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

Insert, Delete, and Display

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* head = NULL;

void insert(int x) {

Node\* temp = new Node();

temp->data = x;

temp->next = head;

head = temp;

}

void Delete(int n) {

Node\* temp1 = head;

if(n == 1) {

head = temp1->next;

delete temp1;

return;

}

for(int i=0; i<n-2; i++) {

temp1 = temp1->next;

}

Node\* temp2 = temp1->next;

temp1->next = temp2->next;

delete temp2;

}

void display() {

Node\* temp = head;

while(temp != NULL) {

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

temp = temp->next;

}

cout << endl;

}

int main() {

int choice, x, n;

while(1) {

cout << "1. Insert" << endl;

cout << "2. Delete" << endl;

cout << "3. Display" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch(choice) {

case 1: cout << "Enter the element: ";

cin >> x;

insert(x);

break;

case 2: cout << "Enter the element you want to delete: ";

cin >> n;

Delete(n);

break;

case 3: display();

break;

case 4: exit(0);

default: cout << "Invalid Input" << endl;

}

}

return 0;

}

Q. 2. Implement Sequential and Binary Search

/\*Implement Sequential Search.\*/

#include <iostream>

using namespace std;

int sequentialSearch(int array[], int size, int key) {

for (int i = 0; i < size; i++) {

if (array[i] == key) {

return i; // return the index of the key if found

}

}

return -1; // return -1 if key is not found

}

int main() {

int array[] = {1, 2, 3, 4, 5};

int size = sizeof(array) / sizeof(array[0]);

int key = 3;

int index = sequentialSearch(array, size, key);

if (index != -1) {

cout << "Key found at index " << index << endl;

} else {

cout << "Key not found" << endl;

}

return 0;

}

/\*Implement Binary Search.\*/

#include <iostream>

using namespace std;

int binarySearch(int arr[], int n, int key) {

int left = 0, right = n - 1;

while (left <= right) {

int mid = (left + right) / 2;

if (arr[mid] == key) {

return mid;

}

else if (arr[mid] < key) {

left = mid + 1;

}

else {

right = mid - 1;

}

}

return -1;

}

int main() {

int arr[] = {1, 2, 3, 4, 5};

int n = sizeof(arr) / sizeof(arr[0]);

int key = 3;

int index = binarySearch(arr, n, key);

if (index != -1) {

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

}

else {

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

}

return 0;

}