1. What is a linked list?
   1. **A linear data structure**
   2. A non-linear data structure
   3. A binary data structure
   4. An array-based data structure
2. Which node of a linked list contains the actual data?
   1. Head node
   2. Tail node
   3. Intermediate node
   4. **All nodes contain data**
3. In a singly linked list, how is the last node identified?
   1. **By a NULL pointer**
   2. By pointing to itself
   3. By pointing to the previous node
   4. By a special end-of-list marker
4. What is the time complexity to insert a node at the beginning of a singly linked list?
   1. **O(1)**
   2. O(n)
   3. O(log n)
   4. O(n^2)
5. What is the time complexity to delete a node from the end of a singly linked list?
   1. O(1)
   2. **O(n)**
   3. O(log n)
   4. O(n^2)
6. Which type of linked list allows traversal in both directions?
   1. Singly linked list
   2. **Doubly linked list**
   3. Circular linked list
   4. Array-based list
7. What is a circular linked list?
   1. A linked list with circular data
   2. **A linked list that points to the same node**
   3. A linked list with a loop
   4. A linked list that connects to another list
8. Which node is pointed to by the "next" pointer of the last node in a circular linked list?
   1. NULL
   2. **The first node**
   3. The last node
   4. The previous node
9. How can you prevent memory leaks in a linked list?
   1. By using a smart pointer
   2. By using automatic memory management
   3. **By manually deallocating memory for each node**
   4. Linked lists don't require memory deallocation
10. What is the time complexity to search for an element in an unsorted linked list?
    1. O(1)
    2. **O(n)**
    3. O(log n)
    4. O(n^2)
11. Which operation is not possible in a singly linked list?
    1. Insertion at the beginning
    2. Insertion at the end
    3. Deletion from the beginning
    4. **Deletion from the middle**
12. What is the space complexity of a linked list?
    1. O(1)
    2. **O(n)**
    3. O(log n)
    4. O(n^2)
13. Which node must be updated when inserting a new node at the beginning of a singly linked list?
    1. Previous node
    2. Last node
    3. **Head node**
    4. Tail node
14. Which type of linked list is suitable for implementing a stack?
    1. **Singly linked list**
    2. Doubly linked list
    3. Circular linked list
    4. Array-based list
15. What is a self-referential structure in the context of linked lists?
    1. **A structure that refers to itself**
    2. A structure with multiple pointers
    3. A structure with a cyclic reference
    4. A structure that refers to another structure
16. Which type of linked list is best suited for implementing a queue?
    1. Singly linked list
    2. Doubly linked list
    3. **Circular linked list**
    4. Array-based list
17. Which operation requires updating both the "next" and "previous" pointers in a doubly linked list?
    1. Insertion at the beginning
    2. Insertion at the end
    3. Deletion from the beginning
    4. **Deletion from the end**
18. What is the time complexity to reverse a singly linked list iteratively?
    1. O(1)
    2. **O(n)**
    3. O(log n)
    4. O(n^2)
19. Which operation in a linked list requires traversal of the entire list?
    1. Insertion at the beginning
    2. Insertion at the end
    3. Deletion from the beginning
    4. **Deletion from the end**
20. In a circular linked list, how do you determine if a given value exists?
    1. **Perform a linear search**
    2. Perform a binary search
    3. Use a hash table
    4. Circular linked lists cannot contain values
21. What is the advantage of using a doubly linked list over a singly linked list?
    1. Smaller memory footprint
    2. Faster traversal
    3. **Ability to traverse in both directions**
    4. Easier memory management
22. struct Node {

int data;

Node\* next;

Node(int val) : data(val), next(nullptr) {}

};

void printList(Node\* head) {

while (head != nullptr) {

cout << head->data << " ";

head = head->next;

}

}

int main() {

Node\* head = new Node(1);

head->next = new Node(2);

head->next->next = new Node(3);

printList(head);

return 0;

}

* 1. **1 2 3**
  2. 3 2 1
  3. 1
  4. Compilation Error

1. Which node is modified when deleting a node from the middle of a singly linked list?
   1. **Previous node**
   2. Next node
   3. Current node
   4. Head node
2. struct Node {

int data;

Node\* next;

Node(int val) : data(val), next(nullptr) {}

};

int main() {

Node\* head = new Node(1);

Node\* temp = head;

head = nullptr;

delete temp;

cout << temp->data;

return 0;

}

* 1. **0**
  2. 1
  3. Error
  4. Garbage Value

1. Which operation is not possible in a circular linked list?
   1. Insertion at the beginning
   2. Insertion at the end
   3. Deletion from the beginning
   4. **Deletion from the middle**
2. struct Node {

int data;

Node\* next;

};

int main() {

Node\* head = nullptr;

Node\* newNode = new Node{2, nullptr};

head->next = newNode;

cout << head->next->data;

return 0;

}

* 1. 0
  2. 2
  3. Garbage value
  4. **Compilation error**

1. What is the primary disadvantage of using a linked list over an array?
   1. **Slower random access**
   2. Inability to store data
   3. Limited memory usage
   4. Difficulty in implementation
2. Which type of linked list is used to implement a dynamic array?
   1. **Singly linked list**
   2. Doubly linked list
   3. Circular linked list
   4. Array-based list
3. #include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

int main() {

Node\* head = nullptr;

Node\* temp = new Node;

temp->data = 5;

temp->next = nullptr;

head = temp;

cout << head->data;

return 0;

}

* 1. 0
  2. **5**
  3. Error
  4. Garbage Value

1. How is a node added to the end of a singly linked list efficiently?
   1. Traverse the entire list and add the node
   2. **Update the tail node's pointer to the new node**
   3. Add the node before the head node
   4. Linked lists cannot have nodes at the end
2. What is the time complexity of merging two sorted singly linked lists into one sorted linked list?
   1. O(1)
   2. **O(n)**
   3. O(log n)
   4. O(n^2)
3. struct Node {

int data;

Node\* next;

Node(int val) : data(val), next(nullptr) {}

};

int main() {

Node\* head = new Node(3);

cout << head->data;

return 0;

}

* 1. 0
  2. **3**
  3. Error
  4. Garbage Value

1. Which linked list type can be easily implemented using an array?
   1. Singly linked list
   2. Doubly linked list
   3. Circular linked list
   4. **Array-based list**
2. struct Node {

int data;

Node\* next;

Node(int val) : data(val), next(nullptr) {}

};

int main() {

Node\* head = new Node(2);

head->next = new Node(4);

cout << head->next->data;

return 0;

}

* 1. 0
  2. 2
  3. **4**
  4. Garbage Value

1. In a doubly linked list, what is the purpose of the "previous" pointer in the head node?
   1. It points to the last node
   2. It points to the next node
   3. It indicates the beginning of the list
   4. **Doubly linked lists cannot have a "previous" pointer in the head node**