|  |  |
| --- | --- |
| **Name** | **Taaha Hussain Khan** |
| **Roll No** | **L1F21BSCS0917** |
| **Section** | **D12** |
| **Assignment** | **01** |

**Task 1**

//Taaha Hussain Khan

//L1F21BSCS0917

//D12

#include<iostream>

using namespace std;

void BinarySearch(int\*,int,int);

int main()

{

cout << "Enter size of your array : ";

int size;

cin >> size;

int\* arr = new int[size];

cout << "Enter values : \n";

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

{

cin >> arr[i];

}

cout << "Array : ";

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

{

cout << arr[i] << " ";

}

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

{

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

{

if (arr[i] < arr[j])

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

cout << endl;

cout << "Sorted Array : ";

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

{

cout << arr[i] << " ";

}

cout << endl;

cout << "Key: ";

int key;

cin >> key;

BinarySearch(arr, size, key);

}

void BinarySearch(int\* arr, int size, int key)

{

int start = 0;

int check = 0;

int end = size;

int mid = 0;

while (start <= end)

{

mid = (start + end) / 2;

if (arr[mid] == key)

{

check = 0;

cout << endl;

cout << key << " found at " << mid;

if (mid == 1)

cout << "st" << " index" << endl;

else if (mid == 2)

cout << "nd" << " index" << endl;

else if (mid == 3)

cout << "rd" << " index" << endl;

else

cout << "th" << " index" << endl;

break;

}

else if (arr[mid] < key)

{

check++;

start = mid + 1;

}

else

{

check++;

end = mid - 1;

}

}

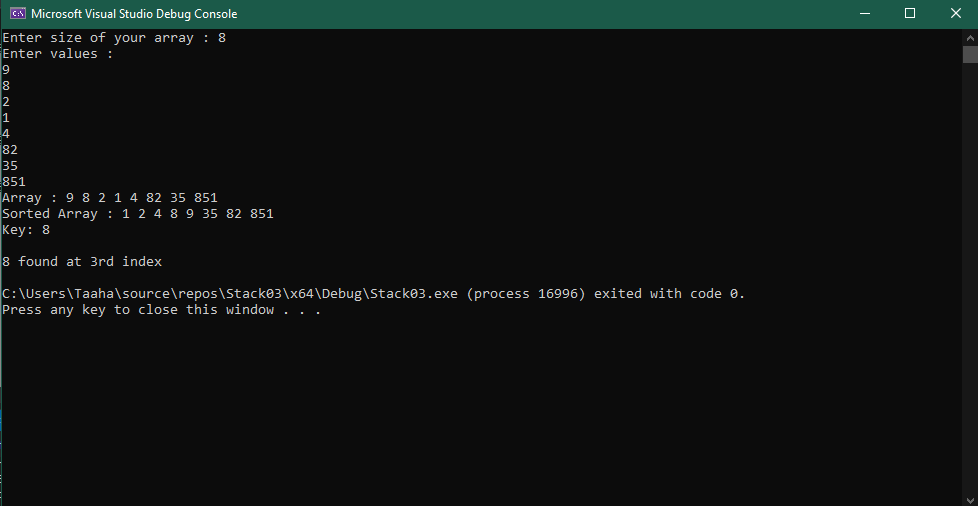
if (check > 0)

{

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

}

}



**Task 2**

//Taaha Hussain Khan

//L1F21BSCS0917

//D12

#include <iostream>

#define SIZE 25

using namespace std;

template<class type>

class Queue

{

protected:

type\* arr;

int currentSize;

int maxSize;

public:

Queue(int s)

{

currentSize = 0;

maxSize = s;

arr = new type[maxSize];

}

virtual void enQueue(type) = 0;

virtual type deQueue() = 0;

};

template<class type>

class myQueue : public Queue<type>

{

public:

myQueue(int s) : Queue<type>(s)

{

//Already values

Queue<type>::arr[0] = 10;

Queue<type>::arr[1] = 9;

Queue<type>::arr[2] = 30;

Queue<type>::arr[3] = 45;

Queue<type>::arr[4] = 50;

Queue<type>::arr[5] = 100;

Queue<type>::arr[6] = 56;

}

virtual void enQueue(type value)

{

this->arr[this->currentSize] = value;

this->currentSize++;

}

virtual type deQueue()

{

if (this->currentSize == 0) {

cout << "The Queue is empty!!!"<<endl;

}

type t = this->arr[0];

for (int i = 0; i < this->currentSize - 1; i++)

{

this->arr[i] = this->arr[i + 1];

}

this->currentSize--;

return t;

}

int getCurrentSize()

{

return this->currentSize;

}

void display()

{

cout << "Queue : ";

for (int i = 0; i < this->currentSize; i++)

{

cout << this->arr[i] << " ";

}

cout << endl;

}

};

int main()

{

myQueue <int> obj(SIZE);

obj.enQueue(45);

obj.enQueue(100);

obj.enQueue(95);

obj.enQueue(78);

obj.enQueue(88);

obj.enQueue(35);

obj.enQueue(36);

bool found = false;

while (obj.getCurrentSize() > 0)

{

if (obj.deQueue() == 88)

{

found = true;

break;

}

}

if (!found)

{

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

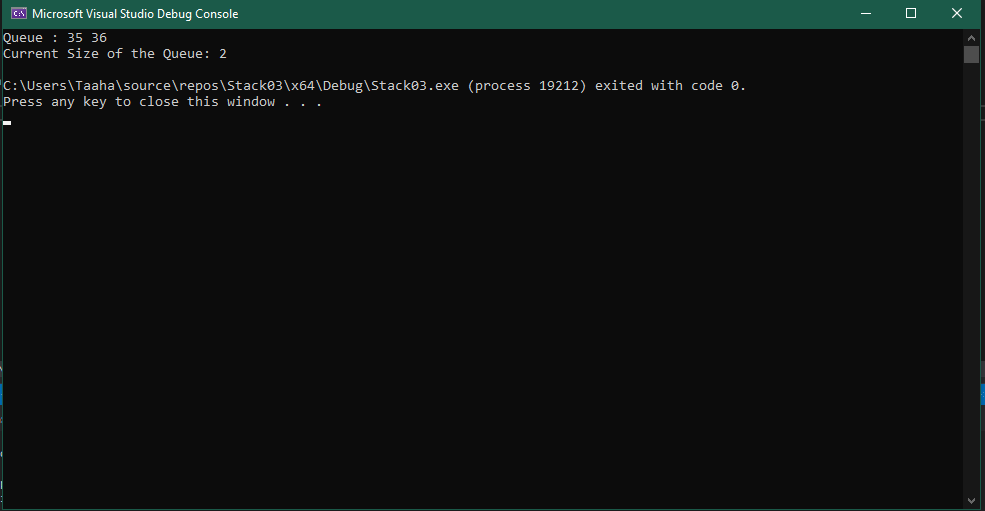
}

obj.display();

cout << "Current Size of the Queue: " << obj.getCurrentSize() << endl;

return 0;

}



**Task 3**

#include<iostream>

using namespace std;

template <class type>

class Stack {

protected:

int top;

int maximumSize;

type\* arr;

public:

Stack() {

maximumSize = 30;

top = -1;

arr = new type[maximumSize];

}

Stack(int s) {

maximumSize = s;

top = -1;

arr = new type[s];

}

virtual bool isEmpty() = 0;

virtual void push(type value) = 0;

virtual type pop() = 0;

virtual void display() = 0;

};

template <class type>

class myStack : public Stack<type> {

public:

myStack() : Stack<type>() {

}

myStack(int s) : Stack<type>(s) {

this->push(25);

this->push(27);

this->push(30);

}

virtual bool isEmpty() {

return this->top == -1;

}

virtual void push(type value) {

if (this->top == this->maximumSize - 1) {

cout << "Stack Overflow" << endl;

return;

}

this->top++;

this->arr[this->top] = value;

}

virtual type pop() {

if (this->top == -1) {

cout << "Stack Underflow" << endl;

return -1;

}

type temp = this->arr[this->top];

this->top--;

return temp;

}

virtual void display() {

cout << "Stack: " << endl;

for (int i = 0; i <= this->top; i++) {

cout << this->arr[i] << endl;

}

}

};

int main() {

myStack<int> obj(10);

obj.push(90);

obj.push(88);

obj.push(100);

for (int i = 0; i < 10; i++) {

if (obj.pop() == 27) {

break;

}

}

obj.display();

return 0;

}

