Problem 1: List Operations

Description:

Write a program that uses the std::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.

```
#include <iostream>

#include <list>

#include <algorithm>

using namespace std;

void displayList(const list<int>& lst) {                    // Function to display elements of the list

    for (int element : lst) {

        cout << element << " ";

    }

    cout << endl;

}

int main() {

    list<int> myList;

    myList.push_front(10);                                  // Inserting elements at the front and back of the list

    myList.push_back(20);

    myList.push_front(5);

    myList.push_back(30);
```

```
    cout << "List after inserting elements: ";
    displayList(myList);
    myList.pop_front();
    myList.pop_back();
    cout << "List after removing elements: ";
    displayList(myList);
    myList.sort();
    cout << "List after sorting in ascending order: ";
    displayList(myList);
    myList.sort(greater<int>());
    cout << "List after sorting in descending order: ";
    displayList(myList);
    myList.reverse();                    // Reversing the list
    cout << "List after reversing: ";
    displayList(myList);
    return 0;
}
```

Problem 2: Vector Manipulation

Description:

Create a program that uses the std::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 element : vec) {
		cout << element << " ";
	}
	cout << endl;
}
int main() {
	vector<float> myVector;
	myVector.push_back(1.1);			// Adding elements to the vector
	myVector.push_back(2.2);
	myVector.push_back(3.3);
	myVector.push_back(4.4);
	myVector.push_back(5.5);
	cout << "Vector after adding elements: ";
	displayVector(myVector);
	int positionToRemove = 2;					// Removing element from a specified position
	if (positionToRemove >= 0 && positionToRemove < myVector.size()) {
		myVector.erase(myVector.begin() + positionToRemove);
	}
	cout << "Vector after removing element at position " << positionToRemove << ": ";
```

```
    displayVector(myVector);
    if (!myVector.empty()) {
        float maxElement = *max_element(myVector.begin(), myVector.end());
        float minElement = *min_element(myVector.begin(), myVector.end());
        cout << "Maximum element: " << maxElement << endl;
        cout << "Minimum element: " << minElement << endl;
    } else {
        cout << "Vector is empty, cannot find maximum and minimum elements." << endl;
    }
    if (!myVector.empty()) {
        float sum = accumulate(myVector.begin(), myVector.end(), 0.0f);
        float average = sum / myVector.size();
        cout << "Average of elements: " << average << endl;
    } else {
        cout << "Vector is empty, cannot calculate average." << endl;
    }
    cout << "Final elements of the vector: ";
    displayVector(myVector);
    return 0;
}
```

Problem 3: Queue Simulation

Description:

Implement a program using the std::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>
using namespace std;
void displayQueue(queue<string> q) {
	while (!q.empty()) {
		cout << q.front() << " ";
		q.pop();
	}
	cout << endl;
}
int main() {
	queue<string> ticketQueue;
	int customersServed = 0;
	ticketQueue.push("Customer 1");                                    // Adding customers to the queue
	ticketQueue.push("Customer 2");
	ticketQueue.push("Customer 3");
	ticketQueue.push("Customer 4");
	cout << "Queue after adding customers: ";
	displayQueue(ticketQueue);
	while (!ticketQueue.empty()) {
		cout << "Serving: " << ticketQueue.front() << endl;
		ticketQueue.pop();
		customersServed++;
		cout << "Current queue: ";
		displayQueue(ticketQueue);
	}
```

```
    cout << "Number of customers served: " << customersServed << endl;
    return 0;
}
```

OUTPUT :-

Problem 4: Stack Operations

Description:

Write a program using the std::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>
using namespace std;
double evaluatePostfix(const string& expression) {                    // Function to evaluate a postfix expression
    stack<double> stack;
    stringstream ss(expression);
    string token;
    while (ss >> token) {
        if (isdigit(token[0]) || (token[0] == '-' && token.length() > 1)) {
            stack.push(stod(token));
        } else {
```

```
            double operand2 = stack.top();
            stack.pop();
            double operand1 = stack.top();
            stack.pop();
            if (token == "+") {                                                    // Perform
the operation and push the result back to the stack
                stack.push(operand1 + operand2);
            } else if (token == "-") {
                stack.push(operand1 - operand2);
            } else if (token == "*") {
                stack.push(operand1 * operand2);
            } else if (token == "/") {
                stack.push(operand1 / operand2);
            } else {
                throw invalid_argument("Invalid operator in postfix expression.");
            }
        }
    }
    if (stack.size() != 1) {                                          // The result is the remaining
element in the stack
        throw invalid_argument("Invalid postfix expression.");
    }

    return stack.top();
}
int main() {
```

```
    string postfixExpression;
    cout << "Enter a postfix expression (tokens separated by spaces): ";
    getline(cin, postfixExpression);
    try {
        double result = evaluatePostfix(postfixExpression);
        cout << "Result of the evaluation: " << result << endl;
    } catch (const invalid_argument& e) {
        cout << "Error: " << e.what() << endl;
    }
    return 0;
}
```

FILE OPERATIONS :-

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
    string fileName;
    char choice;
    cout << "Enter the file name: ";                    // Get file name from user
    cin >> fileName;
    cout << "Enter 'r' to read from the file or 'w' to write to the file: ";                    //
Get user's choice for operation (read or write)
    cin >> choice;
    if (choice == 'r') {
```

```
        ifstream inputFile(fileName);                                   // Open the file in read mode
        if (inputFile.is_open()) {
            string line;
            while (getline(inputFile, line)) {                          // Read data from the file
and print line by line
                cout << line << endl;
            }
            inputFile.close();
        } else {
            cout << "Error opening file for reading." << endl;
        }
    } else if (choice == 'w') {
        ofstream outputFile(fileName);                                  // Open the file in write
mode (truncates existing content)
        if (outputFile.is_open()) {
            string content;
            cout << "Enter the content to write to the file: ";                 // Get
content from user 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 'r' or 'w'." << endl;
```

```
    }

    return 0;

}
```

OUTPUT :-