|  |  |
| --- | --- |
| download | COMSATS University Islamabad, Vehari Campus Department of Computer Science |

**Class: BCS-SP22-4A**

**Subject: Data Structures and Algorithms-Lab**

**Name: Mufeez Aslam**

**Instructor: Yasmeen Jana**

**Reg. No: SP22-BCS-035**

Mid Term Exam

**Question No #02:**

Stack using array

**Code:**

#include <iostream>

class Node {

public:

int data;

Node\* next;

Node(int value) {

data = value;

next = NULL;

}

};

class Stack {

private:

Node\* top;

public:

Stack() {

top = NULL;

}

// Push an element onto the stack

void push(int value) {

Node\* newNode = new Node(value);

newNode->next = top;

top = newNode;

std::cout << value << " pushed onto the stack." << std::endl;

}

// Pop an element from the stack

void pop() {

if (isEmpty()) {

std::cout << "Stack is empty. Cannot pop." << std::endl;

} else {

int value = top->data;

Node\* temp = top;

top = top->next;

delete temp;

std::cout << value << " popped from the stack." << std::endl;

}

}

// Check if the stack is empty

bool isEmpty() {

return top == NULL;

}

// Display the elements in the stack

void display() {

if (isEmpty()) {

std::cout << "Stack is empty." << std::endl;

} else {

Node\* current = top;

std::cout << "Stack elements: ";

while (current != NULL) {

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

current = current->next;

}

std::cout << std::endl;

}

}

};

int main() {

Stack stack;

int choice, value;

while (true) {

std::cout << "Stack Operations:" << std::endl;

std::cout << "1. Push" << std::endl;

std::cout << "2. Pop" << std::endl;

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

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

std::cout << "Enter your choice: ";

std::cin >> choice;

switch (choice) {

case 1:

std::cout << "Enter value to push: ";

std::cin >> value;

stack.push(value);

break;

case 2:

stack.pop();

break;

case 3:

stack.display();

break;

case 4:

exit(0);

default:

std::cout << "Invalid choice. Please try again." << std::endl;

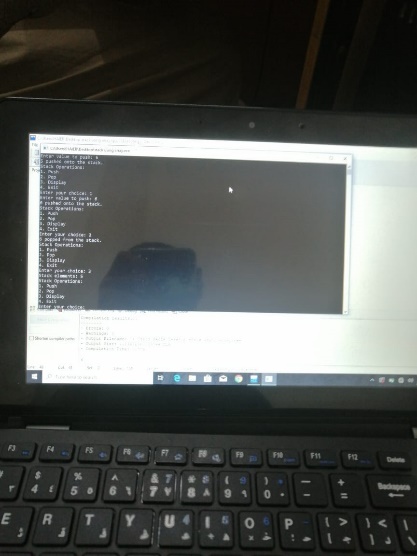
}

}

return 0;

}

**Output:**

****

**Question No #01:**

#include <iostream>

class Node {

public:

int data;

Node\* next;

Node(int data) {

this->data = data;

this->next = NULL;

}

};

class LinkedList {

public:

Node\* head;

LinkedList() {

head = NULL;

}

void addNode(int data) {

Node\* newNode = new Node(data);

newNode->next = head;

head = newNode;

}

bool isPalindrome() {

if (head == NULL) {

return true; // An empty list is considered a palindrome.

}

// Implement a custom stack using a singly linked list

Node\* stackTop = NULL;

Node\* current = head;

int length = 0;

Node\* slow = head;

while (current) {

length++;

current = current->next;

if (length % 2 == 0) {

if (length > 2) {

Node\* temp = slow->next;

slow->next = stackTop;

stackTop = slow;

slow = temp;

}

}

}

// Step 2: Compare the second half with the elements in the stack

if (length % 2 == 0) {

slow = slow->next;

}

while (slow) {

if (slow->data != stackTop->data) {

return false;

}

slow = slow->next;

stackTop = stackTop->next;

}

return true;

}

};

int main() {

LinkedList ll;

ll.addNode(1);

ll.addNode(2);

ll.addNode(2);

ll.addNode(1);

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

Node\* current = ll.head;

while (current != NULL) {

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

current = current->next;

}

std::cout << "NULL" << std::endl;

if (ll.isPalindrome()) {

std::cout << "The linked list is a palindrome." << std::endl;

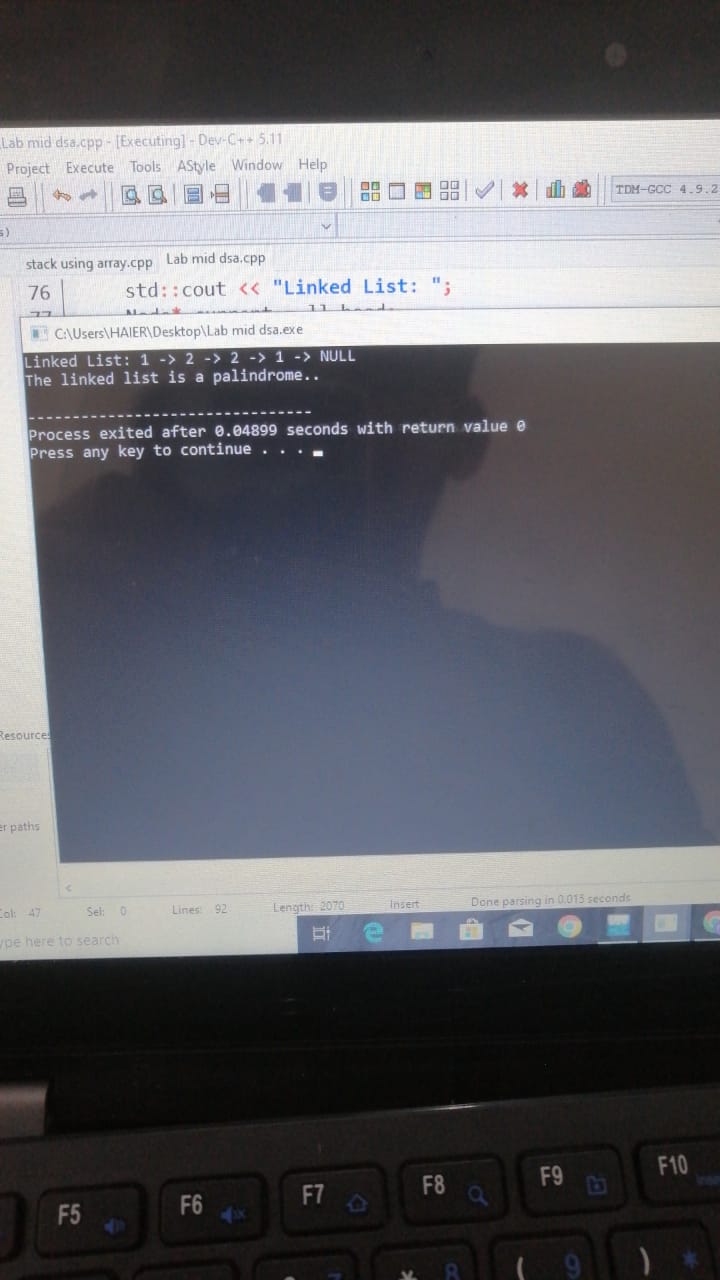
} else {

std::cout << "The linked list is not a palindrome." << std::endl;

}

return 0;

}

****