QUESTION BANK

EASWARI ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

CS6202
PROGRAMMING
AND
DATA STRUCTURES I

QUESTION BANK

I YEAR IT A & B
JANUARY 2015 TO MAY 2015

PREPARED BY

Mrs. M.SOWMIYA

APPROVED BY

HOD

EASWARI ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

CS6202 PROGRAMMING AND DATA STRUCTURES-I

UNIT-I

PART A

1. **Define Program.**
   A computer program is a set of instructions for a computer to perform a specific task.

2. **What is algorithm?**
   Algorithm means the logic of a program. It is a step-by-step description of how to arrive at a solution of a given problem

3. **What are the characteristics of an algorithm?**
- Every instruction should be precise.
- Every instruction should be unambiguous.
- The instructions should not be repeated infinitely.
- Ensure it will ultimately terminate.
- Instructions should be written in sequence.

4. **Name the different programming paradigms.**
- Imperative — Control flow is an explicit sequence of commands.
- Declarative — Programs state the result you want, not how to get it.
- Structured — Programs have clean, goto-free, nested control structures.
- Procedural — Imperative programming with procedure calls.
- Object-Oriented — Computation is effected by sending messages to objects; objects have state and behavior.

5. **What is structured programming?**
   In structured programming, the problem to be solved is broken down into small tasks that can be written independently. Once written, the small tasks are combined together to form the complete task.

6. **What is an array?**
   An array is a collection of variables of the same data type stored contiguously under a common name.
   Example: int a[10];
   Here a[10] is an array with 10 integer values.

7. **What are the main elements of an array declaration?**
- Name of the array
- Type of the elements
- Size of the array

8. How to initialize an array?

You can initialize an array in C either one by one or using a single statement. The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].

(e.g) // Initializing elements one by one

```
double balance[5];
balance[0]=1000.0;
  .
  .
  .
balance[4]=50.0
```

```
//single statement to initialize entire array
    double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
```

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

When an array is declared, the compiler allocates a base address and reserves enough space in the memory for all the elements of the array. Hence, the size must be given in the array declaration.

10. What is a Pointer? How a variable is declared to the pointer?

Pointer is a variable which holds the address of another variable.

Pointer Declaration: datatype *variable-name;

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

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

Array	Pointer
An array is a collection of variables of the same data type stored contiguously under a common name.	A pointer is a variable that holds the address of a variable or a function
They are static in nature. Once memory is allocated , it cannot be resized or freed dynamically.	Pointer is dynamic in nature. The memory allocation can be resized or freed later.
Array can be initialized at definition. Example: int num[] = { 2, 4, 5}	Pointer can't be initialized at definition.
Size of(array name) gives the number of bytes occupied by the array.	Sizeof(pointer name) returns the number of bytes used to store the pointer variable.

12. What are the uses of Pointers?

Pointers are used to return more than one value to the function

- Pointers are more efficient in handling the data in arrays
- Pointers reduce the length and complexity of the program
- They increase the execution speed
- The pointers save data storage space in memory

13. What are * and & operators?

'*' operator is called 'value at the address' operator

'&' operator is called 'address of' operator

14. What is a dangling pointer?

A dangling pointer points to memory that has already been freed. The storage is no longer allocated. It arises when we use the address of an object after its lifetime is over.

15. What is dynamic memory allocation?

Allocating the memory at run time is called as dynamic memory allocation.

16. What are the various dynamic memory allocation functions?

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

17. What are functions?

A function is a group of statements that together perform a task. Functions are used to organize programs into smaller and independent units.

18. State the advantages of modularization.

- Reduction in code redundancy
- Enabling code reuse
- Better readability
- Information hiding
- Improved maintainability

19. What are actual and formal parameters?

Actual parameters are variables that are passed by the caller in function call.
Example:

```
Sum (a, b)  // a and b are actual parameters or actual arguments
```

Formal parameters are variables that are used in header of a function definition.
Example:

```
int Sum (int x, int y)  // x and y are formal parameters
{  …  }
```

20. What are the different ways of passing arguments to a function?

Call Type	Description
Call by value	This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.
Call by reference	This method copies the address of an argument into the formal parameter. Inside the function, the address is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.

21. Why do we include header files in our programs?

The declarations of library functions are available in their respective header files. To access those functions in user defined programs, the corresponding header files are to be included.

22. Define Recursion.

A function that calls itself is known as recursive function and the phenomenon is called recursion.

23. Differentiate Iteration and Recursion.

Iteration	Recursion
Recursion Uses selection structure	Iteration uses repetition structure
Infinite recursion occurs if the recursion step does not reduce the problem in a manner that converges on some condition.(base case)	An infinite loop occurs with iteration if the loop-condition test never becomes false
Recursion terminates when a base case is recognized	Iteration terminates when the loop-condition fails
Recursion is usually slower then iteration due to overhead of maintaining stack	Iteration does not use stack so it's faster than recursion
Recursion uses more memory than iteration	Iteration consume less memory
Infinite recursion can crash the system	infinite looping uses CPU cycles repeatedly

24. Define pre-processor in C.

The preprocessor is a translator that works and processes the source code before it is given to the compiler. It operates under the control of commands known as preprocessor directives.

25. List the various pre-processor directives in C?

- Macro Inclusion directive
- Conditional Inclusion directive
- File Inclusion directive
- Line directive
- Error directive
- Null directive
- Pragma directive

26. What is a macro? List its types.

A *macro* is a fragment of code which has been given a name. Whenever the name is used, it is replaced by the contents of the macro.

There are two kinds of macros:

- *Object-like* macros – Macros without arguments
- *function-like* macros – Macros with arguments

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

The choice between using a macro and using a function is one of deciding between the tradeoff of faster program speed versus smaller program size. Generally, use macros to replace small, repeatable code sections, and use functions for larger coding tasks that might require several lines of code.

28. What are the various ways in which a Source file Inclusion directive can be written?

There are three ways to specify a source file inclusion directive:

1 *#include " filename"*

2 *#include <filename>*

3 #include token-sequence

29. What is a token? List the types of token.

A token is the smallest indivisible element of C language.

Types of tokens:

- Keywords
- Identifiers
- Constants
- String Literals
- Punctuators (operators, separators, terminators)

30. What are conditional Inclusions in Preprocessor Directive?

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

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

Part-B

1. Explain in detail about the Decision making statements.
2. Discuss about the various Looping statements.
3. Explain in detail about C tokens.
4. Write a C program to calculate the average of a set of N numbers using While loop.
5. Write a C program to print the largest of three numbers using If..else.
6. Write a C program to find the root of the quadratic equation.
7. Write a C program to determine whether the given number is odd or even, positive or negative
8. Write a C program to find the factorial of the given number.

UNIT II

1. Define Structure.

Structure is a collection of different data types which are grouped together; each element in a structure is called member. To access structure members in C, a structure variable should be declared.

2. Compare arrays and structures.

Arrays	Structures
An array is a collection of related data elements of same type.	Structure can have elements of different types
An array is a derived data type	A structure is a programmer-defined data type
Any array behaves like a built-in data types. All we have to do is to declare an array variable and use it.	But in the case of structure, first we have to design and declare a data structure before the variable of that type are declared and used.

3. What is a bit-field in structure?

The variables defined with a predefined width are called bit fields. A bit field can hold more than a single bit for example if you need a variable to store a value from 0 to 7 only then you can define a bit field with a width of 3 bits as follows:

```
struct
{
  unsigned int age : 3;
} Age;
```

The above structure definition instructs C compiler that the age variable uses only 3 bits to store the value.

4. What are unnamed bit-fields? Mention its need?

Bit-fields with length 0 are known as unnamed bit-fields. Unnamed bit-fields are used for the alignment purposes. An unnamed bit-field indicates that the next field should be placed in a separate unit and not with the previous field in the same unit.

5. Mention the advantages of using pointer to structures.

- Easier to manipulate the pointer to structures than manipulating structures.
- Passing a pointer to a structure as an argument to a function is efficient as compared to passing a structure to a function.

6. Can a structure have a pointer to itself?

Yes, a structure can have a pointer to an instance of itself. Such a structure is known as referential structure.

7. Mention the two ways of accessing a structure object.

- Direct Member access operator (i.e, . dot operator)
- Indirect Member access operator (i.e, -> arrow operator)

8. List the three ways of passing a structure object to a function.

- Passing each member of a structure object as a separate argument
- Passing the entire structure object by value.
- Passing the structure object by address/reference

9. What are anonymous structures?

Unnamed structure is known as anonymous structure. The tag names are not specified while defining anonymous structure.

10. Mention the categories of operations on structure.

- Aggregate operations
- Segregate operations

11. What is the used of typedef in structure?

typedef defines and names a new type, allowing its use throughout the program. typedefs usually occur just after the #define and #include statements in a file.

(e.g)
```
typedef struct {
    char name[64];
    char course[128];
    int age;
    int year;
    } student;
```
This defines a new type student; variables of type student can be declared as follows: student st_rec;

12. Is it possible for a structure to have an instance of its own?

No. Structure cannot have an instance of its own. For example,
```
struct regression {
    int int_member;
    struct regression self_member;
  };
```
The following error messages will be dispalyed
```
struct.c: In function `main':
struct.c:8: field `self_member' has incomplete type
```

13. What is meant by Union in C?

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

14. How to define a union in C?

The syntax of the union definition is as follows:
```
union [union tag]
{
member definition;
...
member definition;
} [one or more union variables];
```

15. How can you access the members of an union?

To access any member of an union, use the **member access operator (.)**. The member access operator is coded as a period between the union variable name and the union member that we wish to access. Use union keyword to define variables of union type.

16. What is a file?

File is named location of stream of bits. A file is a collection of bytes stored on a secondary storage device, which is generally a disk of some kind. The collection of bytes may be interpreted, for example, as characters, words, lines, paragraphs and pages from a textual document; fields and records belonging to a database.

17. What is a stream?

Stream is a sequence of bytes of data. A sequence of bytes flowing into a program is called input stream. A sequence of bytes flowing out the program is called output stream.

18. Differentiate Binary file and Text file.

Binary File	Text File
A binary file is treated as raw data and read byte-by-byte.	A text file is considered to contain lines of text that are separated by some end-of-line markings.
Newline handling conversions will not takes place	A newline character is converted into carriage return-line feed combination and vice versa, before being return to the disk
Storage of numbers occupy same disk space as memory	Storage of numbers require more disk space than in memory

19. Mention the different file open modes.

Modes	Description
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file

x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

UNIT-III

PART A

1. **Define: data structure.**
   A data structure is a way of storing and organizing data in the memory for efficient usage.The way information is organized in the memory of a computer.

2. **Give few examples for data structures?**
   Arrays, stacks, queue, list, tree, graph, set, map, table and deque.

3. **What are the different types of data structures?**
   i) Primitive
   ii) Composite
   iii) Abstract

4. **What are primitive data types?**
   The basic building blocks for all data structures are called primitive data types.
   (e.g) int, float, char, double, Boolean

5. **What are composite data types?**
   Composite data types are composed of more than one primitive data type.
   (e.g) array,structure,union

6. **What is meant by an abstract data type?**
   An ADT is a mathematical model for a certain class of data structures that have similar behavior. (e.g) list, stack, queue

7. **How can we categorize data structures based on data access?**
   Linear – list, stack, queue
   Non-linear- heap, tree, graph

8. **State the difference between linear and non-linear data structures.**
   The main difference between linear and nonlinear data structures lie in the way they organize data elements.
   In linear data structures, data elements are organized sequentially and therefore they are easy to implement in the computer's memory.
   In nonlinear data structures, a data element can be attached to several other data elements to represent specific relationships that exist among them. Due to this it might be difficult to be implemented in computer's linear memory.

9. **List a few real-time applications of data structures?**
   - Undo and redo feature - stack
   - Decision making - graph
   - Printer (printing jobs) – queue
   - Compilers – hash table
   - Directory structure- trees
   - Communication networks- graphs

10. Define List.

The general form of the list is a_1, a_2, a_3… a_n. The size of the list is 'n'. Any element in the list at the position i is defined to be at a_i, a_{i+1} the successor of a_i, and a_{i-1} is the predecessor of a_i. a_1 doesn't have predecessor and a_n doesn't have successor.

11. What are the various operations done on List ADT?

The operations done under List ADT are Print list, Insert, Delete, FindPrevious, Find k^{th}, Find, MakeEmpty, IsLast and IsEmpty.

12. What are the different ways to implement list?

- Array implementation of list
- Linked list implementation of list
- Cursor implementation of list

13. Arrays are not used to implement lists. Why?

- Requires that the list size to be known in advance
- Running time for insertions and deletions is slow

14. What are the advantages in the array implementation of list?

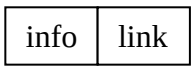
- Print list operation can be carried out at linear time
- Finding K^{th} element takes a constant time

15. What are the disadvantages in the array implementation of list?

The running time for insertions and deletions is so slow and the list size must be known in advance.

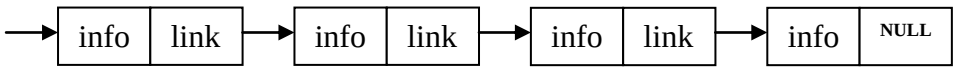
16. Define node.

A node consists of two fields namely an information field called INFO and a pointer field called LINK. The INFO field is used to store the data and the LINK field is used to store the address of the next field.



17. What is a linked list?

Linked list is series of nodes, which are not necessarily adjacent in memory. Each node contains a data element and a pointer to the next node.



18. Distinguish between linked lists and arrays. Mention their relative advantages and disadvantages.

Array is a collection of elements allocated contiguously in memory where individual elements can be accessed using subscript. A[i] refers to i^{th} element in the array A.
Linked list is series of nodes, which are not necessarily adjacent in memory. Each node contains the element and a pointer to the next node.

Array:

Advantage:

- PrintList operation can be carried out at linear time.
- Finding K^{th} element takes a constant time.

Disadvantage:

- Maximum size of the list is required, which will be an overestimate and wastage of space.
- Insertion and deletion are expensive.

Linked list:

Advantage:

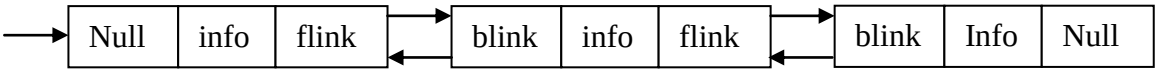
- Insert and delete requires only pointer change and so takes constant time.
- Maximum size of the list is not required.

Disadvantage:

- Finding K^{th} element takes O(i) time.
- Pointer in each node occupies extra space.

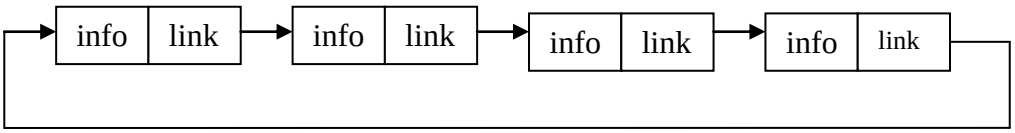
19. What is a doubly linked list?

In a doubly linked list, along with the data field there will be two pointers one pointing the next node(flink) and the other pointing the previous node(blink).



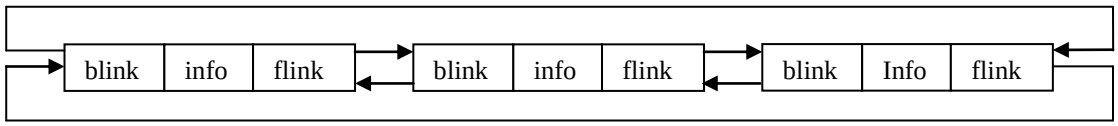
20. Define circularly linked list?

In a singly circular linked list the last node's link points to the first node of the list.



21. Define double circularly linked list?

In a circular doubly linked list the last node's forward link points to the first ode of the list, and the first node's back link points to the last node of the list.



22. Mention the disadvantages of circular list?

The disadvantage of using circular list is

- It is possible to get into an infinite loop.
- It is not possible to detect the end of the list.

23. What is the need for the header?

Header of the linked list is the first element in the list and it may store the number of elements in the list. It points to the first data element of the list.

Without header

- Insertion at the front of the list needs special coding & updating of the linked list address.
- Deletion at the front of the list also needs special coding & updating of the linked list address.

24. List three applications that uses linked list?

Three examples/applications that uses linked list are Polynomial representation, radix sort, & multi lists.

PART-B

1. What is an Abstract Data type (ADT)? Explain?
2. Derive an ADT to perform insertion and deletion in a singly linked list.(8) (Nov 10)
3. Explain the cursor implementation of linked list? (Nov 10)
4. Write the insertion and deletion procedures for cursor based linked lists.(8)(Nov 09)
5. Explain the various applications of linked list?
6. Design an algorithm to reverse the linked list. Trace it with an example?(8)
7. Write an algorithm for inserting and deleting an element from Doubly linked list?(8)
8. Write the algorithm for the deletion and reverse operations on doubly linked list. (8) (Nov 09)
9. Write algorithms to perform the following in doubly linked list: (may 10)
   (i) To insert an element in the beginning, middle, end of the list. (8)
   (ii) To delete an element from anywhere in the list. (8)
   An element is a structure variable that contains an integer data field and a string data field.

UNIT-IV

1. Write the definition of weighted graph?

A graph in which weights are assigned to every edge is called a weighted graph.

2. Define Graph?

A graph G consist of a nonempty set V which is a set of nodes of the graph, a set E which is the set of edges of the graph, and a mapping from the set of edges E to set of pairs of elements of V. It can also be represented as G=(V, E).

3. Define adjacency matrix?

The adjacency matrix is an n x n matrix A whose elements aij are given by
aij = 1 if (vi, vj) Exists
=0 otherwise

4. Define adjacent nodes?

Any two nodes, which are connected by an edge in a graph, are called adjacent nodes. For example, if an edge $x \in E$ is associated with a pair of nodes (u,v) where u, v $\in$V, then we say that the edge x connects the nodes u and v.

5. What is a directed graph?

A graph in which every edge is directed is called a directed graph.

6. What is an undirected graph?

A graph in which every edge is undirected is called an undirected graph.

7. What is a loop?

An edge of a graph, which connects to itself, is called a loop or sling.

8. What is a simple graph?

A simple graph is a graph, which has not more than one edge between a pair of nodes.

9. What is a weighted graph?

A graph in which weights are assigned to every edge is called a weighted graph.

10. Define indegree and out degree of a graph?

In a directed graph, for any node v, the number of edges, which have v as their initial node, is called the out degree of the node v.

Outdegree : Number of edges having the node v as root node is the outdegree of the node v.

11. Define path in a graph?

The path in a graph is the route taken to reach terminal node from a starting node.

12. What is a simple path?

A path in a diagram in which the edges are distinct is called a simple path.
It is also called as edge simple.

13. What is a cycle or a circuit?

A path which originates and ends in the same node is called a cycle or circuit.

14. What is an acyclic graph?

A simple diagram, which does not have any cycles, is called an acyclic graph.

15. What is meant by strongly connected in a graph?

An undirected graph is connected, if there is a path from every vertex to every other vertex. A directed graph with this property is called strongly connected.

16. When a graph said to be weakly connected?

When a directed graph is not strongly connected but the underlying graph is connected, then the graph is said to be weakly connected.

17. Name the different ways of representing a graph? Give examples (Nov 10)

a. Adjacency matrix
b. Adjacency list

18. What is an undirected acyclic graph?

When every edge in an acyclic graph is undirected, it is called an undirected acyclic graph. It is also called as undirected forest.

19. What is meant by depth?

The depth of a list is the maximum level attributed to any element with in the list or with in any sub list in the list.

20.What is the use of BFS?

BFS can be used to find the shortest distance between some starting node and the remaining nodes of the graph. The shortest distance is the minimum number of edges traversed in order to travel from the start node the specific node being examined.

21. Write BFS algorithm

1. Initialize the first node's dist number and place in queue
2. Repeat until all nodes have been examined
3. Remove current node to be examined from queue
4. Find all unlabeled nodes adjacent to current node
5. If this is an unvisited node label it and add it to the queue
6. Finished.

22.Define biconnected graph?

A graph is called biconnected if there is no single node whose removal causes the graph to break into two or more pieces. A node whose removal causes the graph to become disconnected is called a cut vertex.

23.Write the time complexly of BFS algorithm

The time analysis for the BFS algorithm is O(n + e) Where e is the number of edges and n is the number vertices.

24.What are the two traversal strategies used in traversing a graph?

a. Breadth first search
b. Depth first search

25.What is a minimum spanning tree? (Nov 09)

A minimum spanning tree of an undirected graph G is a tree formed from graph edges that connects all the vertices of G at the lowest total cost.

26.Define Stack

A Stack is an ordered list in which all insertions (Push operation) and deletion (Pop operation) are made at one end, called the top. The topmost element is pointed by top. The top is initialized to -1 when the stack is created that is when the stack is empty. In a stack S = (a1,…an), a1 is the bottom most element and element ai is on top of element ai-1. Stack is also referred as Last In First Out (LIFO) list.

27.What are the various Operations performed on the Stack?

The various operations that are performed on the stack are

CREATE(S) – Creates S as an empty stack.
PUSH(S,X) – Adds the element X to the top of the stack.
POP(S) – Deletes the top most elements from the stack.
TOP(S) – returns the value of top element from the stack.
ISEMTPTY(S) – returns true if Stack is empty else false.
ISFULL(S) - returns true if Stack is full else false.

28.How do you test for an empty stack?

The condition for testing an empty stack is top =-1, where top is the pointer pointing to the topmost element of the stack, in the array implementation of stack.

In linked list implementation of stack the condition for an empty stack is the header node link field is NULL.

29.Name two applications of stack?

- Nested and Recursive functions can be implemented using stack.
- Conversion of Infix to Postfix expression can be implemented using stack.
- Evaluation of Postfix expression can be implemented using stack.

30.Define a suffix expression.

The notation used to write the operator at the end of the operands is called suffix notation.

Suffix notation format : operand operand operator

Example: ab+, where a & b are operands and '+' is addition operator.

31.What do you meant by fully parenthesized expression? Give eg.

A pair of parentheses has the same parenthetical level as that of the operator to which it corresponds. Such an expression is called fully parenthesized expression.

Ex: (a+((b*c) + (d * e))

32.Write the postfix form for the expression -A+B-C+D?

A-B+C-D+

33.What are the postfix and prefix forms of the expression?

A+B*(C-D)/(P-R)

Postfix form: ABCD-*PR-/+

Prefix form: +A/*B-CD-PR

34.Explain the usage of stack in recursive algorithm implementation?

In recursive algorithms, stack data structures is used to store the return address when a recursive call is encountered and also to store the values of all the parameters essential to the current state of the function.

35. Define Queues.

A Queue is an ordered list in which all insertions take place at one end called the rear, while all deletions take place at the other end called the front. Rear is initialized to -1 and front is initialized to 0. Queue is also referred as First In First Out (FIFO) list.

36. What are the various operations performed on the Queue?

The various operations performed on the queue are

CREATE(Q) – Creates Q as an empty Queue.

Enqueue(Q,X) – Adds the element X to the Queue.

Dequeue(Q) – Deletes a element from the Queue.

ISEMTPTY(Q) – returns true if Queue is empty else false.

ISFULL(Q) - returns true if Queue is full else false.

37. How do you test for an empty Queue?

The condition for testing an empty queue is rear=front-1. In linked list implementation of queue the condition for an empty queue is the header node link field is NULL.

38. Write down the function to insert an element into a queue, in which the queue is implemented as an array. (May 10)

Q – Queue

X – element to added to the queue Q

IsFull(Q) – Checks and true if Queue Q is full

```
Q->Size - Number of elements in the queue Q
Q->Rear – Points to last element of the queue Q
Q->Array – array used to store queue elements
void enqueue (int X, Queue Q)
{       if(IsFull(Q))
          Error ("Full queue");
        else
         {
Q->Size++;
           Q->Rear = Q->Rear+1;
           Q->Array[ Q->Rear ]=X;
        }
}
```

39. Define Deque.

Deque stands for Double ended queue. It is a linear list in which insertions and deletion are made from either end of the queue structure.

40.Define Circular Queue.

Another representation of a queue, which prevents an excessive use of memory by arranging elements/ nodes $Q_1,Q_2,\ldots Q_n$ in a circular fashion. That is, it is the queue, which wraps around upon reaching the end of the queue

Part – B

1. Write an algorithm for Push and Pop operations on Stack using Linked list. (8)
2. Explain the linked list implementation of stack ADT in detail?
3. Define an efficient representation of two stacks in a given area of memory with n words and explain.
4. Explain linear linked implementation of Stack and Queue?
    a. write an ADT to implement stack of size N using an array. The elements in the stack are to be integers. The operations to be supported are PUSH, POP and DISPLAY. Take into account the exceptions of stack overflow and stack underflow. (8)
    b. A circular queue has a size of 5 and has 3 elements 10,20 and 40 where F=2 and R=4. After inserting 50 and 60, what is the value of F and R. Trying to insert 30 at this stage what happens? Delete 2 elements from the queue and insert 70, 80 & 90. Show the sequence of steps with necessary diagrams with the value of F & R. (8 Marks)
5. What is a Stack? Explain its operations with example?
6. Write the algorithm for converting infix expression to postfix (polish) expression?
7. Explain the applications of stack?
8. Write a C program to perform the following stack operations (may 10)
9. Declare a structure with one integer data type. Add necessary variables to ensure that you can create a linked representation.
10. Write a function called 'push' that takes two parameters: an integer variable and a stack into which it would push this element and returns a 1 or a 0 to show success of addition or failure. (6)
11. Write a function called 'pop' that takes two parameters: An integer pointer and a stack from which we need to pop this element. (6)

12. A display function that would print the contents of stack. Add necessary functions to check for emptiness or fullness of the stack. (4)
13. What is a Queue? Explain its operation with example?
14. Explain the array implementation of queue ADT in detail?
15. Explain the addition and deletion operations performed on a circular queue with necessary algorithms.(8) (Nov 09)
16. Explain the various representation of graph with example in detail?
17. Explain Shortest path algorithm with example?
18. Explain Depth first and breadth first traversal?
19. Explain spanning and minimum spanning tree?
20. What is topological sort? Write an algorithm to perform topological sort?(8) (Nov 09)
21. Find the topological ordering of a graph (8)(Nov 10)
22. Write Dijkstra's algorithm to find the shortest path? (8) (Nov 09/10)
23. (i) Give the Prim's algorithm for determining minimum spanning tree and analyse this algorithm for its worst case time complexity. (10) (May 10)

(ii) Determine the minimum spanning tree for a graph using prim's algorithm.

24. Explain Krushal's algorithm with an example?
25. Write the Kruskal's algorithm and construct a minimum spanning tree for a graph. (16) (Nov 09/10)
26. (i) write an algorithm to determine the biconnected components in the given graph. (10) (may 10)

(ii)determine the biconnected components in a graph. (6)

27. Find all articulation points in a example graph. Show the depth first spanning tree and the values of DFN and Low for each vertex. (8) (nov 10)

UNIT – V

1. What is meant by Sorting?

Sorting is ordering of data in an increasing or decreasing fashion according to some linear relationship among the data items.

2. List the different sorting algorithms.

- Bubble sort
- Selection sort
- Insertion sort
- Shell sort
- Quick sort
- Radix sort
- Heap sort
- Merge sort

3. Why bubble sort is called so?

The bubble sort gets its name because as array elements are sorted they gradually "bubble" to their proper positions, like bubbles rising in a glass of soda.

4. State the logic of bubble sort algorithm.

The bubble sort repeatedly compares adjacent elements of an array. The first and second elements are compared and swapped if out of order. Then the second and third elements are compared and swapped if out of order. This sorting process continues until the last two elements of the array are compared and swapped if out of order.

5. **What number is always sorted to the top of the list by each pass of the Bubble sort algorithm?**
   Each pass through the list places the next largest value in its proper place. In essence, each item "bubbles" up to the location where it belongs.

6. **When does the Bubble Sort Algorithm stop?**
   The bubble sort stops when it examines the entire array and finds that no "swaps" are needed. The bubble sort keeps track of the occurring swaps by the use of a flag.

7. **State the logic of selection sort algorithm.**
   It finds the lowest value from the collection and moves it to the left. This is repeated until the complete collection is sorted.

8. **What is the output of selection sort after the 2nd iteration given the following sequence?** 16 3 46 9 28 14
   Ans: 3 9 46 16 28 14

9. **How does insertion sort algorithm work?**
   In every iteration an element is compared with all the elements before it. While comparing if it is found that the element can be inserted at a suitable position, then space is created for it by shifting the other elements one position up and inserts the desired element at the suitable position. This procedure is repeated for all the elements in the list until we get the sorted elements.

10. **What operation does the insertion sort use to move numbers from the unsorted section to the sorted section of the list?**
    The Insertion Sort uses the swap operation since it is ordering numbers within a single list.

11. **How many key comparisons and assignments an insertion sort makes in its worst case?**
    The worst case performance in insertion sort occurs when the elements of the input array are in descending order. In that case, the first pass requires one comparison, the second pass requires two comparisons, third pass three comparisons,….kth pass requires (k-1), and finally the last pass requires (n-1) comparisons. Therefore, total numbers of comparisons are:
    $f(n) = 1+2+3+\ldots\ldots+(n-k)+\ldots..+(n-2)+(n-1) = n(n-1)/2 = O(n2)$

12. **Which sorting algorithm is best if the list is already sorted? Why?**
    Insertion sort as there is no movement of data if the list is already sorted and complexity is of the order O(N).

13. **Which sorting algorithm is easily adaptable to singly linked lists? Why?**
    Insertion sort is easily adaptable to singly linked list. In this method there is an array link of pointers, one for each of the original array elements. Thus the array can be thought of as a linear link list pointed to by an external pointer first initialized to 0. To insert the k^{th} element the linked list is traversed until the proper position for x[k] is found, or until the end of the list is reached. At that point x[k] can be inserted into the

list by merely adjusting the pointers without shifting any elements in the array which reduces insertion time.

14. Why Shell Sort is known diminishing increment sort?

The distance between comparisons decreases as the sorting algorithm runs until the last phase in which adjacent elements are compared. In each step, the sortedness of the sequence is increased, until in the last step it is completely sorted.

15. Which of the following sorting methods would be especially suitable to sort a list L consisting of a sorted list followed by a few "random" elements?

Quick sort is suitable to sort a list L consisting of a sorted list followed by a few "random" elements.

16.Which sorting algorithm follows the divide-and-conquer strategy?

Quick sort and Merge sort

17. What is the output of quick sort after the 3rd iteration given the following sequence? 24 56 47 35 10 90 82 31

Pass 1:- (10) 24 (56 47 35 90 82 31)
Pass 2:- 10 24 (56 47 35 90 82 31)
Pass 3:- 10 24 (47 35 31) 56 (90 82)

18. Mention the different ways to select a pivot element.

The different ways to select a pivot element are

- Pick the first element as pivot
- Pick the last element as pivot
- Pick the Middle element as pivot
- Median-of-three elements
  - Pick three elements, and find the median x of these elements
  - Use that median as the pivot.
- Randomly pick an element as pivot.

19. What is divide-and-conquer strategy?

- Divide a problem into two or more sub problems
- Solve the sub problems recursively
- Obtain solution to original problem by combining these solutions

20. Compare quick sort and merge sort.

Quicksort has a best-case linear performance when the input is sorted, or nearly sorted. It has a worst-case quadratic performance when the input is sorted in reverse, or nearly sorted in reverse.

Merge sort performance is much more constrained and predictable than the performance of quicksort. The price for that reliability is that the average case of merge sort is slower than the average case of quicksort because the constant factor of merge sort is larger.

21. What is the key idea of radix sort?

Sort the keys digit by digit, starting with the least significant digit to the most significant digit.

22.Define Searching.

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

23.Mention the various types of searching techniques in C

✓Linear search

✓Binary search

24.What is linear search?

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

25.What is Binary search?

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

26.What is the need for hashing?

Hashing is used to perform insertions, deletions and find in constant average time.

27. Define hash function?

Hash function takes an identifier and computes the address of that identifier in the hash table using some function.

28.Why do we need a Hash function as a data structure as compared to any other data structure? (may 10)

Hashing is a technique used for performing insertions, deletions, and finds in constant average time.

29.What are the important factors to be considered in designing the hash function? (Nov 10)

- To avoid lot of collision the table size should be prime
- For string data if keys are very long, the hash function will take long to compute.

30.What are the problems in hashing?

a. Collision

b. Overflow

31.What is collision?

When two keys compute in to the same location or address in the hash table through any of the hashing function then it is termed collision.

32.Define collision resolution. What are the methods for avoiding collision? (may 10)

Collision resolution is the process of finding another position for the collide record. Various collision resolution techniques are

- Separate chaining

- Open addressing

33.What are the applications of hash table? (Nov 09)

- Compliers use hash table to keep track of declared variable in source code. (Symbol table)
- Used in graph theory problem where the nodes have real names instead of numbers.
- Used in programs those play game. As the program searches through different lines of play, it keeps track of positions it has seen by computing a hash function based on the position (and storing its move for that position).
- Used in on-line spelling checkers. If misspelling detection is important, an entire dictionary can be prehashed and words can be checked in constant time.

PART B

1. Explain any two techniques to overcome hash collision.
2. Discuss the common collision resolution strategies used in closed hashing system.(16) (Nov 09)
3. (i) given the input { 4371, 1323, 6173, 4199, 4344, 9679, 1989 } and a hash function of h(X)=X (mod 10) show the resulting: (may10)
    a. Separate Chaining hash table (5)
    b. Open addressing hash table using linear probing (5)

   (ii) what are the advantages and disadvantages of various collision resolution strategies? (6)
4. Formulate an ADT to implement separate chaining hashing scheme. (8) (Nov 10)
5. Explain Open addressing.
6. Explain Re-hashing and Extendible hashing.
7. Show the result of inserting the keys 2,3,5,7,11,13,15,6,4 into an initially empty extendible hashing data structure with M=3. (8) (Nov 10)
8. (i) State and explain the dynamic equivalence problem. (6) (May 10)
   (ii) explain the path compression algorithm and analyse the Union/Find algorithm used. (10)
9. Explain the implementation of Disjoint set.
10. Explain smart union algorithm and path compression.
11. What is union-by-height? Write the algorithm to implement it.(8) (Nov 09)
12. (i) Formulate an ADT to perform for the union and find operations of disjoint sets. (8) (Nov 10)
    (ii) Describe about union-by-rank and find with path compression with code. (8)
13. Explain path compression with example. (8) (Nov 09).