LAB MANUAL 10

TASK 01

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
   vector<int> myVector;

   myVector.push_back(1);
   myVector.push_back(2);
   myVector.push_back(3);
   myVector.push_back(4);

   cout << "Elements in the vector: ";
   for (auto it = myVector.begin(); it != myVector.end(); ++it) {
      cout << *it << " ";
   }
   cout << endl;

   myVector.push_back(5);

   if (!myVector.empty()) {
      int positionToRemove = 2;
      myVector.erase(myVector.begin() + positionToRemove);
   }

   cout << "Vector after pushing 5 and removing an element: ";
   for (const auto& element : myVector) {
```

```
        cout << element << " ";
    }
    cout << endl;

    return 0;
}
```

TASK 02

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

using namespace std;

double calculateMean(const vector<int>& grades) {
    int sum = 0;
    for (int grade : grades) {
        sum += grade;
    }
    return static_cast<double>(sum) / grades.size();
}

double calculateMedian(vector<int>& grades) {
    size_t size = grades.size();
    sort(grades.begin(), grades.end());
```

```
    if (size % 2 == 0) {
        return static_cast<double>(grades[size / 2 - 1] + grades[size / 2]) / 2;
    } else {
        return grades[size / 2];
    }
}

vector<int> calculateMode(const vector<int>& grades) {
    unordered_map<int, int> countMap;

    for (int grade : grades) {
        countMap[grade]++;
    }

    int maxFrequency = 0;
    for (const auto& pair : countMap) {
        maxFrequency = max(maxFrequency, pair.second);
    }

    vector<int> modeGrades;
    for (const auto& pair : countMap) {
        if (pair.second == maxFrequency) {
            modeGrades.push_back(pair.first);
        }
    }

    return modeGrades;
}
```

```
int main() {
    int numPairs;

    cout << "Enter the number of name/grade pairs: ";
    cin >> numPairs;

    vector<string> names;
    vector<int> grades;

    for (int i = 0; i < numPairs; ++i) {
        string name;
        int grade;

        cout << "Enter name: ";
        cin >> name;

        cout << "Enter grade: ";
        cin >> grade;

        names.push_back(name);
        grades.push_back(grade);
    }

    cout << "Mean of grades: " << calculateMean(grades) << endl;
    cout << "Median of grades: " << calculateMedian(grades) << endl;

    vector<int> modeGrades = calculateMode(grades);
    cout << "Mode of grades: ";
```

```
    for (int grade : modeGrades) {
        cout << grade << " ";
    }
    cout << endl;

    cout << "Students with the mode as their grade: ";
    for (size_t i = 0; i < grades.size(); ++i) {
        if (find(modeGrades.begin(), modeGrades.end(), grades[i]) != modeGrades.end()) {
            cout << names[i] << " ";
        }
    }
    cout << endl;

    return 0;
}
```