**LAB MID**

|  |  |
| --- | --- |
| **Name:** | **Saira Sana** |
| **Registration No:** | **SP22-BCS-069** |
| **Section:** | **B** |
| **Submitted To:** | **Mam Yasmeen** |

# COMSATS university Islamabad Vehari Campus

**Activity N0 01:**

**Code:**

#include <iostream>

#include <unordered\_set>

class Node {

public:

int data;

Node\* next;

Node(int value) : data(value), next(nullptr) {}

};

class LinkedList {

public:

Node\* head;

LinkedList() : head(nullptr) {}

void insert(int value) {

Node\* newNode = new Node(value);

if (head == nullptr) {

head = newNode;

} else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

void removeDuplicates() {

std::unordered\_set<int> uniqueValues;

Node\* current = head;

Node\* prev = nullptr;

while (current != nullptr) {

if (uniqueValues.find(current->data) != uniqueValues.end()) {

prev->next = current->next;

delete current;

current = prev->next;

} else {

uniqueValues.insert(current->data);

prev = current;

current = current->next;

}

}

}

void display() {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

std::cout << std::endl;

}

};

int main() {

LinkedList list;

list.insert(1);

list.insert(2);

list.insert(3);

list.insert(2);

list.insert(4);

list.insert(3);

list.insert(5);

std::cout << "Original Linked List: ";

list.display();

list.removeDuplicates();

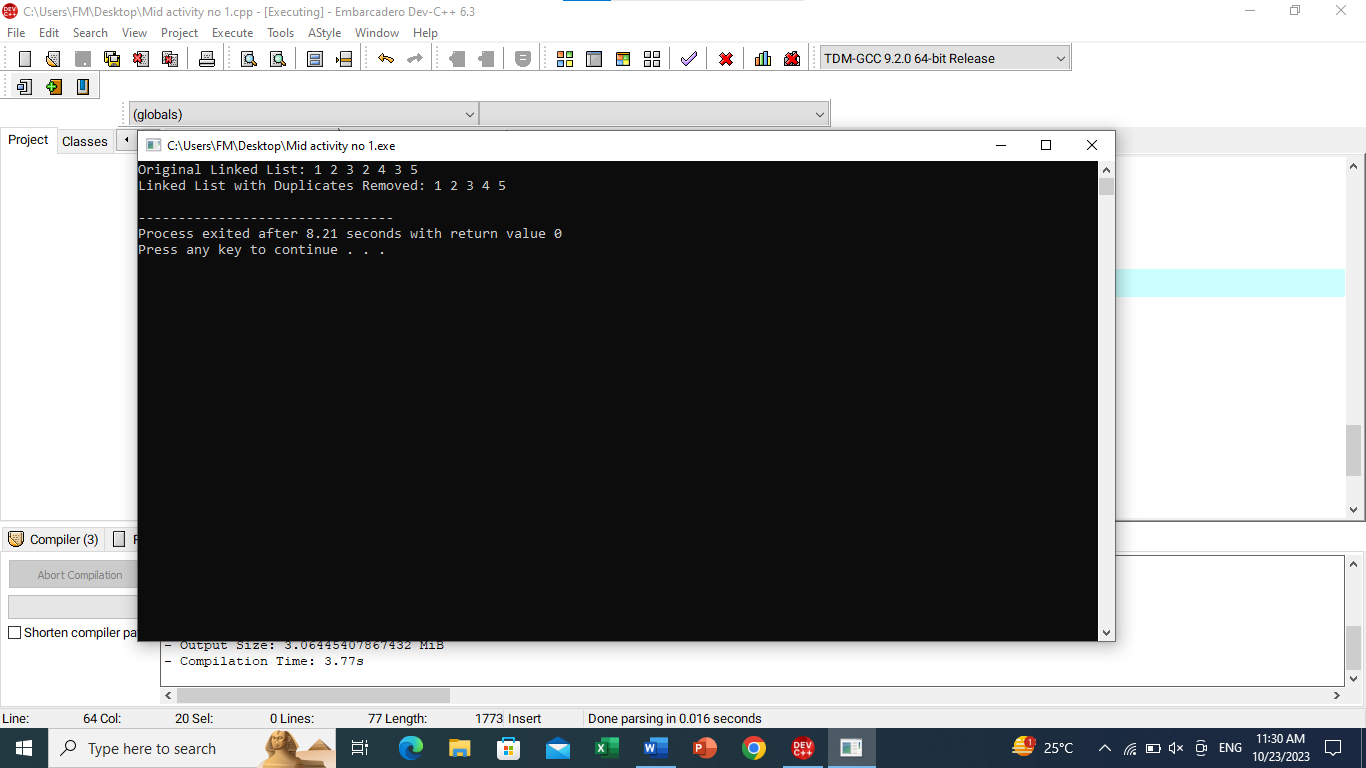
std::cout << "Linked List with Duplicates Removed: ";

list.display();

return 0;

}

**Image:**

****

**Activity No 02:**

**Code:**

#include<iostream>

using namespace std;

class Node {

private:

int data;

Node \*next;

public:

Node \*front,\*rear=NULL;

void enqueue(int x){

Node \*p=new Node;

p->data=x;

p->next=NULL;

if(front==NULL || rear==NULL){

front=p;

rear=p;

cout<<"\nThe inserted element in queue is: \n"<<rear->data;

}

else{

rear->next=p;

rear=p;

cout<<"\nThe inserted element in queue is: \n"<<rear->data;

}

}

/\*void dequeue() {

if (front == NULL) {

cout << "\nEmpty queue. Cannot dequeue." << endl;

} else {

Node \*d = front;

cout << "\nThe dequeued element is: " << d->data << endl;

front = front->next;

delete d;

}

}\*/

void dequeue(){

Node \*d=new Node();

d=front;

if(d==NULL)

{

cout<<"\nEmpty queue";

}

else{

//if(d==NULL)

cout<<"\nThe dequeue elements is: ";

cout<<d->data;

front=front->next;

delete d;

d=NULL;

}

}

/\* else{

cout<<"\nThe dequeue elements is: \n";

cout<<front->data;

front=front->next;

delete d;

d=NULL;

}

}

}\*/

void display() {

Node \*temp = front;

cout << "\nThe queue elements are: ";

if (temp == NULL) {

cout << "empty";

}

while (temp != NULL) {

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

temp = temp->next;

}

}

};

int main(){

Node i;

i.enqueue(1);

i.enqueue(2);

i.display();

i.dequeue();

i.display();

i.enqueue(3);

i.enqueue(4);

i.display();

i.dequeue();

i.dequeue();

i.dequeue();

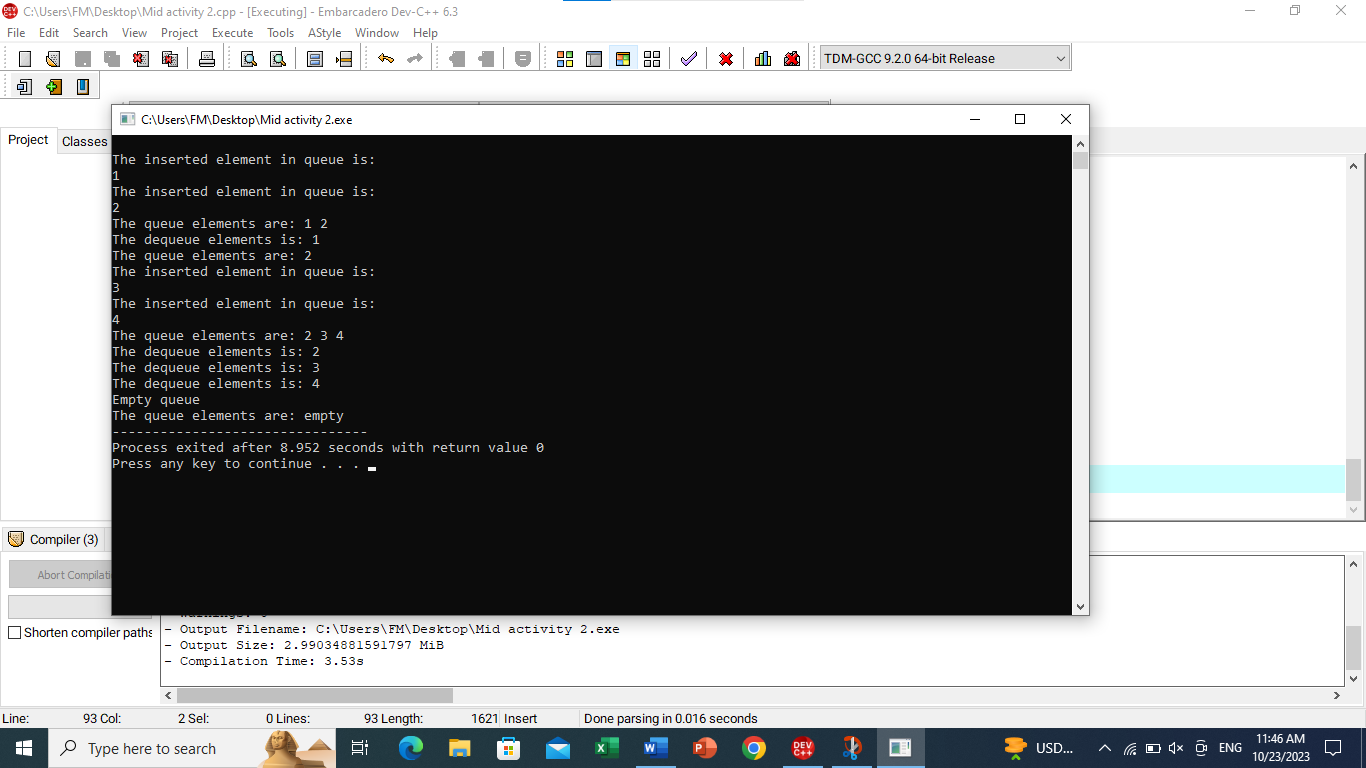
i.dequeue();

i.display();

return 0;

}

**Image:**

****