Q. 1. 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;

}

Q. 2. Implement circular queue using arrays.

/\*Implement circular queue using arrays. \*/

#include <iostream>

using namespace std;

class CircularQueue {

int \*queue, size, front, rear;

public:

CircularQueue(int s) {

size = s;

queue = new int[size];

front = rear = -1;

}

void enqueue(int x);

int dequeue();

void display();

};

void CircularQueue::enqueue(int x) {

if ((front == 0 && rear == size - 1) || (front == rear + 1)) {

cout << "Queue is full\n";

return;

}

else if (front == -1) {

front = rear = 0;

}

else if (rear == size - 1 && front != 0) {

rear = 0;

}

else {

rear++;

}

queue[rear] = x;

}

int CircularQueue::dequeue() {

if (front == -1) {

cout << "Queue is empty\n";

return -1;

}

int x = queue[front];

if (front == rear) {

front = rear = -1;

}

else if (front == size - 1) {

front = 0;

}

else {

front++;

}

return x;

}

void CircularQueue::display() {

if (front == -1) {

cout << "Queue is empty\n";

return;

}

if (rear >= front) {

for (int i = front; i <= rear; i++)

cout << queue[i] << " ";

}

else {

for (int i = front; i < size; i++)

cout << queue[i] << " ";

for (int i = 0; i <= rear; i++)

cout << queue[i] << " ";

}

}

int main() {

CircularQueue q(5);

q.enqueue(1);

q.enqueue(2);

q.enqueue(3);

q.enqueue(4);

q.enqueue(5);

q.enqueue(6);

q.display();

cout << endl;

q.dequeue();

q.dequeue();

q.display();

cout << endl;

return 0;

}