Q:-1. Which data structure is used for implementing recursion?
a) Stack
b) Queue
c) List
d) Array
Answer: a
Explanation: Stacks are used for the implementation of Recursion.

Q:-2. The data structure required to check whether an expression contains a balanced parenthesis is?
a) Queue
b) Stack
c) Tree
d) Array
Answer: b
Explanation: The stack is a simple data structure in which elements are added and removed based on the LIFO principle. Open parenthesis is pushed into the stack and a closed parenthesis pops out elements till the top element of the stack is its corresponding open parenthesis. If the stack is empty, parenthesis is balanced otherwise it is unbalanced.

Q:-3. Which of the following is not the application of stack?
a) Data Transfer between two asynchronous process
b) Compiler Syntax Analyzer
c) Tracking of local variables at run time
d) A parentheses balancing program
Answer: a
Explanation: Data transfer between the two asynchronous process uses the queue data structure for synchronisation. The rest are all stack applications.

Q:-4. Which data structure is needed to convert infix notation to postfix notation?
a) Tree
b) Branch
c) Stack
d) Queue
Answer: c
Explanation: The Stack data structure is used to convert infix expression to postfix expression. The purpose of stack is to reverse the order of the operators in the expression. It also serves as a storage structure, as no operator can be printed until both of its operands have appeared.

Q:-5. What is the value of the postfix expression 6 3 2 4 + – *?
a) 74
b) -18

c) 22
d) 40
Answer: b
Explanation: Postfix Expression is (6*(3-(2+4))) which results -18 as output

Q:-6. Which of the following statement(s) about stack data structure is/are NOT correct?
a) Top of the Stack always contain the new node
b) Stack is the FIFO data structure
c) Null link is present in the last node at the bottom of the stack
d) Linked List are used for implementing Stacks
Answer: b
Explanation: Stack follows LIFO.

Q:-7. The optimal data structure used to solve Tower of Hanoi is _________
a) Tree
b) Heap
c) Priority queue
d) Stack
View Answer
Answer: d
Explanation: The Tower of Hanoi involves moving of disks 'stacked' at one peg to another peg with respect to the size constraint. It is conveniently done using stacks and priority queues. Stack approach is widely used to solve Tower of Hanoi.

Q:-8. Which of the following application makes use of a circular linked list?
a) Recursive function calls
b) Undo operation in a text editor
c) Implement Hash Tables
d) Allocating CPU to resources

Answer: d
Explanation: Generally, round robin fashion is employed to allocate CPU time to resources which makes use of the circular linked list data structure. Recursive function calls use stack data structure. Undo Operation in text editor uses doubly linked lists. Hash tables uses singly linked lists.

Q:-9. Which of the following is not the type of queue?
a) Priority queue
b) Circular queue
c) Single ended queue
d) Ordinary queue

Answer: c
Explanation: Queue always has two ends. So, single ended queue is not the type of queue.

Q:-10. What is the need for a circular queue?
a) easier computations
b) implement LIFO principle in queues
c) effective usage of memory
d) to delete elements based on priority
Answer: c
Explanation: In a linear queue, dequeue operation causes the starting elements of the array to be empty, and there is no way you can use that space, while in a circular queue, you can effectively use that space. Priority queue is used to delete the elements based on their priority.

Q:-11. What is a dequeue?
a) A queue implemented with both singly and doubly linked lists
b) A queue with insert/delete defined for front side of the queue
c) A queue with insert/delete defined for both front and rear ends of the queue
d) A queue implemented with a doubly linked list
View Answer
Answer: c
Explanation: A dequeue or a double ended queue is a queue with insert/delete defined for both front and rear ends of the queue.

Q:-12. A data structure in which elements can be inserted or deleted at/from both ends but not in the middle is?
a) Priority queue
b) Dequeue
c) Circular queue
d) Queue
View Answer
Answer: b
Explanation: In dequeuer, we can insert or delete elements from both the ends. In queue, we will follow first in first out principle for insertion and deletion of elements. Element with least priority will be deleted in a priority queue.

13. The type of expression in which operator succeeds its operands is?
a) Infix Expression
b) Prefix Expression
c) Postfix Expression
d) Both Prefix and Postfix Expressions

Answer: c
Explanation: The expression in which operator succeeds its operands is called postfix expression. The expression in which operator precedes the operands is called prefix expression. If an operator is present between two operands, then it is called infix expressions.

14. If the elements "A", "B", "C" and "D" are placed in a stack and are deleted one at a time, what is the order of removal?
a) ABCD
b) DCBA
c) DCAB
d) ABDC

Answer: b
Explanation: Stack follows LIFO(Last In First Out). So the removal order of elements are DCBA.

15. Reversing a word using stack can be used to find if the given word is a palindrome or not.
a) True
b) False

Answer: a
Explanation: This application of stack can also be used to find if the given word is a palindrome because, if the reversed is same as that of the original word, the given word is a palindrome.

16. Which is the most appropriate data structure for reversing a word?
a) queue
b) stack
c) tree
d) graph

Answer: b
Explanation: Stack is the most appropriate data structure for reversing a word because stack follows LIFO principle.
51. Operations required for reversing a word or a string using stack are push() and pop().
a) True
b) False

Answer: a
Explanation: Push operation inserts a character into the stack and pop operation pops the top of the stack.
17. What is the time complexity of reversing a word using stack algorithm?
a) O (N log N)

b) O (N^2)
c) O (N)
d) O (M log N)

Answer: c
Explanation: The time complexity of reversing a stack is mathematically found to be O (N) where N is the input.

18. What is the best case time complexity of deleting a node in a Singly Linked list?
a) O (n)
b) O (n^2)
c) O (nlogn)
d) O (1)

Answer: d
Explanation: Deletion of the head node in the linked list is taken as the best case. The successor of the head node is changed to head and deletes the predecessor of the newly assigned head node. This process completes in O(1) time.

19. Which of the following statements are not correct with respect to Singly Linked List(SLL) and Doubly Linked List(DLL)?
a) Complexity of Insertion and Deletion at known position is O(n) in SLL and O(1) in DLL
b) SLL uses lesser memory per node than DLL
c) DLL has more searching power than SLL
d) Number of node fields in SLL is more than DLL

Answer: d
Explanation: To insert and delete at known positions requires complete traversal of the list in worst case in SLL, SLL consists of an item and a node field, while DLL has an item and two node fields, hence SLL occupies lesser memory, DLL can be traversed both ways(left and right), while SLL can traverse in only one direction, hence more searching power of DLL. Node fields in SLL is 2 (data and address of next node) whereas in DLL is 3(data, address to next node, address to previous node).

20. What does 'stack overflow' refer to?
a) accessing item from an undefined stack
b) adding items to a full stack
c) removing items from an empty stack
d) index out of bounds exception
Answer: b
Explanation: Adding items to a full stack is termed as stack underflow.

21. Which of the following data structures can be used for parentheses matching?
a) n-ary tree
b) queue
c) priority queue
d) stack
Answer: d
Explanation: For every opening brace, push it into the stack, and for every closing brace, pop it off the stack. Do not take action for any other character. In the end, if the stack is empty, then the input has balanced parentheses.
22. Minimum number of queues to implement stack is ___________
a) 3
b) 4
c) 1
d) 2
Answer: c
Explanation: Use one queue and one counter to count the number of elements in the queue.

23. Which of the following is not an inherent application of stack?
a) Reversing a string
b) Evaluation of postfix expression
c) Implementation of recursion
d) Job scheduling

Answer: d
Explanation: Job Scheduling is not performed using stacks.

24. What will be the word obtained if the word “abbcabb” is reversed using a stack?
a) bbabbca
b) abbcabb
c) bbacbba
d) bbacabb

Answer: c
Explanation: The string “abbcabb” is pushed on to the stack. If the characters are popped one by one, the word obtained will be bbacbba.

25. What is the key advantage of using a circular queue over a linear queue?

a) Easy implementation
b) Efficient use of memory
c) Faster operations
d) Decreased time complexity

Answer: b) Efficient use of memory
Explanation: Circular queues reuse the freed-up space unlike linear queues.

26. In a circular queue implementation, what condition indicates that the queue is full?

a) front == rear
b) front == -1
c) (rear + 1) % size == front
d) rear == size - 1

Answer: c) (rear + 1) % size == front
Explanation: This condition checks if the next rear position is the current front.

27. Which of the following correctly initializes a circular queue in Java using arrays?

a)

```
int front = -1, rear = -1;

int[] queue = new int[size];
```

b)

```
Queue queue = new LinkedList<>();
```

c)

```
Stack<Integer> queue = new Stack<>();
```

d)

```
int front = 0, rear = size - 1;
```

Answer: a)
Explanation: Initialization with front and rear at -1 is standard for array-based circular queues.

28. What will be the output of the following code?

```
int size = 5;

int[] queue = new int[size];
int front = -1, rear = -1;

rear = (rear + 1) % size;
queue[rear] = 10;
front = 0;

System.out.println(queue[rear]);
```

a) 0
b) 10
c) 1
d) Compile error

Answer: b) 10
Explanation: 10 is inserted and printed successfully.

29. What happens when trying to dequeue from an empty circular queue?

a) Rear is incremented
b) Front is incremented
c) Underflow occurs
d) Elements shift left

Answer: c) Underflow occurs
Explanation: Dequeuing from an empty queue triggers underflow.

30. How is the front element accessed in a circular queue?

a) queue[rear]
b) queue[0]
c) queue[front]
d) queue[size - 1]

Answer: c) queue[front]
Explanation: The front pointer points to the element to be dequeued next.

31. In the below code, what condition checks if the queue is empty?

```
if (front == -1)
```

a) The queue is full
b) The queue is empty
c) The queue is overflowing
d) The front is in middle

Answer: b) The queue is empty
Explanation: Initialization to -1 indicates the queue has no elements.

32. What is the time complexity of enqueue and dequeue operations in a circular queue (array-based)?

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

Answer: b) O(1)
Explanation: Enqueue and dequeue are done in constant time.

33. Which of the following statements is true about a circular queue?

a) It cannot wrap around the array
b) Front always moves forward, never resets
c) Queue becomes full when (rear + 1) % size == front
d) Rear equals front when empty

Answer: c)
Explanation: That's the standard condition for fullness in a circular queue.

34. What will be the final values of front and rear after the following operations on a size 3 circular queue?

Enqueue(1)

Enqueue(2)

Dequeue()

Enqueue(3)

Enqueue(4)

a) front = 0, rear = 2
b) front = 1, rear = 0
c) front = 2, rear = 1
d) front = 1, rear = 2

b) front = 1, rear = 0
Explanation: After wrap-around insertion, rear points to 0 and front to next valid value.

35. Which of these correctly returns the front element of the queue without removing it?

a) queue[rear]
b) queue[front++]
c) queue[front]
d) queue[++rear]

Answer: c) queue[front]
Explanation: Accessing the front element without incrementing pointer returns the value without removing it.

36. What is printed by the following code?

```
int[] queue = new int[5];

int front = 0, rear = -1;

rear++;

queue[rear] = 10;

rear++;

queue[rear] = 20;
```

37. What will happen when you dequeue from an empty array-based queue?

a) Returns -1
b) Returns 0
c) Underflow condition
d) Overflow condition

Answer: c) Underflow condition
Explanation: Dequeuing from an empty queue is not allowed and causes underflow.

System.out.println(queue[front]);

a) 10
b) 20
c) 0
d) 1

Answer: a) 10
Explanation: front points to the first element, which is 10.

38. What will happen if dequeue is called on an empty linked list-based queue?

a) Returns 0
b) Causes underflow
c) Deletes rear
d) Adds a null value

Answer: b) Causes underflow
Explanation: Dequeue on an empty queue is invalid and causes an underflow condition.

39. Which node in the linked list-based queue is removed during a dequeue operation?

a) Tail node
b) Head node
c) Middle node
d) Last inserted node

Answer: b) Head node
Explanation: Queue uses FIFO, so the first (head) node is removed.

40. In a queue using linked list, which end is used for insertion?

a) Front
b) Rear
c) Middle
d) Random

Answer: b) Rear
Explanation: In queues, insertion happens at the rear and deletion at the front.