Aim- implementing queue using stack

#include <iostream>

#include <stack>

using namespace std;

class Queue {

private:

stack<int> s1, s2;

public:

// Enqueue operation

void enqueue(int value) {

s1.push(value);

cout << value << " enqueued successfully.\n";

}

// Dequeue operation

void dequeue() {

if (s1.empty() && s2.empty()) {

cout << "Queue is empty. Cannot dequeue.\n";

return;

}

if (s2.empty()) {

while (!s1.empty()) {

s2.push(s1.top());

s1.pop();

}

}

cout << s2.top() << " dequeued successfully.\n";

s2.pop();

}

// Check if the queue is empty

bool isEmpty() {

return s1.empty() && s2.empty();

}

// Display queue elements

void display() {

if (isEmpty()) {

cout << "Queue is empty.\n";

return;

}

stack<int> temp1 = s1, temp2;

while (!temp1.empty()) {

temp2.push(temp1.top());

temp1.pop();

}

cout << "Queue elements: ";

while (!temp2.empty()) {

cout << temp2.top() << " ";

temp2.pop();

}

temp2 = s2;

while (!temp2.empty()) {

cout << temp2.top() << " ";

temp2.pop();

}

cout << "\n";

}

};

int main() {

Queue q;

int choice, value;

do {

cout << "\n1. Enqueue\n2. Dequeue\n3. Check if Empty\n4. Display\n5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to enqueue: ";

cin >> value;

q.enqueue(value);

break;

case 2:

q.dequeue();

break;

case 3:

if (q.isEmpty())

cout << "Queue is empty.\n";

else

cout << "Queue is not empty.\n";

break;

case 4:

q.display();

break;

case 5:

cout << "Exiting program.\n";

break;

default:

cout << "Invalid choice. Try again.\n";

}

} while (choice != 5);

return 0;

}