Q. 1. Write a menu driven program to perform following operations on singly linked list: Create,

reverse, search, count and Display

/\*Write a menu driven program to perform following

operations on singly linked list: Create, reverse,

search, count and Display.

\*/

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

class LinkedList {

private:

Node\* head;

int count;

public:

LinkedList() {

head = NULL;

count = 0;

}

void create() {

int data;

cout << "Enter the data for the node: ";

cin >> data;

Node\* newNode = new Node();

newNode->data = data;

newNode->next = head;

head = newNode;

count++;

}

void reverse() {

Node\* prev = NULL;

Node\* current = head;

Node\* next = NULL;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

}

int search(int key) {

Node\* current = head;

int index = 0;

while (current != NULL) {

if (current->data == key) {

return index;

}

current = current->next;

index++;

}

return -1;

}

int countNodes() {

return count;

}

void display() {

Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

cout << endl;

}

};

int main() {

int choice;

LinkedList list;

while (true) {

cout << "1. Create Node" << endl;

cout << "2. Reverse List" << endl;

cout << "3. Search Element" << endl;

cout << "4. Count Nodes" << endl;

cout << "5. Display List" << endl;

cout << "6. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

list.create();

break;

case 2:

list.reverse();

break;

case 3: {

int key;

cout << "Enter the element to be searched: ";

cin >> key;

int index = list.search(key);

if (index == -1) {

cout << "Element not found." << endl;

} else {

cout << "Element found at index: " << index << endl;

}

break;

}

case 4:

cout << "Number of nodes: " << list.countNodes() << endl;

break;

case 5:

list.display();

break;

case 6:

return 0;

default:

cout << "Invalid choice. Please enter a valid choice." << endl;

}

}

return 0;

}

Q. 2. Implement Linked queue

/\*. Implement Linked queue\*/

#include <bits/stdc++.h>

using namespace std;

struct QNode {

int data;

QNode\* next;

QNode(int d)

{

data = d;

next = NULL;

}

};

struct Queue {

QNode \*front, \*rear;

Queue() { front = rear = NULL; }

void enQueue(int x)

{

// Create a new LL node

QNode\* temp = new QNode(x);

// If queue is empty, then

// new node is front and rear both

if (rear == NULL) {

front = rear = temp;

return;

}

// Add the new node at

// the end of queue and change rear

rear->next = temp;

rear = temp;

}

// Function to remove

// a key from given queue q

void deQueue()

{

// If queue is empty, return NULL.

if (front == NULL)

return;

// Store previous front and

// move front one node ahead

QNode\* temp = front;

front = front->next;

// If front becomes NULL, then

// change rear also as NULL

if (front == NULL)

rear = NULL;

delete (temp);

}

};

// Driver code

int main()

{

Queue q;

q.enQueue(10);

q.enQueue(20);

q.deQueue();

q.deQueue();

q.enQueue(30);

q.enQueue(40);

q.enQueue(50);

q.deQueue();

cout << "Queue Front : " << (q.front)->data << endl;

cout << "Queue Rear : " << (q.rear)->data;

}