Lab manual-10

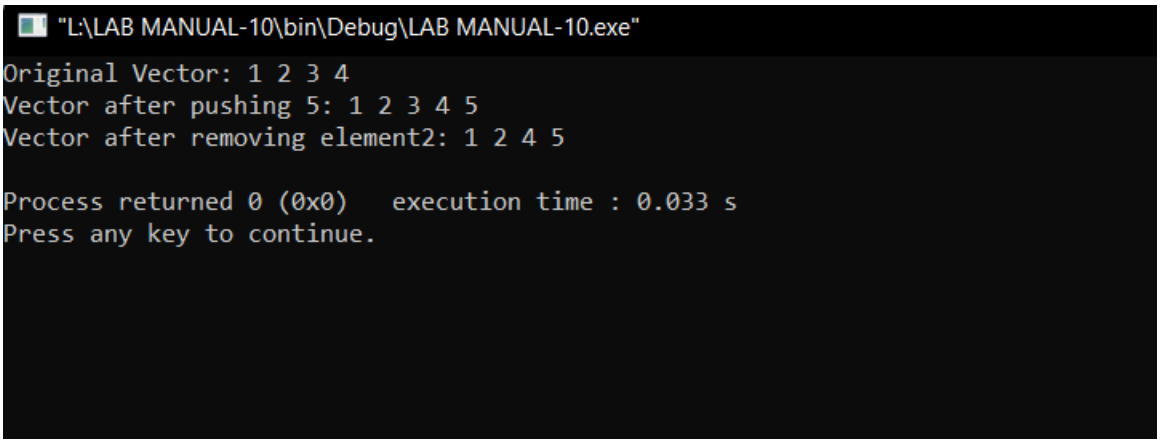
Muhammad Sibghat Rasool

457035

TASK 1:

```
#include <bits/stdc++.h>
int main() {
   std::vector<int> myVector;
   myVector.push_back(1);
   myVector.push_back(2);
   myVector.push_back(3);
   myVector.push_back(4);
   std::cout << "Original Vector: ";
   for (std::vector<int>::iterator it = myVector.begin(); it != myVector.end(); ++it) {
      std::cout << *it << " ";
   }
   std::cout << std::endl;
   myVector.push_back(5);
   std::cout << "Vector after pushing 5: ";
   for (std::vector<int>::iterator it = myVector.begin(); it != myVector.end(); ++it) {
      std::cout << *it << " ";
   }
   std::cout << std::endl;
   size_t positionToRemove = 2;
   if (positionToRemove < myVector.size()) {
      myVector.erase(myVector.begin() + positionToRemove);
      std::cout << "Vector after removing element" << positionToRemove << ": ";
```

```
    for (std::vector<int>::iterator it = myVector.begin(); it != myVector.end(); ++it) {
        std::cout << *it << " ";
    }
    std::cout << std::endl;
  } else {
    std::cout << "Invalid position" << std::endl;
  }
  return 0;
}
```



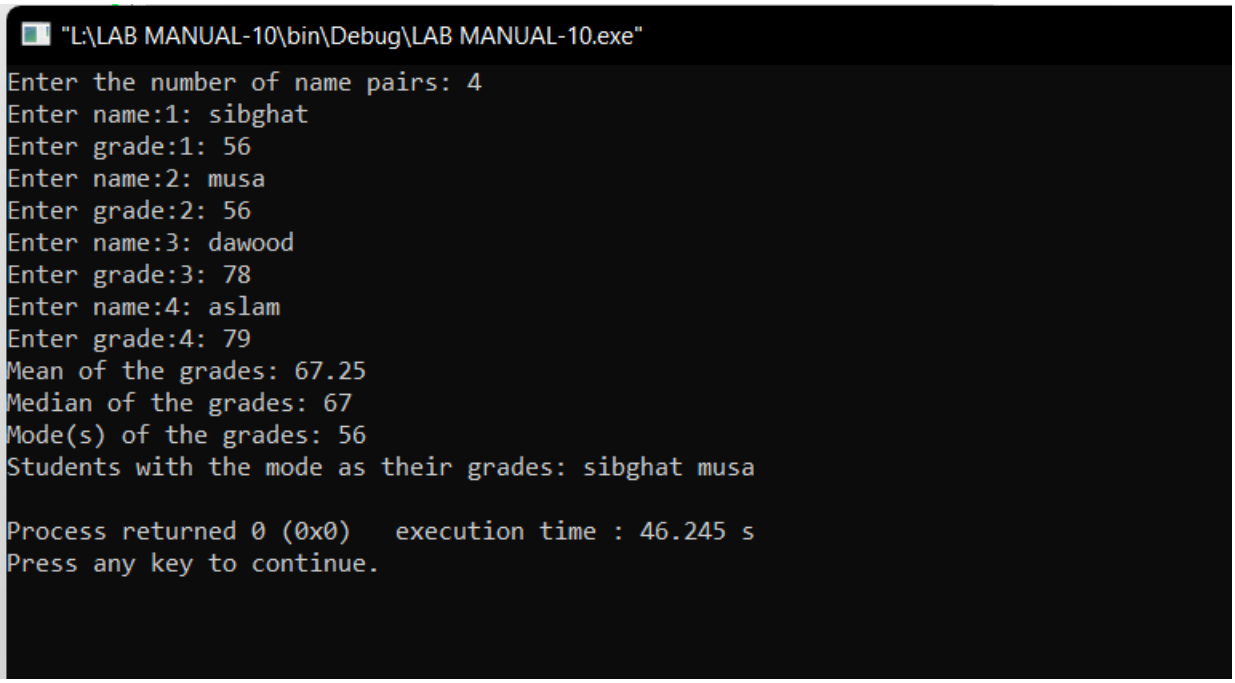
Task 2:

```
#include<bits/stdc++.h>
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) {
    map<int, int> frequencyMap;
    for (int grade : grades) {
        frequencyMap[grade]++;
    }
    int maxFrequency = 0;
    vector<int> mode;
    for (const auto& entry : frequencyMap) {
        if (entry.second > maxFrequency) {
            maxFrequency = entry.second;
            mode = {entry.first};
        } else if (entry.second == maxFrequency) {
            mode.push_back(entry.first);
        }
    }
    return mode;
}
```

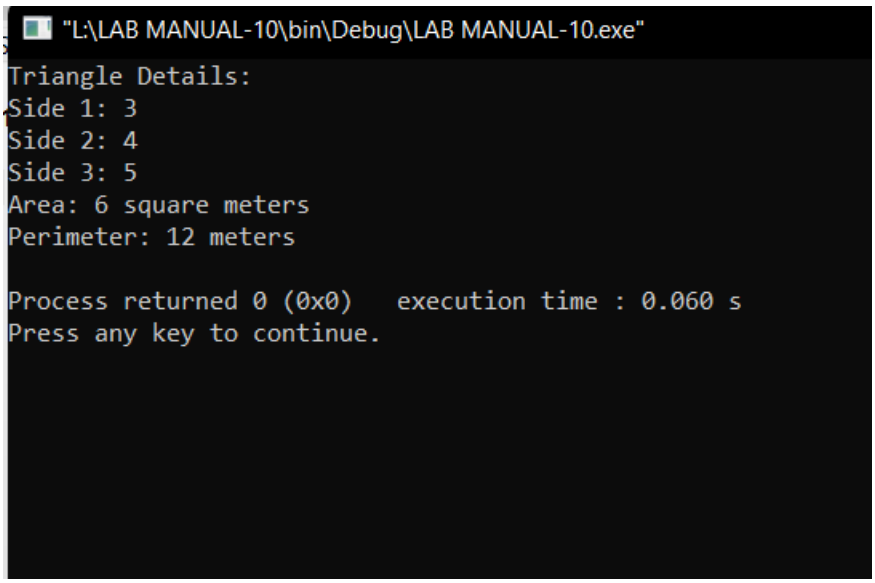
```
int main() {
   int numPairs;
   cout << "Enter the number of name/grade pairs: ";
   cin >> numPairs;
   if (numPairs <= 0) {
      cout << "Invalid input. Exiting program." << endl;
      return 1;
   }
   vector<string> names;
   vector<int> grades;
   for (int i = 0; i < numPairs; ++i) {
      string name;
      int grade;
      cout << "Enter name #" << i + 1 << ": ";
      cin >> name;
      cout << "Enter grade #" << i + 1 << ": ";
      cin >> grade;
      names.push_back(name);
      grades.push_back(grade);
   }
   cout << "Mean of the grades: " << calculateMean(grades) << endl;
return 0;
}
```



TASK 3:

```
#include <bits/stdc++.h>
using namespace std;
class Triangle {
private:
  double side1, side2, side3;
public:
  Triangle(double s1, double s2, double s3) : side1(s1), side2(s2), side3(s3) {}
  double calculateArea() const {
    double s = (side1 + side2 + side3) / 2;
    return sqrt(s * (s - side1) * (s - side2) * (s - side3));
  }
  double calculatePerimeter() const {
```

```
    return side1 + side2 + side3;
  }
  void printDetails() const {
    cout << "Triangle Details:" <<endl;
    cout << "Side 1: " << side1 <<endl;
    cout << "Side 2: " << side2 <<endl;
    cout << "Side 3: " << side3 << endl;
    cout << "Area: " << calculateArea() << " square meters" << endl;
    cout << "Perimeter: " << calculatePerimeter() << " meters" << endl;
  }
};
int main() {
  Triangle myTriangle(3, 4, 5);
  myTriangle.printDetails();
  return 0;
}
```



TASK 4:

```
#include <bits/stdc++.h>
using namespace std;
struct Employee {
  string name;
  double salary;
  int hoursWorkedPerDay;
};
void increaseSalary(Employee& emp) {
  if (emp.hoursWorkedPerDay > 8) {
    emp.salary += (emp.hoursWorkedPerDay - 8) * 10;
  }
}
int main() {
  const int numEmployees = 10;
  Employee employees[numEmployees];
  for (int i = 0; i < numEmployees; ++i) {
    cout << "Enter name:" << i + 1 << ": ";
    getline(cin, employees[i].name);
    cout << "Enter salary:" << i + 1 << ": ";
    cin >> employees[i].salary;
    cout << "Enter hours of work per day:" << i + 1 << ": ";
    cin >> employees[i].hoursWorkedPerDay;
    cin.ignore();
  }
  for (int i = 0; i < numEmployees; ++i) {
    increaseSalary(employees[i]);
  }
```

```
    cout << "\nEmployee Details after Salary Increase:\n";
    for (int i = 0; i < numEmployees; ++i) {
        cout << "Name: " << employees[i].name << ", Final Salary: $" << employees[i].salary << endl;
    }
    return 0;
}
```