1. Which operation is used to traverse a linked list and process each node one by one?

a) Insertion

b) Deletion

**c) Traversal**

d) Searching

2. Consider the following code snippet to delete a node with a given key from a singly linked list:

void deleteNode(Node\* head, int key) {

Node\* prev = nullptr;

Node\* current = head;

while (current != nullptr && current->data != key) {

prev = current;

current = current->next;

}

if (current != nullptr) {

prev->next = current->next;

delete current;

}

}

What will happen if the node with the given key is not found in the linked list?

a) The program will crash due to a segmentation fault.

b) The program will enter an infinite loop.

c) The program will print an error message.

**d) The program will continue executing normally.**

3. 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

4. Consider the following code to delete a node from a singly linked list:

void deleteNode(Node\* node) {

if (node == nullptr || node->next == nullptr) {

return;

}

Node\* temp = node->next;

node->data = temp->data;

node->next = temp->next;

delete temp;

}

What does this function do?

a) Deletes the node at the given address and deallocates the memory.

b) Deletes the first node in the linked list.

c) Deletes the last node in the linked list.

**d) Deletes the node at the given address without deallocating the memory**

5. 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))**

6. Which of the following code snippets correctly implements a function to delete the entire linked list?

**a)**

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

**Node\* current = head;**

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

**Node\* temp = current;**

**current = current->next;**

**delete temp;**

**}**

**}**

b)

void deleteList(Node\* head) {

delete head;

}

c)

void deleteList(Node\* head) {

while (head != nullptr) {

delete head;

head = head->next;

}

}

d)

void deleteList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

head = head->next;

delete current;

current = head;

}

}

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

a) Deletion

b) Insertion

**c) Deletion**

d) Searching

8. #include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

void deleteNode(Node\*& head, int value) {

if (head == NULL) return;

if (head->data == value) {

Node\* temp = head;

head = head->next;

delete temp;

return;

}

Node\* current = head;

while (current->next != NULL) {

if (current->next->data == value) {

Node\* temp = current->next;

current->next = temp->next;

delete temp;

return;

}

current = current->next;

}

}

int main() {

Node\* list = NULL;

for (int i = 1; i <= 5; i++) {

Node\* newNode = new Node;

newNode->data = i;

newNode->next = list;

list = newNode;

}

deleteNode(list, 3);

Node\* current = list;

while (current != NULL) {

cout << current->data << " ";

current = current->next;

}

return 0;

}

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

a) 1 2 4 5

**b) 5 4 2 1**

c) 1 2 3 4 5

d) Compilation Error

9. Which type of linked list has a loop/cycle in the list?

a) Singly linked list

b) Doubly linked list

**c) Circular linked list**

d) Cyclic linked list

10. #include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

void deleteAlternate(Node\*& head) {

if (head == NULL || head->next == NULL) return;

Node\* current = head;

while (current != NULL && current->next != NULL) {

Node\* temp = current->next;

current->next = temp->next;

delete temp;

current = current->next;

}

}

int main() {

Node\* list = NULL;

for (int i = 1; i <= 5; i++) {

Node\* newNode = new Node;

newNode->data = i;

newNode->next = list;

list = newNode;

}

deleteAlternate(list);

Node\* current = list;

while (current != NULL) {

cout << current->data << " ";

current = current->next;

}

return 0;

}

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

**a) 5 3 1**

b) 5 4 3 2 1

c) 1 2 3 4 5

d) Compilation Error

11. In a doubly linked list, what is the time complexity to delete a node with a given value?

a) O(1)

b) O(log n)

**c) O(n)**

d) O(n log n)

12. Node\* do(Node\* head) {

if (head == NULL) return NULL;

Node\* current = head;

while (current->next != NULL) {

if (current->data == current->next->data) {

Node\* temp = current->next;

current->next = temp->next;

delete temp;

} else {

current = current->next;

}

}

return head;

}

What the above function do?

a) Remove first element

b) Remove second element

**c) Remove duplicate elements**

d) Remove all elements

13. What is the space complexity of a linked list with n nodes?

a) O(n)

b) O(1)

**c) O(n)**

d) O(log n)

14. void deleteNode(Node\*& head, int key) {

if (head == nullptr)

return;

if (head->data == key) {

Node\* temp = head;

head = head->next;

delete temp;

return;

}

Node\* current = head;

Node\* prev = nullptr;

while (current != nullptr && current->data != key) {

prev = current;

current = current->next;

}

if (current == nullptr)

return;

prev->next = current->next;

delete current;

}

What will be the output of the code when executed with the key value of 5?

a) The linked list will remain unchanged.

b) The first occurrence of the node with the value 5 will be deleted.

**c) All occurrences of the node with the value 5 will be deleted.**

d) The last occurrence of the node with the value 5 will be deleted.

15. Which operation is used to insert a new node after a specific node in a linked list?

a) Insertion at the head

**b) Insertion after a given node**

c) Insertion at the tail

d) Insertion before a given node

16. void deleteNode(Node\*& head, int key) {

if (head == nullptr)

return;

if (head->data == key) {

Node\* temp = head;

head = head->next;

delete temp;

return;

}

Node\* current = head;

while (current->next != nullptr) {

if (current->next->data == key) {

Node\* temp = current->next;

current->next = current->next->next;

delete temp;

return;

}

current = current->next;

}

}

What will be the output of the code when executed with the key value of 5?

a) The linked list will remain unchanged.

**b) The first occurrence of the node with the value 5 will be deleted.**

c) All occurrences of the node with the value 5 will be deleted.

d) The last occurrence of the node with the value 5 will be deleted.

17. In a doubly linked list, what is the time complexity to insert a new node after a given node?

a) O(1)

b) O(log n)

**c) O(1)**

d) O(n)

18. void operation(Node\* &head, int key) {

Node\* temp = head;

Node\* prev = nullptr;

while (temp != nullptr && temp->data != key) {

prev = temp;

temp = temp->next;

}

if (temp == nullptr) return;

if (prev == nullptr) {

head = head->next;

delete temp;

} else {

prev->next = temp->next;

delete temp;

}

}

What will be the output when the above code is executed if the linked list is 2 -> 5 -> 7 -> 9 -> 12, and key = 7?

**a) 2 -> 5 -> 9 -> 12**

b) 2 -> 5 -> 7 -> 9 -> 12

c) 2 -> 5 -> 9

d) 2 -> 5 -> 7 -> 12

19. The process of freeing the memory occupied by a node after it has been deleted from a linked list is known as:

a) Memory deallocation

b) Memory release

**c) Memory deallocation**

d) Memory garbage collection

20. void deleteNode(Node\*& head, int key) {

Node\* temp = head;

while (temp != nullptr && temp->data != key) {

temp = temp->next;

}

if (temp == nullptr) {

return;

}

if (temp->prev != nullptr) {

temp->prev->next = temp->next;

}

if (temp->next != nullptr) {

temp->next->prev = temp->prev;

}

if (temp == head) {

head = temp->next;

}

delete temp;

}

What will happen if the node with data value 2 is the only node in the doubly linked list?

**a) The doubly linked list will become empty after the deletion.**

b) The program will delete the first node in the list.

c) The program will not delete any node.

d) The program will crash due to a null pointer exception.