements of a pushdown stack are accessed
a. FIFO b. LIFO c. LILO d. none

192. Which of the following is a tabular listing of contents of certain registers and Memory locations at different times during the execution of a program
a. loop program b. program trace c. subroutine program d. sorting program

193. How many values can be held by an array A(-1..m, 1..m)
a. m b. 2m c. m(m+1) d. m(m+2)

194. An undirected graph with n vertices and e edges are represented by adjacency Matrix. What is the time required to determine whether the graph is connected
a. O(e) b. O(n) c. O(n2) d. O(e+n)

195. An undirected graph with n vertices and e edges are represented by adjacency Matrix. What is the time required to determine the degree of any vertex
a. O(e) b. O(n) c. O(n2) d. O(e+n)

196. A directed graph with n vertices and e edges are represented by adjacency Matrix. What is the time required to determine the in-degree of any vertex
a. O(e) b. O(n) c. O(n2) d. O(e+n)

197. Which of the following statements is false
a. every tree is bipartite graph b. a tree contains a cycle
c. a tree with n nodes contains n-1 edges d. a tree is a connected graph

198. A graph G with n nodes is bipartite if it contains
a. n edges b. a cycle of odd length c. no cycle of odd length d. none

199. In what tree, for every node the height of its left subtree and right subtree differ atleast by one
a. binary search tree b. AVL tree c. complete tree d. threaded tree

200. Which of the following sorting method is stable
a. straight insertion sort b. binary insertion sort
c. shell sort d. heap sort

201. A complete binary tree with the property that the value at each node is at least as Large as the values at its children is known as
  a. binary search tree b. AVL tree c. completely balanced tree d. Heap

202. The time required to find the shortest path in a graph with n vertices and e edges is
  a. O(e) b. O(n) c. O(e+n) d. $O(n^2)$

203. In which of the following sorting algorithm the number of comparision needed is the minimum if the items are initially in reverse order and is the maximum if the items are in order
  a. straight insertion sort b. binary insertion sort
  c. heap sort d. bubble sort

204. Which of the following best describes sorting
  a. accessing and processing each record exactly once
  b. finding the location of the record with a given key
  c. arranging the data in some given order
  d. adding a new record to the data structure

205. The order of magnitude of the worst case performance of the binary search over n elements is
  a. nlog n b. n c. n^2 d. log n

206. A search procedure which associates an address with a key value and provides a Mechanism for dealing with two or more values assigned to the same address is called
  a. linear search b. binary search c. hash coded search d. radix search

207. The order of magnitude of the worst case performance of a hash coded search over n elements is
  a. n b. n log n c. log n d. not dependent on n

208. A characteristic of the data that binary search uses but the linear search ignores is
  a. order of the list b. length of the list
  c. maximum value of the list d. mean of data values

209. A sort which compares adjacent elements in a list and switches where necessary is
  a. insertion sort b. heap sort c. quick sort d. bubble sort

210. A full binary tree with n leaves contains
  a. n nodes b. log n nodes c. 2n-1 nodes d. 2n +1 nodes

211. A full binary tree with n non-leaf nodes contains
  a. log n nodes b. n+1 nodes c. 2n-1 nodes d. 2n+1 nodes

212. Which of the following sorting algorithms does not have the worst case running time of O(n2)
  a. insertion sort b. merge sort c. quick sort d. bubble sort

213. Which of the following sorting algorithms has a worst case running time of O(nr) where 1<r<2
  a. bubble sort b. insertion sort c. shell sort d. merge sort

214. What is the maximum number of fields with each node of doubly linked list
  a. 1 b. 2 c. 3 d. 4

215. Given two sorted list of size m and n respectively. The number of comparisions needed in the worst case by the merge sort algorithm will be
  a. mn b. max(m,n) c. min(m,n) d. m+n-1

216. Preorder is nothing but
  a. depth first order b. breadth first order

c. topological order d. linear order

217. Which traversal techniques lists the nodes of a binary search tree in ascending order

a. post order b. in order c. pre order d. none

218. The initial configuration of queue is a,b,c,d (a at the front). To get the configuration d,c,b,a one needs a minimum of

a. 2 deletions and 3 insertions b. 3 deletions and 2 insertions

c. 3 deletions and 3 insertions d. 3 deletions and 4 insertions

219. The number of swappings need to sort the numbers 8,22,7,9,31,19,5,13 in Ascending order , using bubble sort is

a. 11 b. 12 c. 13 d. 14

220. There are four algorithms namely A1,A2,A3,A4 with time complexities O(log n), O(log log n) , O(n log n) , O(n/ log n) respectively. Which is the best algorithm

a. A1 b. A2 c. A3 d. A4

221. Queues serve a major role in

a. simulation of recursion b. simulation of arbitrary linked list

c. simulation of limited resource allocation d. expression evaluation

222. What is the number of nodes in a complete binary tree of level 5

a. 15 b. 25 c. 63 d. 71

Key

1–A	2–C	3–B	4–B	5–A	6–D	7–B	8–D	9–D	10–C
11–B	12–B	13–A	14–B	15–A	16–A	17–C	18–A	19–D	20–B
21–D	22–B	23–A	24–A	25–B	26–D	27–D	28–D	29–B	30–D
31–A	32–D	33–A	34–C	35–B	36–D	37–B	38–C	39–A	40–C
41–A	42–D	43–D	44–B	45–B	46–D	47–B	48–A	49–C	50–C
51–B	52–B	53–A	54–C	55–A	56–B	57–D	58–C	59–D	60–B
61–C	62–A	63–A	64–A	65–A	66–C	67–C	68–A	69–C	70–C
71–B	72–B	73–B	74–D	75–D	76–D	77–D	78–A	79–A	80–A
81–C	82–A	83–C	84–A	85–B	86–C	87–B	88–B	89–C	90–B
91–B	92–A	93–B	94–D	95–A	96–D	97–C	98–B	99–B	100–B
101–C	102–C	103–C	104–D	105–C	106–C	107–A	108–D	109–A	110–B
111–B	112–A	113–B	114–C	115–C	116–A	117–B	118–A	119–C	120–B
121–A	122–C	123–D	124–B	125–D	126–C	127–B	128–C	129–B	130–B
131–A	132–A	133–D	134–D	135–C	136–A	137–C	138–A	139–B	140–B
141–B	142–B	143–A	144–A	145–D	146–C	147–B	148–A	149–C	150–D
151–A	152–B	153–D	154–B	155–A	156–A	157–B	158–C	159–C	160–C
161–C	162–A	163–C	164–D	165–C	166–D	167–C	168–B	169–A	170–B
171–C	172–A	173–A	174–A	175–D	176–B	177–A	178–B	179–D	180–D
181–B	182–D	183–C	184–D	185–D	186–C	187–D	188–B	189–C	190–A
191–B	192–B	193–D	194–C	195–B	196–B	197–B	198–C	199–B	200–B
201–D	202–D	203–B	204–C	205–D	206–C	207–D	208–A	209–D	210–C
211–D	212–B	213–C	214–C	215–D	216–A	217–B	218–C	219–D	220–B
221–C	222–C								