Name :- Saurabh Arun Deulkar

Mis Id :- 612203038

Div :- 1

Batch :- T4

Lab 1: Implementation of Stack ADT and Queue ADT

Q1). A) Use the stack abstract data type to solve the Parenthesis Matching Problem, where the input is the string containing parentheses ((, ), {, }, [, ]), and the goal is to determine if the parentheses are balanced.

Code :-

#include <iostream>

#include <string>

// implement the stack using the linked list

class Stack {

private:

struct Node {

char data;

Node\* next;

};

Node\* top;

public:

Stack() {

top = NULL;

}

void push(char data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = top;

top = newNode;

}

char pop() {

if (top == NULL) {

return '\0';

}

Node\* temp = top;

top = top->next;

char data = temp->data;

delete temp;

return data;

}

char peek() {

if (top == NULL) {

return '\0';

}

return top->data;

}

bool isEmpty() {

return top == NULL;

}

};

using namespace std;

bool isMatched(string str) {

// consider string consisting of {}[]() characters

Stack stack;

for (int i = 0; i < str.length(); i++) {

char ch = str[i];

if (ch == '{' || ch == '[' || ch == '(') {

stack.push(ch);

} else if (ch == '}' || ch == ']' || ch == ')') {

if (stack.isEmpty()) {

return false;

}

char top = stack.pop();

if ((ch == '}' && top != '{') || (ch == ']' && top != '[') || (ch == ')' && top != '(')) {

return false;

}

}

}

return stack.isEmpty();

}

int main() {

string str;

cout << "Enter the string of parenthesis: ";

cin >> str;

if(isMatched(str)) {

cout << "The parenthesis are matched." << endl;

} else {

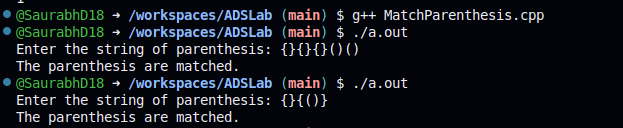
cout << "The parenthesis are not matched." << endl;

}

return 0;

}

Output : -



B) Name the data structure used to solve the Parenthesis Matching Problem.

:- Stack

Q2). A) Design a ticket booking system where customers arrive to book tickets and join a queue. The system should process customers in a first-come, first-served (FIFO) manner. The queue must provide the following operations (options) to the user:

1. Add Customer -> Add a new customer to the booking queue.

2. Process Booking -> Process the ticket booking for the current customer in line and remove them from the queue after processing.

3. View Queue -> Display the current list of customers waiting in the queue.

The system should simulate real-life ticket booking, where customers are served in the order of arrival.

Code :-

#include <iostream>

#include <string>

using namespace std;

class Node {

public:

string data;

Node\* next;

Node(string data) {

this->data = data;

this->next = NULL;

}

};

class Queue {

private:

Node\* front;

Node\* rear;

public:

Queue() {

front = NULL;

rear = NULL;

}

void enqueue(string data) {

Node\* newNode = new Node(data);

if (rear == NULL) {

front = newNode;

rear = newNode;

} else {

rear->next = newNode;

rear = newNode;

}

}

string dequeue() {

if (front == NULL) {

return "";

}

Node\* temp = front;

front = front->next;

string data = temp->data;

delete temp;

return data;

}

void display() {

Node\* temp = front;

while (temp != NULL) {

cout << temp->data;

if (temp->next != NULL) {

cout << "-> ";

}

temp = temp->next;

}

cout << endl;

}

};

int main() {

Queue queue;

int choice;

string name;

do {

cout << "--- Welcome to ABC Ticket Booking System ---" << endl;

cout << "1. Add Customer" << endl;

cout << "2. Process Booking" << endl;

cout << "3. View Queue" << endl;

cout << "Enter Your Choice (1-3): ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter Customer Name: ";

cin >> name;

queue.enqueue(name);

cout << "Customer " << name << " has joined the queue." << endl;

break;

case 2:

name = queue.dequeue();

if (name == "") {

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

} else {

cout << "Booking processed for customer: " << name << endl;

}

break;

case 3:

cout << "Current Queue: ";

queue.display();

break;

default:

cout << "Invalid choice. Please enter a valid choice (1-3)." << endl;

}

} while (choice != 0);

return 0;

}

Output :-



B) Name the data structure used for the ticket booking system.

Ans :- Queue