Stack and Queue Data Structure
Assignment -#2

Q1. Which element would get first deleted from the stack?
a) The element in the middle of the stack
b) The element at the top of the stack
c) The element at the bottom of the stack
d) More than one of the above
e) None of the above

Q2. A stack is a linear data structure in which data is stored and retrieved in a:
a) Last Out First In (LOFI)
b) Last In First Out (LIFO)
c) First In First Out (FIFO)
d) More than one of the above
e) None of the above

Q3. The five items P, Q, R, S, and T are pushed in a stack, one after the other starting from P. The stack is popped four times and each element is inserted in a queue. Then, 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) P
b) R
c) Q
d) More than one of the above
e) None of the above

Q4. A stack can be implemented using a queue, but then we need to use at least:
a) 3 queues
b) 2 queues
c) Only one queue is sufficient
d) More than one of the above
e) None of the above

Q5. A recursive problem like Tower of Hanoi can be rewritten without recursion using:
a) Stack
b) Priority queue
c) Graph
d) More than one of the above
e) None of the above

Q6. Suppose a stack is implemented with a linked list instead of an array. What would be the effect on the time complexity of the push and pop operations of the stack (assuming efficient implementation)?
a) O(1) for insertion and O(n) for deletion
b) O(1) for insertion and O(1) for deletion
c) O(n) for insertion and O(1) for deletion
d) More than one of the above
e) None of the above

Q7. Which of the following permutations can be obtained in the same order using a stack assuming the input sequence is 5, 6, 7, 8, 9 in that order?
a) 7, 8, 9, 5, 6
b) 5, 9, 6, 7, 8
c) 7, 8, 9, 6, 5
d) More than one of the above
e) None of the above

Q8. The best data structure to check whether an arithmetic expression has balanced parentheses is a:
a) Queue
b) Stack
c) Tree
d) More than one of the above
e) None of the above

Q9. 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. Then, 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) More than one of the above
e) None of the above

Q10. Which of the following statement(s) about the stack data structure is/are NOT correct?
a) Stack data structure can be implemented using a linked list
b) A new node can only be added at the top of the stack
c) Stack is the FIFO data structure
d) The last node at the bottom of the stack has a NULL link
e) None of the above

Q11. The seven elements A, B, C, D, E, F, and G are pushed onto a stack in reverse order, i.e., starting from G. The stack is popped five times and each element is inserted into a queue. Two elements are deleted from the queue and pushed back onto the stack. Now, one element is popped from the stack. The popped item is:

a) A
b) B
c) F
d) More than one of the above
e) None of the above

Q12. If the sequence of operations - push(1), push(2), pop, push(1), push(2), pop, pop, pop, push(2), pop are performed on a stack, the sequence of popped-out values is:

a) 2, 2, 1, 1, 2
b) 2, 2, 1, 2, 2
c) 2, 1, 2, 2, 1
d) More than one of the above
e) None of the above

Q13. Consider the following operations performed on a stack of size 5:
Push(a); Pop(); Push(b); Push(c); Pop(); Push(d); Pop(); Pop(); Push(e)
Which of the following statements is correct?

a) Underflow occurs
b) Stack operations are performed smoothly
c) Overflow occurs
d) More than one of the above
e) None of the above

Q14. Which of the following is NOT an inherent application of stack?

a) Implementation of recursion
b) Evaluation of a postfix expression
c) Job scheduling
d) More than one of the above
e) None of the above

Q15. Which of the following is true about the linked list implementation of stack?

a) In push operation, if new nodes are inserted at the beginning of the linked list, then in pop operation, nodes must be removed from the end.
b) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from the beginning.
c) More than one of the above
d) None of the above

Q16. Which one of the following is an application of the Stack Data Structure?

a) Managing function calls
b) The stock span problem
c) Arithmetic expression evaluation
d) More than one of the above
e) None of the above

Q17. Consider the following operations performed on a stack of size 5:
Push(1); Pop(); Push(2); Push(3); Pop(); Push(4); Pop(); Pop(); Push(5);
After the completion of all operations, the number of elements present on the stack is:

a) 1
b) 2
c) 3
d) More than one of the above
e) None of the above

Q18. Which of the following applications generally use a stack?

a) Parenthesis balancing program
b) Syntax analyzer in a compiler
c) Keeping track of local variables at runtime
d) More than one of the above
e) None of the above

Q19. What happens when you try to pop an element from an empty Stack?

a) It returns null
b) It causes an underflow
c) It causes an overflow
d) More than one of the above
e) None of the above

Q20 What is the time complexity of a push operation in a Stack?

a) O(n)
b) O(n log n)
c) O(1)
d) More than one of the above
e) None of the above

Q21. What is the time complexity to get the minimum element from the Stack?

a) O(1)
b) O(n)
c) O(n log n)
d) More than one of the above
e) None of the above

Q22. 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) More than one of the above
e) None of the above

Q23. The data structure required for Breadth First Traversal on a graph is?
a) Stack
b) Array
c) Queue
d) More than one of the above
e) None of the above

Q24. A queue follows _____.
a) FIFO (First In First Out) principle
b) LIFO (Last In First Out) principle
c) Ordered array
d) More than one of the above
e) None of the above

Q25. Circular Queue is also known as _____.
a) Ring Buffer
b) Square Buffer
c) Rectangle Buffer
d) More than one of the above
e) None of the above

Q26. If the elements "A", "B", "C", and "D" are placed in a queue and are deleted one at a time, in what order will they be removed?
a) ABCD
b) DCBA
c) DCAB
d) More than one of the above
e) None of the above

Q27. A data structure in which elements can be inserted or deleted at/from both ends but not in the middle is?
a) Queue
b) Circular queue
c) Deque
d) More than one of the above
e) None of the above

Q28. A normal queue, if implemented using an array of size MAX_SIZE, gets full when?
a) Rear = MAX_SIZE – 1
b) Front = (rear + 1) mod MAX_SIZE
c) Front = rear + 1
d) More than one of the above
e) None of the above

Q29. Queues serve a major role in _____.
a) Simulation of recursion
b) Simulation of arbitrary linked list
c) Simulation of limited resource allocation
d) More than one of the above
e) None of the above

Q30. Which of the following is not a type of queue?
a) Ordinary queue
b) Single-ended queue
c) Circular queue
d) More than one of the above
e) None of the above

Q31. In linked list implementation of a queue, if only the front pointer is maintained, which of the following operations takes worst-case linear time?
a) Insertion
b) Deletion
c) To empty a queue
d) More than one of the above
e) None of the above

Q32. In linked list implementation of a queue, where is a new element inserted?
a) At the head of the linked list
b) At the center position in the linked list
c) At the tail of the linked list
d) More than one of the above
e) None of the above

Q33. In linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into a non-empty queue?
a) Only the front pointer
b) Only the rear pointer
c) Both front and rear pointers
d) More than one of the above
e) None of the above

Q34. In linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into an empty queue?
a) Only the front pointer
b) Only the rear pointer
c) Both front and rear pointers
d) More than one of the above
e) None of the above

Q35. In linked list implementation of a queue, from where is the item deleted?
a) At the head of the linked list
b) At the center position in the linked list
c) At the tail of the linked list
d) More than one of the above
e) None of the above

Q36. In linked list implementation of a queue, the important condition for a queue to be empty is?
a) FRONT is null
b) REAR is null
c) LINK is empty
d) More than one of the above
e) None of the above

Q37. The essential condition that is checked before insertion in a linked queue is?
a) Underflow
b) Overflow
c) Front value
d) More than one of the above
e) None of the above

Q38. The essential condition that is checked before deletion in a linked queue is?
a) Underflow
b) Overflow
c) Front value
d) More than one of the above
e) None of the above

Q39. Which of the following is true about linked list implementation of a queue?
a) In enqueue operation, if new nodes are inserted at the beginning of the linked list, then in dequeue operation, nodes must be removed from the end
b) In enqueue operation, if new nodes are inserted at the beginning, then in dequeue operation, nodes must be removed from the end
c) In enqueue operation, if new nodes are inserted at the end, then in dequeue operation, nodes must be removed from the end
d) More than one of the above
e) None of the above

Q40. Which of the following is not an advantage of a priority queue?
a) Easy to implement
b) Processes with different priorities can be efficiently handled
c) Easy to delete elements in any case
d) More than one of the above
e) None of the above

Q41. What is the time complexity to insert a node based on position in a priority queue?
a) O(nlogn)
b) O(logn)
c) O(n)
d) More than one of the above
e) None of the above

Q42. With what data structure can a priority queue be implemented?
a) Array
b) List
c) Heap
d) More than one of the above
e) None of the above

Q43. Which of the following is not an application of a priority queue?
a) Huffman codes
b) Interrupt handling in an operating system
c) Undo operation in text editors
d) More than one of the above
e) None of the above

Q44. What is the need for a circular queue?
a) Effective usage of memory
b) Easier computations
c) To delete elements based on priority
d) More than one of the above
e) None of the above

Q45. What is the time complexity of an enqueue operation in a queue?
a) O(logn)
b) O(nlogn)
c) O(n)
d) More than one of the above
e) None of the above

Q46. In a circular queue, how do you increment the rear end of the queue?
a) rear++
b) (rear + 1) % CAPACITY
c) (rear % CAPACITY) + 1
d) More than one of the above
e) None of the above

Q47. What is the term for inserting into a full queue?
a) Overflow
b) Underflow
c) Null pointer exception
d) More than one of the above
e) None of the above

Q48. Given a queue with a linked list implementation where the rear pointer points to the rear node of the queue and the front node points to the front node, which of the following operations is impossible to do in O(1) time?
a) Delete the front item from the list
b) Delete the rear item from the list
c) Insert at the front of the list
d) More than one of the above
e) None of the above

Q49. Which of the following operations on a queue data structure has a time complexity of O(1)?
a) Enqueue
b) Dequeue
c) Peek
d) More than one of the above
e) None of the above

Q50. Suppose a circular queue of capacity n−1 elements is implemented with an array of n elements. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are:
a) Full: (REAR+1)mod n==FRONT
empty: REAR==FRONT
b) Full: (REAR+1)mod n==FRONT
empty: (FRONT+1)mod n==REAR
c) Full: REAR==FRONT,
empty: (REAR+1)mod n==FRONT
d) More than one of the above
e) None of the above

Q51. The minimum number of stacks needed to implement a queue is:
a) 3
b) 1
c) 2
d) More than one of the above
e) None of the above

Q52. Which data structure is commonly used to implement the event-driven simulation of complex systems, such as in computer network simulations or traffic simulations?
a) Stack
b) Tree
c) Array
d) More than one of the above
e) None of the above

Q53. Which of the following is/are advantages of a circular queue?
a) Memory Management
b) Traffic system
c) CPU Scheduling
d) More than one of the above
e) None of the above

Q54. A circular queue is implemented using an array of size 10. The array index starts with 0, front is 6, and rear is 9. The insertion of the next element takes place at the array index:
a) 0
b) 7
c) 9
d) More than one of the above
e) None of the above

Q55. A normal queue, if implemented using an array of size MAX_SIZE, gets full when:

a) **Rear = MAX_SIZE - 1**
b) **Front = (rear + 1) mod MAX_SIZE**
c) **Front = rear + 1**
d) **More than one of the above**
e) **None of the above**

Solution with Explanation

Answer1: b. The element at the top of the stack
Explanation: A stack follows a LIFO (Last In First Out) structure. The element at the top is the most recent addition, so it is the first one to be deleted.

Answer2: b. Last In First Out (LIFO)
Explanation: In a stack, the element added last is the first to be retrieved, making it a LIFO structure.

Answer3: e) None of the above (S)
Explanation: After the stack is popped four times, the order of elements in the queue is T, S, R, Q (T at the front, Q at the rear). After two elements (T, S) are deleted from the queue and pushed back onto the stack, the stack order becomes P, S, with S at the top. The next pop operation removes S.

Answer4: b. b queues
Explanation: A stack can be implemented using two queues to efficiently manage push and pop operations while maintaining the LIFO property.

Answer5: a. Stack
Explanation: Recursive problems can be converted to iterative solutions using stacks. The Tower of Hanoi can be implemented using a stack because of its LIFO nature, which matches the recursive call stack behavior.

Answer6: b. O(1) for insertion and O(1) for deletion
Explanation: When a stack is implemented using a linked list, both push and pop operations can be performed in O(1) time because insertion and deletion occur at the head of the list.

Answer7: c. 7, 8, 9, 6, 5
Explanation: The sequence 7, 8, 9, 6, 5 can be obtained using stack operations (pushing 5, 6, 7, 8, 9 and popping them in the desired order).

Answer8: b. Stack
Explanation: A stack efficiently manages balanced parentheses by pushing opening parentheses onto the stack and popping when matching closing parentheses are found.

Answer9: . E) None of the above
Explanation: After pushing and popping the elements, the stack contains A, D. Since D is at the top, it will be the first element to be popped.

Answer17: a. 1

Answer10: c. Stack is the FIFO data structure
Explanation: A stack is a LIFO (Last In First Out) data structure, not FIFO.

Answer11: b. B
Explanation: Initially, the stack looks like this (bottom to top): G, F, E, D, C, B, A. After five pops, the elements A, B, C, D, E are inserted into a queue. The two front elements, A and B, are removed from the queue and pushed back onto the stack. The stack now contains (from bottom to top): G, F,A ,B. When one pop is performed, B is removed.

Answer12: a. 2, 2, 1, 1, 2
Explanation:
push(1): Stack = [1]
push(2): Stack = [1, 2]
pop(): Stack = [1], Output = 2
push(1): Stack = [1, 1]
push(2): Stack = [1, 1, 2]
pop(): Stack = [1, 1], Output = 2
pop(): Stack = [1], Output = 1
pop(): Stack = [], Output = 1
push(2): Stack = [2]
pop(): Stack = [], Output = 2
Thus, the sequence of popped-out values is 2, 2, 1, 1, 2.

Answer13: b. Stack operations are performed smoothly
Explanation: The operations on the stack are within its size limit, and there is no underflow or overflow. All operations are performed smoothly.

Answer14: c. Job scheduling
Explanation: Stack is used for recursion, postfix evaluation, and string reversal, but job scheduling typically uses different structures like priority queues or other scheduling algorithms.

Answer15: c. Both of the above (a and b)
Explanation: A stack can be implemented using a linked list in two ways:
Insert at the beginning and remove from the beginning (efficient).
Insert at the end and remove from the end (also maintains LIFO order).

Answer16: d. More than one of the above (a,b,c)
Explanation: Stack is used for managing function calls (as a call stack), arithmetic expression evaluation (infix to postfix conversion, etc.), and the stock span problem.

Answer26: a. ABCD

Explanation: push(1): Stack = [1]
pop(): Stack = []
push(2): Stack = [2]
push(3): Stack = [2, 3]
pop(): Stack = [2]
push(4): Stack = [2, 4]
pop(): Stack = [2]
pop(): Stack = []
push(5): Stack = [5]
Thus, after all operations, only one element (5) remains on the stack.

Answer18: d. More than one of the above
Explanation: Stack is used for multiple purposes, including parenthesis balancing, syntax analysis, and tracking local variables during function calls.

Answer19: C) It causes an underflow
Explanation: Attempting to pop an element from an empty stack will cause an underflow.

Answer20: C) O(1)
Explanation: The time complexity of a push operation in a stack is O(1), which means it performs the operation in constant time.

Answer21: B) O(n)
Explanation: If the stack does not maintain a separate data structure to track the minimum element, it requires O(n) time complexity to get the minimum as we need to traverse all elements of the stack.

Answer22: a. Queue
Explanation: A queue allows deletion from the front and insertion at the rear, following a First In First Out (FIFO) principle. In a stack, deletion occurs from the top (Last In First Out or LIFO).

Answer23: c. Queue
Explanation: Breadth-First Search (BFS) traversal requires a queue to explore all vertices at the current level before moving to the next, following the FIFO principle.

Answer24: a. FIFO (First In First Out) principle
Explanation: A queue operates on a FIFO principle, where the first element added is the first one to be removed.

Answer25: a. Ring Buffer
Explanation: A circular queue is called a ring buffer because the last position is connected back to the first position to form a circle.

Explanation: A queue follows the FIFO approach, so the elements are removed in the order they were inserted: A, B, C, D.

Answer27: c. Dequeue
Explanation: A deque (double-ended queue) allows insertions and deletions at both ends, unlike a regular queue which operates on a FIFO basis.

Answer28: a. Rear = MAX_SIZE – 1
Explanation: A queue implemented using an array is full when the rear reaches the last index (MAX_SIZE - 1), meaning there is no space left for new elements.

Answer29: c. Simulation of limited resource allocation
Explanation: Queues are used to simulate resource allocation since resources are allocated based on the order of requests (FIFO). Recursion uses stacks, and linked lists are used to simulate arbitrary linked lists.

Answer30: b. Single-ended queue
Explanation: Queues always have two ends: one for insertion (rear) and one for deletion (front). There is no such thing as a single-ended queue.

Answer31: d. More than one of the above (a and c)
Explanation: Since only the front pointer is maintained, both insertion and emptying the queue will take linear time. This is because inserting at the rear without a rear pointer requires traversing the entire list.

Answer32: c. At the tail of the linked list
Explanation: In a queue, new elements are inserted at the rear (tail) to maintain the First In First Out (FIFO) property.

Answer33: b. Only the rear pointer
Explanation: In a non-empty queue, only the rear pointer changes during insertion, as new elements are added at the tail of the list.

Answer34: c. Both front and rear pointers
Explanation: When inserting into an empty queue, both front and rear pointers are updated to point to the new element.

Answer35: a. At the head of the linked list
Explanation: In a queue, elements are deleted from the front (head), following the FIFO principle.

Answer36: a. FRONT is null
Explanation: When the front pointer is null, the queue is empty, indicating no elements are present for deletion.

Answer37: b. Overflow
Explanation: Before insertion, the condition of overflow is checked to ensure there's enough memory space to accommodate new elements.

Answer38: a. Underflow
Explanation: Before deletion, underflow is checked to determine if there are elements in the queue to remove. If the queue is empty, underflow occurs.

Answer39: d. More than one of the above (a and b)
Explanation: A queue can be implemented by inserting elements at either the beginning or the end, depending on the implementation. However, to maintain the FIFO order, nodes must be inserted at the rear and removed from the front.

Answer40: c. Easy to delete elements in any case
Explanation: Although priority queues handle processes efficiently, deletion can be time-consuming in the worst case. Therefore, "easy to delete elements" is not an advantage.

Answer41: c. O(n)
Explanation: Inserting a node in the correct position in a priority queue might require traversing the entire queue, leading to a time complexity of O(n). Also Inserting a node based on the key may require traversing the entire list, so time complexity is O(n)

Answer42: d. More than one of the above
Explanation: A priority queue can be implemented using an array, a list, a binary search tree, or a heap. However, the most efficient implementation is with a heap.

Answer43: c. Undo operation in text editors
Explanation: Undo operations are typically implemented using a stack, while priority queues are used for tasks like Huffman coding and interrupt handling.

Answer44: a. Effective usage of memory
Explanation: A circular queue allows the re-use of space that becomes available when elements are dequeued, unlike a linear queue where memory at the front of the queue becomes wasted.

Answer45: e. None of the above O(1)
Explanation: The time complexity of enqueue in a queue is O(1), as it typically involves inserting an element at the rear of the queue.

Answer46: b. (rear + 1) % CAPACITY
Explanation: This formula ensures that the rear pointer wraps around and stays within the bounds of the queue, preventing overflow.

Answe47: a. Overflow
Explanation: Just like a stack, attempting to insert into a full queue results in an overflow condition.

Answer48: d. More than one of the above(b and c)
Explanation: Deleting the rear item and inserting at the front both take O(n) time in a singly linked list queue. Deleting the front item takes O(1).

Answer49: d. More than one of the above (a and b)
Explanation: Both enqueue and dequeue operations in a queue take O(1) time, as they involve adding/removing an element at the front or rear of the queue.

Answer50: a. Full: (REAR+1)mod n==FRONT
empty: REAR==FRONT
Explanation: In a circular queue, the queue is full when the next position of REAR equals the FRONT, and it is empty when both REAR and FRONT are equal.

Answer51: c. 2
Explanation: Two stacks are required to implement a queue using stack operations, where one stack is used for enqueue and the other for dequeue operations.

Answer52: e. None of the above (Queue)
Explanation: A queue is typically used in event-driven simulations to process events in the order they occur (FIFO principle), but none of the provided options is correct.

Answer53: d. More than one of the above (all)
Explanation: Circular queues are useful in memory management, traffic systems, and CPU scheduling because they optimize the use of memory and help in managing processes in a cyclic order.

Answer54: a. 0
Explanation: In a circular queue, after the rear reaches the last position, the next insertion takes place at the beginning of the array (index 0).

Answer55: a. Rear = MAX_SIZE - 1
Explanation: In a normal queue (non-circular), the queue is full when the rear reaches the last position in the array.