1. What is a circular queue?
   1. A queue that allows insertion only at the front
   2. A queue that allows insertion only at the rear
   3. **A queue in which the last element points to the first element**
   4. A queue in which the first element points to the last element
2. Which of the following is not an advantage of using a circular queue?
   1. Efficient memory utilization
   2. Fast insertion and deletion operations
   3. **Support for dynamic resizing**
   4. Ease of implementation
3. In a circular queue, if the front and rear pointers meet, it means that the queue is:
   1. Full
   2. **Empty**
   3. Overflowed
   4. Underflowed
4. How many elements can a circular queue of size N hold?
   1. **N-1**
   2. N
   3. N+1
   4. 2N
5. Suppose a circular queue of capacity (n – 1) elements is implemented with an array of n elements. Assume that the insertion and deletion operation are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are
   1. **Full: (REAR+1) mod n == FRONT, empty: REAR == FRONT**
   2. Full: (REAR+1) mod n == FRONT, empty: (FRONT+1) mod n == REAR
   3. Full: REAR == FRONT, empty: (REAR+1) mod n == FRONT
   4. Full: (FRONT+1) mod n == REAR, empty: REAR == FRONT
6. Which operation is used to insert an element into a circular queue?
   1. push
   2. pop
   3. **enqueue**
   4. dequeue
7. Which operation is used to remove an element from a circular queue?
   1. push
   2. pop
   3. enqueue
   4. **dequeue**
8. The time complexity of enqueue and dequeue operations in a circular queue is:
   1. **O(1)**
   2. O(n)
   3. O(log n)
   4. O(n^2)
9. Which of the following is not required to implement a circular queue?
   1. Array to store the elements
   2. Front pointer
   3. Rear pointer
   4. **Auxiliary pointer**
10. How can you check if a circular queue is empty?
    1. By checking if the front and rear pointers are NULL
    2. **By checking if the front and rear pointers are equal**
    3. By checking if the front and rear pointers are -1
    4. By checking if the front and rear pointers are 0
11. How can you check if a circular queue is full?
    1. By checking if the front and rear pointers are NULL
    2. **By checking if the front and rear pointers are equal**
    3. By checking if the front and rear pointers are -1
    4. By checking if the front and rear pointers are N-1
12. Which data structure is suitable for implementing a circular queue?
    1. **Array**
    2. Linked list
    3. Stack
    4. Tree
13. Which of the following statements is true for a circular queue?
    1. It follows the Last In, First Out (LIFO) principle.
    2. **It follows the First In, First Out (FIFO) principle.**
    3. It follows the Last In, Last Out (LILO) principle.
    4. It follows the First In, Last Out (FILO) principle.
14. The front and rear pointers of a circular queue are initially set to:
    1. 0
    2. **-1**
    3. N
    4. NULL
15. Which operation is used to find the number of elements in a circular queue?
    1. size
    2. **length**
    3. count
    4. capacity
16. Which of the following statements is true for a circular queue with a capacity of N?
    1. It can hold N elements.
    2. **It can hold N-1 elements.**
    3. It can hold N+1 elements.
    4. It can hold 2N elements.
17. The front pointer of a circular queue is incremented by:
    1. **1**
    2. -1
    3. N
    4. Rear pointer
18. The rear pointer of a circular queue is incremented by:
    1. **1**
    2. -1
    3. N
    4. Front pointer
19. Which of the following is an advantage of using a circular queue over a linear queue?
    1. **Better memory utilization**
    2. Faster insertion and deletion operations
    3. Simplicity of implementation
    4. Support for dynamic resizing
20. What happens when you try to insert an element into a full circular queue?
    1. The element is inserted at the front of the queue.
    2. The element is inserted at the rear of the queue.
    3. The element is discarded.
    4. **An error or exception is raised.**
21. What happens when you try to remove an element from an empty circular queue?
    1. The element is removed from the front of the queue.
    2. The element is removed from the rear of the queue.
    3. Nothing happens.
    4. **An error or exception is raised.**
22. Which of the following operations can be performed on a circular queue?
    1. Insertion at the front and removal from the rear
    2. **Insertion at the rear and removal from the front**
    3. Insertion and removal only at the rear
    4. Insertion and removal only at the front
23. Which of the following statements is true for a circular queue with no elements?
    1. The front and rear pointers are both NULL.
    2. The front and rear pointers are both -1.
    3. **The front and rear pointers are equal.**
    4. The front and rear pointers are not defined.
24. In a circular queue, if the front and rear pointers are both -1, it means that the queue is:
    1. Full
    2. **Empty**
    3. Overflowed
    4. Underflowed
25. Which of the following statements is true for a circular queue with only one element?
    1. **The front and rear pointers are equal.**
    2. The front and rear pointers are both -1.
    3. The front and rear pointers are both NULL.
    4. The front and rear pointers are not defined.
26. What is the maximum number of elements that can be stored in a circular queue of size 5?
    1. 4
    2. **5**
    3. 6
    4. 10
27. In a circular queue, if the front pointer is ahead of the rear pointer by 1 position, it means that the queue is:
    1. **Full**
    2. Empty
    3. Overflowed
    4. Underflowed
28. What is the time complexity of finding the front element of a circular queue?
    1. **O(1)**
    2. O(n)
    3. O(log n)
    4. O(n^2)
29. What is the time complexity of finding the rear element of a circular queue?
    1. **O(1)**
    2. O(n)
    3. O(log n)
    4. O(n^2)
30. How can you access the elements of a circular queue?
    1. Random access using indices
    2. Sequential access using iterators
    3. Random access using pointers
    4. **Sequential access using front and rear pointers**
31. Which of the following operations is used to resize a circular queue?
    1. resizeQueue
    2. expandQueue
    3. shrinkQueue
    4. **Circular queues cannot be resized.**
32. In a circular queue, the front and rear pointers move:
    1. Clockwise
    2. Counterclockwise
    3. **Both clockwise and counterclockwise**
    4. They don't move
33. Which of the following statements is true for a circular queue with N elements?
    1. **The front and rear pointers are equal.**
    2. The front and rear pointers are both NULL.
    3. The front and rear pointers are both -1.
    4. The front and rear pointers are N-1.
34. How can you implement a circular queue using a dynamic array in C?
    1. **By using a fixed-size array and adjusting the front and rear pointers accordingly**
    2. By using a linked list instead of an array
    3. By using a resizable array implementation like the vector in C++
    4. Circular queues cannot be implemented using dynamic arrays.
35. What will be the output of the following code?

#include <stdio.h>

#define MAX\_SIZE 5

typedef struct {

int front;

int rear;

int elements[MAX\_SIZE];

} CircularQueue;

void enqueue(CircularQueue\* queue, int value) {

if ((queue->rear + 1) % MAX\_SIZE == queue->front) {

printf("Queue is full!\n");

return;

}

queue->rear = (queue->rear + 1) % MAX\_SIZE;

queue->elements[queue->rear] = value;

}

int dequeue(CircularQueue\* queue) {

if (queue->front == queue->rear) {

printf("Queue is empty!\n");

return -1;

}

queue->front = (queue->front + 1) % MAX\_SIZE;

int value = queue->elements[queue->front];

return value;

}

int main() {

CircularQueue queue = {0, 0, {0}};

enqueue(&queue, 10);

enqueue(&queue, 20);

enqueue(&queue, 30);

printf("%d\n", dequeue(&queue));

enqueue(&queue, 40);

enqueue(&queue, 50);

enqueue(&queue, 60);

printf("%d\n", dequeue(&queue));

return 0;

}

Options:

a) 10, Queue is empty!

b) 10, 20

**c) 10, 20, Queue is full!**

d) Queue is empty!