TASK 1

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {

    vector<int> nums = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    sort(nums.begin(), nums.end());

    cout << "Sorted vector: ";
    for (int num : nums) cout << num << " ";
    cout << endl;

    auto minIt = min_element(nums.begin(), nums.end());
    cout << "Min element: " << *minIt << endl;


    auto maxIt = max_element(nums.begin(), nums.end());
    cout << "Max element: " << *maxIt << endl;

    auto last = unique(nums.begin(), nums.end());
    nums.erase(last, nums.end());

    cout << "Vector after removing duplicates: ";
    for (int num : nums) cout << num << " ";
    cout << endl;

    return 0;
}
```

TASK 2

```
#include <iostream>
#include <set>
using namespace std;

int main() {
    int num;
    set<int> nums = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

    cout << "Enter a number to check: ";
    cin >> num;

    if (nums.find(num) != nums.end()) {
        cout << num << " exists in the set.\n";
    } else {
        cout << num << " does not exist in the set.\n";
    }
```

```
    cout << "Set elements: ";
    for (int number : nums) {
        cout << number << " ";
    }
    cout << endl;

    return 0;
}
```

TASK 3

```
#include <iostream>
#include <map>
using namespace std;

int main() {
    string text = "this is a test this is only a test";
    map<string, int> wordCount;

    string word = "";
    for (char ch : text) {
        if (ch != ' ') {
            word += ch;
        } else {
            if (!word.empty()) {
                wordCount[word]++;
                word = "";
            }
        }
    }


    if (!word.empty()) {
        wordCount[word]++;
    }

    for (const auto& pair : wordCount) {
        cout << pair.first << " -> " << pair.second << endl;
    }

    return 0;
}
```

TASK 4

```
#include <iostream>
#include <stack>
```

```
using namespace std;

int main() {
    string expr = "((a+b)*(c-d))";
    stack<char> s;
    bool balanced = true;

    for (char ch : expr) {
        if (ch == '(') {
            s.push(ch);
        } else if (ch == ')') {
            if (s.empty()) {
                balanced = false;
                break;
            } else {
                s.pop();
            }
        }
    }

    if (balanced && s.empty()) {
        cout << "Correct" << endl;
    } else {
        cout << "Incorrect" << endl;
    }

    return 0;
}
```

TASK 5

```
#include <iostream>
#include <queue>
#include <string>
using namespace std;

int main() {
    queue<string> visitors;
    string name;

    cout << "Enter 5 visitor names:\n";
    for (int i = 0; i < 5; ++i) {
        cout << "Name " << i + 1 << ": ";
        cin >> name;
        visitors.push(name);
    }

    while (!visitors.empty()) {
        cout << "Now serving: " << visitors.front() << endl;
        visitors.pop();
    }
```

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

    return 0;
}
```

TASK 6

```
#include <iostream>
#include <vector>
using namespace std;
double average(const vector<int>& v) {
    if (v.empty()) return 0.0;

    int sum = 0;
    for (auto it = v.begin(); it != v.end(); ++it) {
        sum += *it;
    }

    return static_cast<double>(sum) / v.size();
}

int main() {

    vector<int> numbers = {10, 20, 30, 40, 50};


    double avg = average(numbers);


    cout << "Average: " << avg << endl;

    return 0;
}
```

TASK 7

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {

    vector<int> nums;
    for (int i = 1; i <= 20; ++i) {
        nums.push_back(i);
    }
```

```
    reverse(nums.begin(), nums.end());


    int evenCount = count_if(nums.begin(), nums.end(), [](int n) {
        return n % 2 == 0;
    });

    cout << "Number of even elements: " << evenCount << endl;


    nums.erase(remove_if(nums.begin(), nums.end(), [](int n) {
        return n % 3 == 0;
    }), nums.end());


    cout << "Vector after removing numbers divisible by 3: ";
    for (int n : nums) {
        cout << n << " ";
    }
    cout << endl;

    return 0;
}
```