1. struct Node {

int data;

Node\* next;

};

What is the size (in bytes) of the Node structure on a 64-bit system?

a) 4 bytes

b) 8 bytes

c) 12 bytes

**d) 16 bytes**

2. In a singly linked list, each node contains:

**a) Data and a pointer to the next node**

b) Data and a pointer to the previous node

c) Data only

d) A pointer to the previous node only

3. When using dynamic memory allocation to create nodes in a linked list, what is the correct way to deallocate the entire list to avoid memory leaks?

**a)**

**void deallocateList(Node\* head) {**

**Node\* temp;**

**while (head != nullptr) {**

**temp = head;**

**head = head->next;**

**delete temp;**

**}**

**}**

b)

void deallocateList(Node\* head) {

while (head != nullptr) {

delete head;

head = head->next;

}

}

c)

void deallocateList(Node\* head) {

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

}

d)

void deallocateList(Node\* head) {

delete head;

}

4. What is the time complexity to search for an element in a singly linked list with n nodes?

a) O(1)

b) O(n)

**c) O(n)**

d) O(log n)

5. struct Node {

int data;

Node\* next;

};

int main() {

Node\* head = new Node;

head->data = 10;

head->next = new Node;

head->next->data = 20;

head->next->next = NULL;

delete head->next;

delete head;

return 0;

}

What is the memory status after executing the above code?

a) Memory is leaked due to not deallocating the nodes

**b) Memory is properly deallocated**

c) The code contains a syntax error

d) The code will throw a runtime error

6. Which operation is used to add an element at the beginning of a linked list?

**a)Insertion at the head**

b) Insertion at the tail

c) Deletion from the head

d) Deletion from the tail

7.

struct Node {

int data;

Node\* next;

};

Node\* createNode(int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = NULL;

return newNode;

}

int main() {

Node\* head = createNode(10);

head->next = createNode(20);

head->next->next = createNode(30);

head = head->next;

delete head;

return 0;

}

What is the memory status after executing the above code?

**a) Memory is leaked due to not deallocating the first node**

b) Memory is properly deallocated

c) The code contains a syntax error

d) The code will throw a runtime error

8. The time complexity for inserting an element at the end of a linked list with n nodes is:

a) O(1)

b) O(log n)

c) O(n)

**d) O(1) (if tail is known, otherwise O(n))**

9. struct Node {

int data;

Node\* next;

};

int main() {

Node\* head = new Node;

head->data = 10;

head->next = new Node;

head->next->data = 20;

head->next->next = head;

delete head->next;

delete head;

return 0;

}

What is the memory status after executing the above code?

**a) Memory is leaked due to a cycle in the linked list**

b) Memory is properly deallocated

c) The code contains a syntax error

d) The code will throw a runtime error

10. In a singly linked list, the last node points to:

a) The head node

b) The next node in the list

**c) NULL**

d) The previous node

11. struct Node {

int data;

Node\* next;

};

int main() {

Node\* head = new Node;

head->data = 10;

head->next = new Node;

head->next->data = 20;

head->next->next = NULL;

delete head;

return 0;

}

What is the memory status after executing the above code?

a) Memory is properly deallocated

**b) Memory is leaked due to not deallocating the second node**

c) The code contains a syntax error

d) The code will throw a runtime error

12. The process of removing a node from a linked list is known as:

**a) Deletion**

b) Insertion

c) Traversal

d) Searching

13. struct Node {

int data;

Node\* next;

};

Node\* createNode(int value) {

Node newNode;

newNode.data = value;

newNode.next = NULL;

return &newNode;

}

int main() {

Node\* head = createNode(10);

head->next = createNode(20);

head->next->next = createNode(30);

delete head;

return 0;

}

What is the memory status after executing the above code?

a) Memory is properly deallocated

**b) Memory is leaked due to returning a local variable's address**

c) The code contains a syntax error

d) The code will throw a runtime error

14. The memory allocation for a linked list is typically done using:

**a) Dynamic memory allocation (e.g., using 'new' or 'malloc')**

b) Static memory allocation (e.g., declaring an array)

c) Heap memory allocation

d) Stack memory allocation

15. #include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

int main() {

Node\* head = new Node();

head->data = 10;

head->next = nullptr;

Node\* newNode = new Node();

newNode->data = 20;

newNode->next = head;

head = newNode;

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

return 0;

}

What will be the output when the above code is executed?

a) 10 20

**b) 20 10**

c) 10 10

d) Compilation Error

16. The process of accessing each node of a linked list in sequence is known as:

**a) Traversal**

b) Insertion

c) Deletion

d) Searching

17. #include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

int main() {

Node\* head = new Node();

head->data = 10;

head->next = nullptr;

Node\* newNode = new Node();

newNode->data = 20;

newNode->next = head;

head = newNode;

delete head;

cout << head->data;

return 0;

}

What will be the output when the above code is executed?

a) 10

b) 20

**c) Garbage Value**

d) Runtime Error

18. Which of the following operations can be performed efficiently on a linked list?

**a) Insertion and deletion of elements in the middle**

b) Directly accessing the element at any index

c) Sorting the elements using an efficient algorithm

d) Searching for an element in constant time

19. #include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

int main() {

Node\* head = nullptr;

Node newNode;

newNode.data = 10;

newNode.next = nullptr;

head = &newNode;

cout << head->data;

return 0;

}

What will be the output when the above code is executed?

**a) 10**

b) Garbage Value

c) Compilation Error

d) Runtime Error

20. The "head" pointer in a linked list points to:

a) The first element of the list

b) The last element of the list

c) The middle element of the list

**d) The first node (starting point) of the list**