|  |  |
| --- | --- |
| **MID TERM** | |
| **Name** | **M . Hassan** |
| **Registration No** | **SP22-BCS-045** |
| **Section** | **A** |
| **Subject** | **Data Structure lab** |
| **Submitted to** | **Mam Yasmeen Jana** |
| **Date** | **25/10/2023** |
| **Comsats University Vehari** | |

**Question no 1:**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

Node(int data) : data(data), next(NULL) {}

};

void insert(Node\*& head, int data) {

Node\* newNode = new Node(data);

if (head == NULL) {

head = newNode;

} else {

Node\* current = head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

void printList(Node\* head) {

Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

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

}

Node\* reverseList(Node\* head) {

Node\* prev = NULL;

Node\* current = head;

Node\* next;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

return prev;

}

bool isPalindrome(Node\* head) {

if (head == NULL || head->next == NULL) {

return true;

}

Node\* slow = head;

Node\* fast = head;

while (fast->next != NULL && fast->next->next != NULL) {

slow = slow->next;

fast = fast->next->next;

}

Node\* secondHalf = reverseList(slow->next);

Node\* firstHalf = head;

while (secondHalf != NULL) {

if (firstHalf->data != secondHalf->data) {

return false;

}

firstHalf = firstHalf->next;

secondHalf = secondHalf->next;

}

return true;

}

int main() {

Node\* head = NULL;

insert(head, 1);

insert(head, 2);

insert(head, 3);

insert(head, 2);

insert(head, 1);

cout << "Linked List: ";

printList(head);

if (isPalindrome(head)) {

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

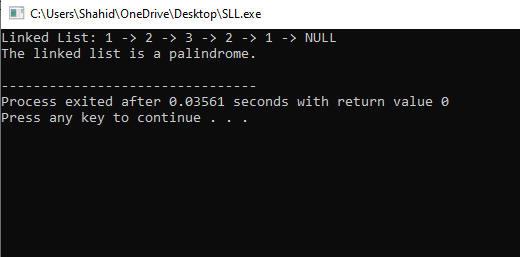
} else {

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

}

return 0;

}



**Question no 2:**

#include <iostream>

using namespace std;

class Stack {

private:

int\* arr;

int capacity;

int top;

public:

Stack(int size) {

capacity = size;

arr = new int[capacity];

top = -1;

}

~Stack() {

delete[] arr;

}

bool isEmpty() {

return top == -1;

}

bool isFull() {

return top == capacity - 1;

}

void push(int value) {

if (isFull()) {

cout << "Stack overflow. Cannot push " << value << endl;

return;

}

arr[++top] = value;

cout << "Pushed " << value << " into the stack." << endl;

}

void pop() {

if (isEmpty()) {

cout << "Stack underflow." << endl;

return;

}

int value = arr[top--];

cout << "Popped " << value << " from the stack." << endl;

}

void display() {

if (isEmpty()) {

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

return;

}

cout << "Stack: ";

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

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

}

cout << endl;

}

};

int main() {

int size;

cout << "Enter the size of the stack: ";

cin >> size;

Stack stack(size);

int choice, value;

while (true) {

cout << "Enter your choice:\n";

cout << "1. Push\n";

cout << "2. Pop\n";

cout << "3. Display\n";

cout << "4. Quit\n";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter a value to push onto the stack: ";

cin >> value;

stack.push(value);

break;

case 2:

stack.pop();

break;

case 3:

stack.display();

break;

case 4:

return 0;

default:

cout << "Invalid choice. Please try again.\n";

}

}

}

OUTPUT:

