1. Define Stack.
   A stack is an ordered list in which all insertions and deletions are made at one end, called the top. It is an abstract data type and based on the principle of LIFO (Last In First Out).

2. What are the operations of the stack?
   a. CreateStack/ InitStack(Stack) – creates an empty stack
   b. Push(Item) – pushes an item on the top of the stack
   c. Pop(Item) – removes the top most element from the stack
   d. Top(Stack) – returns the first element from the stack
   e. IsEmpty(Stack) – returns true if the stack is empty

3. Write the routine to push an element into a stack.
   Push(Element X, Stack S)
   {
     if(IsFull(S))
     {
       Error(“Full Stack”);
     }
     else
     {
       S→Array[++S→TopOfStack]=X;
     }
   }

4. How the operations performed on linked list implementation of stack?
   a. Push and pop operations at the head of the list
   b. New nodes should be inserted at the front of the list, so that they become the top of the stack
   c. Nodes are removed from the front(top) of the stack

5. What are the applications of stack?
   The following are the applications of stacks
   • Evaluating arithmetic expressions
   • Balancing the parenthesis
   • Towers of Hanoi
   • Function calls
   • Tree traversal
6. **What are the methods to implement stack in C?**
   The methods to implement stacks are:
   - Array based
   - Linked list based

7. **How the stack is implemented by linked list?**
   It involves dynamically allocating memory space at run time while performing stack operations.
   Since it consumes only that much amount of space is required for holding its data elements, it prevents wastage of memory space.
   ```c
   struct stack
   {
      int element;
      struct stack *next;
   } *top;
   ```

8. **Write the routine to pop a element from a stack.**
   ```c
   int pop()
   {
      if(top==NULL)
      {
         printf("n Stack is empty.\n");
         getch();
         exit(1);
      }
      else
      {
         int temp;
         temp=top->element; /* retrieving the top element*/
         top=top->next;    /* Updating the stack pointer */
         return temp;      /* returning the popped value */
      }
   }
   ```

9. **Define queue.**
   It is a linear data structure that maintains a list of elements such that insertion happens at rear end and deletion happens at front end.
   FIFO – First In First Out principle
10. What are the operations of a queue?
   The operations of a queue are
   • isEmpty()
   • isFull()
   • insert()
   • delete()
   • display()

11. Write the routine to insert an element onto a queue.
    ```
    void insert(int element)
    {
        if(front==-1 )
            {
                front = rear = front +1;
                queue[front] = element;
                return;
            }
        if(rear==99)
            {
                printf("Queue is full");
                getch();
                return;
            }
        rear = rear +1;
        queue[rear]=element;
    }
    ```

12. What are the types of queue?
    The following are the types of queue:
    • Double ended queue
    • Circular queue
    • Priority queue
### 13. Define double ended queue
- It is a special type of queue that allows insertion and deletion of elements at both ends
- It is also termed as DEQUE.

![Double Ended Queue Diagram](https://via.placeholder.com/150)

### 14. What are the methods to implement queue in C?
The methods to implement queues are:
- Array based
- Linked list based

### 15. How the queue is implemented by linked list?
- It is based on the dynamic memory management techniques which allow allocation and de-allocation of memory space at runtime.

**Insert operation**
- It involves the following subtasks:
  - Reserving memory space of the size of a queue element in memory
  - Storing the added value at the new location
  - Linking the new element with existing queue
  - Updating the `rear` pointer

**Delete operation**
- It involves the following subtasks:
  1. Checking whether queue is empty
  2. Retrieving the front most element of the queue
  3. Updating the front pointer
  4. Returning the retrieved value

### 16. Write the routine to delete a element from a queue
```c
int del()
{
    int i;
    if(front == NULL) /*checking whether the queue is empty*/
    {
        return(-9999);
    }
    else
    {
```

www.studentsfocus.com
i = front→element;
front = front→next;
return i;
}

17. What are the applications of queue?
The following are the areas in which queues are applicable
a. Simulation
b. Batch processing in an operating systems
c. Multiprogramming platform systems
d. Queuing theory
e. Printer server routines
f. Scheduling algorithms like disk scheduling, CPU scheduling
g. I/O buffer requests

18. Define circular queue
A Circular queue is a queue whose start and end locations are logically connected with each other. That means the start location comes after the end location.

![Circular Queue Diagram]

19. What are push and pop operations?
• Push – adding an element to the top of stack
• Pop – removing or deleting an element from the top of stack

20. What are enqueue and dequeue operations?
• Enqueue - adding an element to the queue at the rear end
• Dequeue – removing or deleting an element from the queue at the front end
Part – B

1. Explain Stack ADT and its operations
2. Explain array based implementation of stacks
3. Explain linked list implementation of stacks
4. Explain the applications of Stacks
5. Explain how to evaluate arithmetic expressions using stacks
6. Explain queue ADT
7. Explain array based implementation of queues
8. Explain linked list implementation of queues
9. Explain the applications of queues
10. Explain circular queue and its implementation
11. Explain double ended queue and its operations