**DAY – 14 Assignment and LABs**

**// Assignment - 1**

**/\* Problem 1: List Operations**

**Description:**

**Write a program that uses the list container to manage a collection of integers. Your program should perform the following operations:**

**Insert elements at the front and back of the list.**

**Remove elements from the front and back of the list.**

**Sort the list in ascending and descending order.**

**Reverse the list.**

**Display the elements of the list. \*/**

**// Code a :**

/\* #include <iostream>

#include <list>

#include <algorithm>

using namespace std;

void displayList(const list<int>& lst) {

for (int num : lst) { // Range Based loop

cout << num << " ";

}

cout << endl;

}

int main() {

list<int> myList;

// Insert elements at the front and back of the list

myList.push\_back(10);

myList.push\_front(20);

myList.push\_back(30);

myList.push\_front(40);

cout << "List after inserts: ";

displayList(myList);

// Remove elements from the front and back of the list

myList.pop\_front();

myList.pop\_back();

cout << "List after pops: ";

displayList(myList);

// Sort the list in ascending order

myList.sort();

cout << "List sorted in ascending order: ";

displayList(myList);

// Sort the list in descending order

myList.sort(greater<int>());

cout << "List sorted in descending order: ";

displayList(myList);

// Reverse the list

myList.reverse();

cout << "List after reversing: ";

displayList(myList);

return 0;

} \*/

**// Another code for List API's**

**// Code b:**

/\* #include <iostream>

#include <list>

int main() {

// Create a list

std::list<int> myList;

// Insert elements at the end

myList.push\_back(10);

myList.push\_back(20);

myList.push\_back(30);

// Insert elements at the front

myList.push\_front(5);

myList.push\_front(1);

// Display elements

std::cout << "List after push\_back and push\_front: ";

for (int val : myList) {

std::cout << val << " ";

}

std::cout << std::endl;

// Insert element at a specific position

auto it = myList.begin();

std::advance(it, 2);

myList.insert(it, 15);

std::cout << "List after insert: ";

for (int val : myList) {

std::cout << val << " ";

}

std::cout << std::endl;

// Erase element at a specific position

it = myList.begin();

std::advance(it, 3);

myList.erase(it);

std::cout << "List after erase: ";

for (int val : myList) {

std::cout << val << " ";

}

std::cout << std::endl;

// Remove elements by value

myList.remove(10);

std::cout << "List after remove: ";

for (int val : myList) {

std::cout << val << " ";

}

std::cout << std::endl;

// Remove elements based on a condition

myList.remove\_if([](int n) { return n < 10; });

std::cout << "List after remove\_if: ";

for (int val : myList) {

std::cout << val << " ";

}

std::cout << std::endl;

// Sorting the list

myList.sort();

std::cout << "List after sort: ";

for (int val : myList) {

std::cout << val << " ";

}

std::cout << std::endl;

// Reversing the list

myList.reverse();

std::cout << "List after reverse: ";

for (int val : myList) {

std::cout << val << " ";

}

std::cout << std::endl;

// Merging two lists

std::list<int> otherList = {40, 50, 60};

myList.merge(otherList);

std::cout << "List after merge: ";

for (int val : myList) {

std::cout << val << " ";

}

std::cout << std::endl;

// Clearing the list

myList.clear();

std::cout << "List after clear: ";

for (int val : myList) {

std::cout << val << " ";

}

std::cout << std::endl;

// Checking if the list is empty

if (myList.empty()) {

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

}

// Adding elements again

myList.push\_back(100);

myList.push\_back(200);

// Accessing front and back elements

std::cout << "Front element: " << myList.front() << std::endl;

std::cout << "Back element: " << myList.back() << std::endl;

return 0;

} \*/

**/\* Problem 2: Vector Manipulation**

**Description:**

**Create a program that uses the vector container to store a collection of floating-point numbers. The program should:**

**Add elements to the vector.**

**Remove elements from a specified position.**

**Find the maximum and minimum elements in the vector.**

**Calculate the average of the elements.**

**Display the elements of the vector. \*/**

/\* #include <iostream>

#include <vector>

#include <algorithm>

#include <numeric>

using namespace std;

void displayVector(const vector<float>& vec) {

for (float num : vec) {

cout << num << " ";

}

cout <<endl;

}

int main() {

vector<float> myVector;

// Add elements to the vector

myVector.push\_back(10.5);

myVector.push\_back(20.3);

myVector.push\_back(30.2);

// myVector.push\_back(15.8);

cout << "Vector after adds: ";

displayVector(myVector);

// Remove element from a specified position

if (!myVector.empty() && myVector.size() > 1) {

myVector.erase(myVector.begin());

}

cout << "Vector after removal: ";

displayVector(myVector);

auto maxElement = \*max\_element(myVector.begin(), myVector.end());

auto minElement = \*min\_element(myVector.begin(), myVector.end());

cout << "Max element: " << maxElement << endl;

cout << "Min element: " << minElement << endl;

// Calculate the average of the elements

float sum = accumulate(myVector.begin(), myVector.end(), 0.0f);

float average = sum / myVector.size();

std::cout << "Average: " << average << std::endl;

cout << "Final vector elements are : ";

displayVector(myVector);

return 0;

} \*/

**/\* Problem 3: Queue Simulation**

**Description:**

**Implement a program using the queue container to simulate a ticketing system.**

**The program should:**

**Add customers to the queue.**

**Serve customers (remove from front of the queue).**

**Display the current queue.**

**Display the number of customers served. \*/**

/\*#include <iostream>

#include <queue>

void displayQueue(std::queue<std::string> q) {

while (!q.empty()) {

std::cout << q.front() << " ";

q.pop();

}

std::cout << std::endl;

}

int main() {

std::queue<std::string> ticketQueue;

int customersServed = 0;

// Add customers to the queue

ticketQueue.push("Customer1");

ticketQueue.push("Customer2");

ticketQueue.push("Customer3");

std::cout << "Current queue: ";

displayQueue(ticketQueue);

// Serve customers (remove from front of the queue)

while (!ticketQueue.empty()) {

std::cout << "Serving: " << ticketQueue.front() << std::endl;

ticketQueue.pop();

customersServed++;

}

std::cout << "Number of customers served: " << customersServed << std::endl;

return 0;

} \*/

**/\* Problem 4: Stack Operations**

**Description:**

**Write a program using the stack container to evaluate a postfix expression. The program should:**

**Read a postfix expression.**

**Use a stack to evaluate the expression.**

**Display the result of the evaluation. \*/**

/\* #include <iostream>

#include <stack>

#include <sstream>

#include <string>

int evaluatePostfix(const std::string& expression) {

std::stack<int> stack;

std::istringstream tokens(expression);

std::string token;

while (tokens >> token) {

if (isdigit(token[0])) {

stack.push(std::stoi(token));

} else {

int op2 = stack.top();

stack.pop();

int op1 = stack.top();

stack.pop();

if (token == "+") {

stack.push(op1 + op2);

} else if (token == "-") {

stack.push(op1 - op2);

} else if (token == "\*") {

stack.push(op1 \* op2);

} else if (token == "/") {

stack.push(op1 / op2);

}

}

}

return stack.top();

}

int main() {

std::string postfixExpression = "5 9 8 + 4 6 \* \* 7 + \*";

int result = evaluatePostfix(postfixExpression);

std::cout << "Result of postfix evaluation: " << result << std::endl;

return 0;

} \*/

**// Code - 1 Program to find maxElement and minElement in a Vector.**

/\* #include <iostream>

#include <vector>

#include <algorithm>

int main() {

vector<float> myVector = {10.5, 20.3, 30.2, 5.1, 25.8};

auto maxElement = \*max\_element(myVector.begin(), myVector.end());

auto minElement = \*min\_element(myVector.begin(), myVector.end());

cout << "Max element: " << maxElement << endl;

cout << "Min element: " << minElement << endl;

return 0;

} \*/

**// (\*\*\* Important \*\*\*) Implementation of code on File Handling**

#include <iostream>

#include <fstream> // File input / output stream

#include <string>

using namespace std;

int main() {

string fileName;

char choice;

// Get file name from user

cout << "Enter the file name: ";

cin >> fileName;

// Get user's choice for operation (read or write)

cout << "**Enter '1' to read from the file or '2' to write to the file:** ";

cin >> choice;

if (choice == '1') {

// Open the file in read mode

ifstream inputFile(fileName);

if (inputFile.is\_open()) {

string line;

// Read data from the file and print line by line

while (getline(inputFile, line)) {

cout << line << endl;

}

inputFile.close();

} else {

cout << "Error opening file for reading." << endl;

}

} else if (choice == '2') {

// Open the file in write mode (truncates existing content)

ofstream outputFile(fileName);

if (outputFile.is\_open()) {

string content;

// Get content from user to write to the file

cout << "**Enter the content to write to the file:** ";

getline(cin, content, '\n'); // Include newline character

outputFile << content << endl;

outputFile.close();

cout << "**Content written to the file successfully."** << endl;

} else {

cout << "**Error opening file for writing**." << endl;

}

} else {

cout << "**Invalid choice. Please enter '1' or '2'."** << endl;

}

return 0;

}

