1. What is a linked list?

a) A list of elements arranged in a circular manner

b) A list of elements with a fixed size

**c)A linear data structure in which elements are connected using pointers**

d) A list of elements that can only be accessed sequentially

2. The following C function takes a simply-linked list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.

typedef struct node

{

int value;

struct node \*next;

}Node;

Node \*move\_to\_front(Node \*head)

{

Node \*p, \*q;

if ((head == NULL: || (head->next == NULL))

return head;

q = NULL; p = head;

while (p-> next !=NULL)

{

q = p;

p = p->next;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return head;

}

a) q = NULL; p->next = head; head = p;

b) q->next = NULL; head = p; p->next = head;

c) head = p; p->next = q; q->next = NULL;

**d) q->next = NULL; p->next = head; head = p;**

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

a) Insertion

b) Deletion

**c) Traversal**

d) Searching

4. What is the task of below function?

void fun1(struct node\* head)

{

if(head == NULL)

return;

fun1(head->next);

printf("%d ", head->data);

}

a) Prints all nodes of linked lists

**b) Prints all nodes of linked list in reverse order**

c) Prints alternate nodes of Linked List

d) Prints alternate nodes in reverse order

5. In a doubly 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, a pointer to the next node, and a pointer to the previous node**

d) Data only

6.

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* insertBefore(Node\* head, Node\* nextNode, int value) {

if (head == nullptr) {

return nullptr;

}

if (head == nextNode) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = head;

head = newNode;

return head;

}

Node\* temp = head;

while (temp->next != nextNode && temp->next != nullptr) {

temp = temp->next;

}

if (temp->next == nullptr) {

return head;

}

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nextNode;

temp->next = newNode;

return head;

}

void display(Node\* head) {

Node\* temp = head;

while (temp != nullptr) {

cout << temp->data << " ";

temp = temp->next;

}

}

int main() {

Node\* head = nullptr;

Node\* second = nullptr;

Node\* third = nullptr;

head = new Node;

second = new Node;

third = new Node;

head->data = 1;

head->next = second;

second->data = 3;

second->next = third;

third->data = 5;

third->next = nullptr;

head = insertBefore(head, second, 2);

head = insertBefore(head, third, 4);

display(head);

return 0;

}

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

**a) 1 2 3 4 5**

b) 1 3 2 5 4

c) 1 3 5 2 4

d) 1 5 4 3 2

7. The process of creating a new node and connecting it to the linked list is known as:

a) Deletion

**b) Insertion**

c) Traversal

d) Searching

8. #include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* reverseList(Node\* head) {

Node\* prev = nullptr;

Node\* current = head;

Node\* next = nullptr;

while (current != nullptr) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

return prev;

}

void display(Node\* head) {

Node\* temp = head;

while (temp != nullptr) {

cout << temp->data << " ";

temp = temp->next;

}

}

int main() {

Node\* head = nullptr;

head = new Node;

head->data = 1;

Node\* second = new Node;

second->data = 2;

head->next = second;

Node\* third = new Node;

third->data = 3;

second->next = third;

third->next = nullptr;

head = reverseList(head);

display(head);

return 0;

}

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

**a) 3 2 1**

b) 1 2 3

c) 1 3 2

d) 3 1 2

9. What is the main advantage of using a linked list over an array?

a) Linked lists have a fixed size.

b) Linked lists have better cache locality.

c) Linked lists are stored in contiguous memory.

**d) Linked lists allow efficient insertions and deletions.**

10. void fun(struct node \*\*head\_ref)

{

struct node \*temp = NULL;

struct node \*current = \*head\_ref;

while (current != NULL)

{

temp = current->prev;

current->prev = current->next;

current->next = temp;

current = current->prev;

}

if(temp != NULL )

\*head\_ref = temp->prev;

}

linked list is passed to above function is 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the new linked list after the function call?

a) 2 <--> 1 <--> 4 <--> 3 <--> 6 <-->5

b) 5 <--> 4 <--> 3 <--> 2 <--> 1 <-->6.

**c) 6 <--> 5 <--> 4 <--> 3 <--> 2 <--> 1.**

d) 6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2

11. 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

12. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

struct node

{

int data;

struct node\* next;

};

static void reverse(struct node\*\* head\_ref)

{

struct node\* prev = NULL;

struct node\* current = \*head\_ref;

struct node\* next;

while (current != NULL)

{

next = current->next;

current->next = prev;

prev = current;

current = next;

}

/\*-------------------------\*/

}

What should be added in place of "---------------", so that the function correctly reverses a linked list.

**a) \*head\_ref = prev;**

b) \*head\_ref = current;

c) \*head\_ref = next;

d) \*head\_ref = NULL

13. What is the time complexity to delete an element from the end of a singly linked list with n nodes?

a) O(1)

b) O(log n)

**c) O(n) (if tail is not known)**

d) O(n) (if tail is known)

14. #include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* insertAfter(Node\* head, Node\* prevNode, int value) {

if (prevNode == nullptr) {

return head;

}

Node\* newNode = new Node;

newNode->data = value;

newNode->next = prevNode->next;

prevNode->next = newNode;

return head;

}

void display(Node\* head) {

Node\* temp = head;

while (temp != nullptr) {

cout << temp->data << " ";

temp = temp->next;

}

}

int main() {

Node\* head = nullptr;

Node\* second = nullptr;

Node\* third = nullptr;

head = new Node;

second = new Node;

third = new Node;

head->data = 1;

head->next = second;

second->data = 3;

second->next = third;

third->data = 5;

third->next = nullptr;

head = insertAfter(head, second, 2);

head = insertAfter(head, third, 4);

display(head);

return 0;

}

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

a) 1 3 5 2 4

**b) 1 3 2 5 4**

c) 1 2 3 4 5

d) 1 5 4 3 2

15. 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**

16.

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* insertAtBegin(Node\* head, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = head;

head = newNode;

return head;

}

void display(Node\* head) {

Node\* temp = head;

while (temp != nullptr) {

cout << temp->data << " ";

temp = temp->next;

}

}

int main() {

Node\* head = nullptr;

head = insertAtBegin(head, 1);

head = insertAtBegin(head, 2);

head = insertAtBegin(head, 3);

head = insertAtBegin(head, 4);

display(head);

return 0;

}

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

**a) 4 3 2 1**

b) 1 2 3 4

c) 1 4 3 2

d) 2 3 4 1

17. 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)

18. #include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* insertAtEnd(Node\* head, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (head == nullptr) {

return newNode;

}

Node\* temp = head;

while (temp->next != nullptr) {

temp = temp->next;

}

temp->next = newNode;

return head;

}

void display(Node\* head) {

Node\* temp = head;

while (temp != nullptr) {

cout << temp->data << " ";

temp = temp->next;

}

}

int main() {

Node\* head = nullptr;

head = insertAtEnd(head, 1);

head = insertAtEnd(head, 2);

head = insertAtEnd(head, 3);

head = insertAtEnd(head, 4);

display(head);

return 0;

}

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

**a) 1 2 3 4**

b) 4 3 2 1

c) 1 4 2 3

d) 1 2 4 3

19. 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)

20. What is the output of following function in which linked list is 1->2->3->4->5->6 ?

void fun(struct node\* start)

{

if(start == NULL)

return;

printf("%d ", start->data);

if(start->next != NULL )

fun(start->next->next);

printf("%d ", start->data);

}

a) 1 4 6 6 4 1

b) 1 3 5 1 3 5

c) 1 2 3 5

**d) 1 3 5 5 3 1**