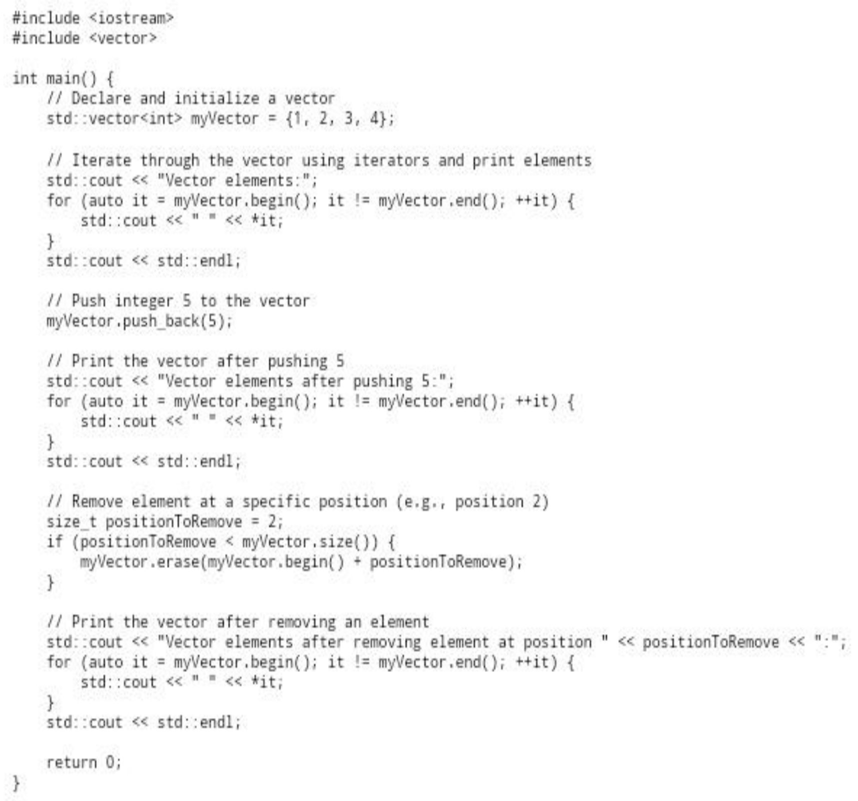
National University of Science and Technology, NUST
School of Mechanical and Manufacturing, SMME

Prepared by:

Name	CMS	Class	Section
M.Asim Shah	470474	ME-15	C

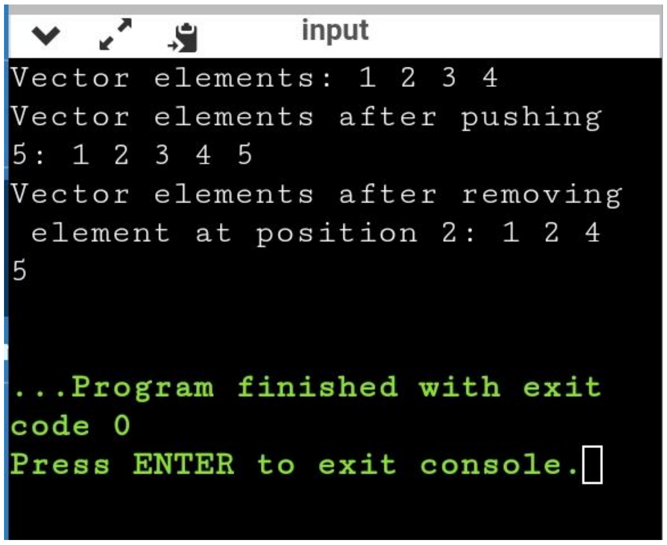
Task 1:

Input:



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

int main() {
    // Declare and initialize a vector
    std::vector<int> myVector = {1, 2, 3, 4};

    // Iterate through the vector using iterators and print elements
    std::cout << "Vector elements:";
    for (auto it = myVector.begin(); it != myVector.end(); ++it) {
        std::cout << " " << *it;
    }
    std::cout << std::endl;

    // Push integer 5 to the vector
    myVector.push_back(5);

    // Print the vector after pushing 5
    std::cout << "Vector elements after pushing 5:";
    for (auto it = myVector.begin(); it != myVector.end(); ++it) {
        std::cout << " " << *it;
    }
    std::cout << std::endl;

    // Remove element at a specific position (e.g., position 2)
    size_t positionToRemove = 2;
    if (positionToRemove < myVector.size()) {
        myVector.erase(myVector.begin() + positionToRemove);
    }

    // Print the vector after removing an element
    std::cout << "Vector elements after removing element at position " << positionToRemove << ":";
    for (auto it = myVector.begin(); it != myVector.end(); ++it) {
        std::cout << " " << *it;
    }
    std::cout << std::endl;

    return 0;
}
```

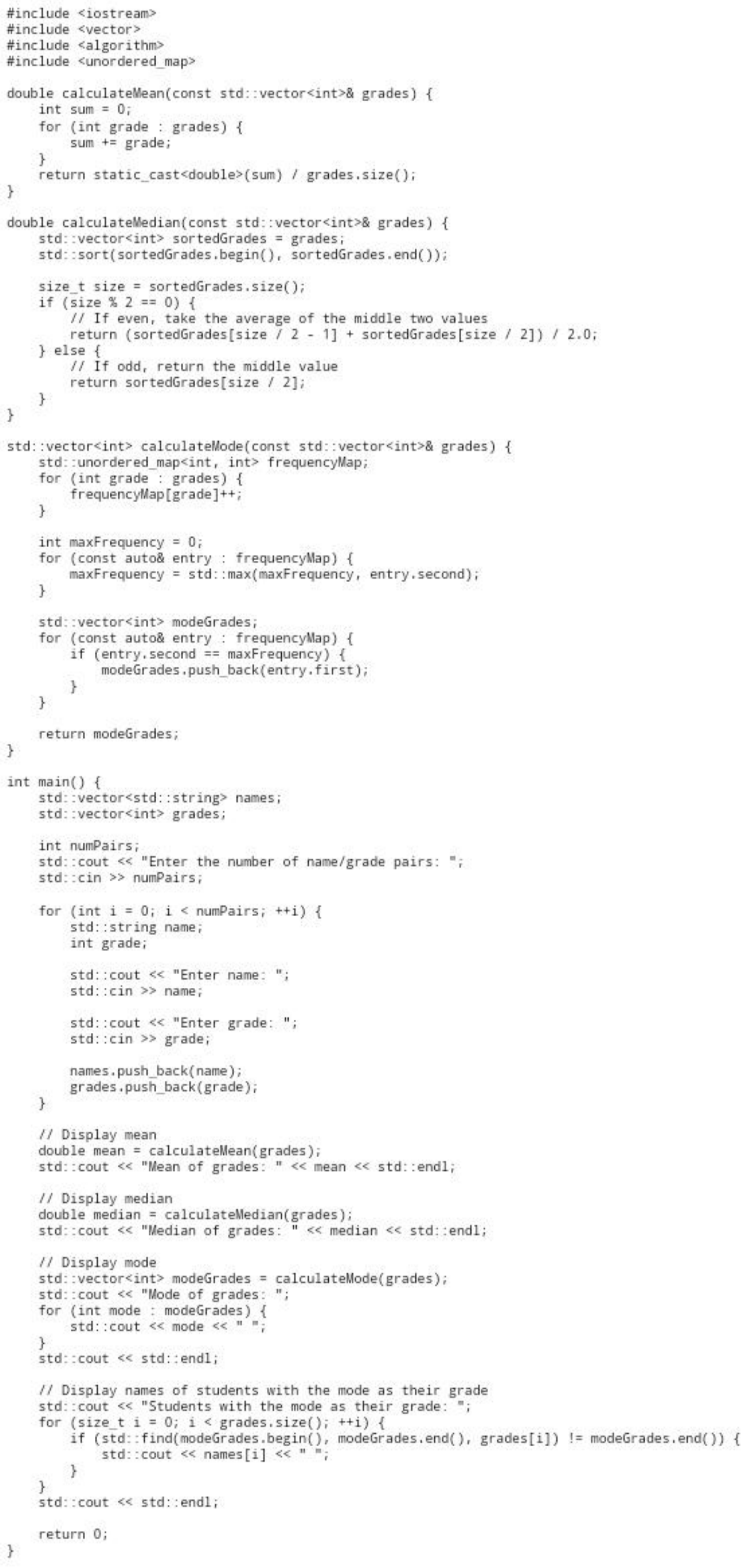
Output



```
Vector elements: 1 2 3 4
Vector elements after pushing
5: 1 2 3 4 5
Vector elements after removing
 element at position 2: 1 2 4
5


...Program finished with exit
code 0
Press ENTER to exit console.
```

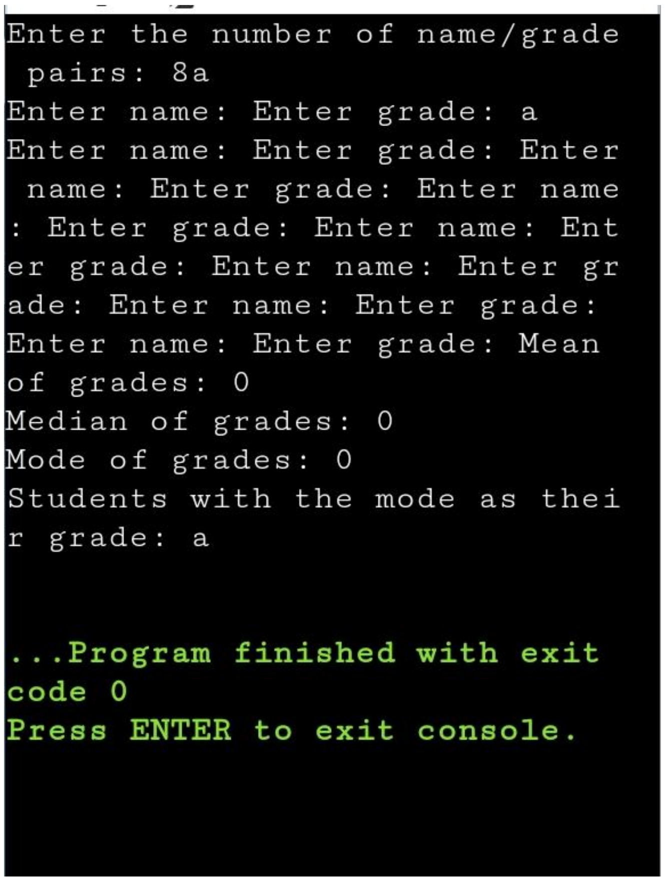
Task 2:

Input:



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

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

double calculateMedian(const std::vector<int>& grades) {
    std::vector<int> sortedGrades = grades;
    std::sort(sortedGrades.begin(), sortedGrades.end());

    size_t size = sortedGrades.size();
    if (size % 2 == 0) {
        // If even, take the average of the middle two values
        return (sortedGrades[size / 2 - 1] + sortedGrades[size / 2]) / 2.0;
    } else {
        // If odd, return the middle value
        return sortedGrades[size / 2];
    }
}

std::vector<int> calculateMode(const std::vector<int>& grades) {
    std::unordered_map<int, int> frequencyMap;
    for (int grade : grades) {
        frequencyMap[grade]++;
    }

    int maxFrequency = 0;
    for (const auto& entry : frequencyMap) {
        maxFrequency = std::max(maxFrequency, entry.second);
    }

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

    return modeGrades;
}

int main() {
    std::vector<std::string> names;
    std::vector<int> grades;

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

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

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

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

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

    // Display mean
    double mean = calculateMean(grades);
    std::cout << "Mean of grades: " << mean << std::endl;

    // Display median
    double median = calculateMedian(grades);
    std::cout << "Median of grades: " << median << std::endl;

    // Display mode
    std::vector<int> modeGrades = calculateMode(grades);
    std::cout << "Mode of grades: ";
    for (int mode : modeGrades) {
        std::cout << mode << " ";
    }
    std::cout << std::endl;

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

    return 0;
}
```

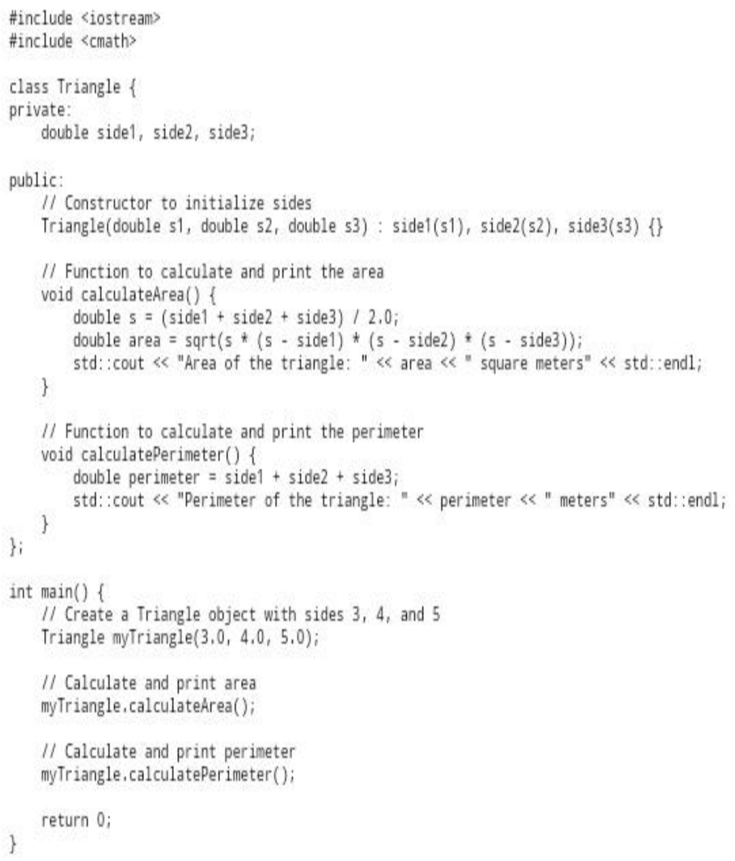
Output:



```
Enter the number of name/grade
 pairs: 8a
Enter name: Enter grade: a
Enter name: Enter grade: Enter
 name: Enter grade: Enter name
: Enter grade: Enter name: Ent
er grade: Enter name: Enter gr
ade: Enter name: Enter grade:
Enter name: Enter grade: Mean
of grades: 0
Median of grades: 0
Mode of grades: 0
Students with the mode as thei
r grade: a


...Program finished with exit
code 0
Press ENTER to exit console.
```

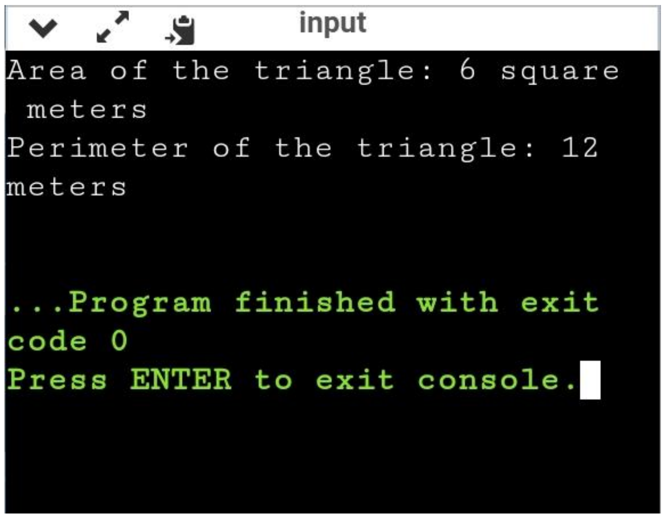
Task 3:

Input:



```
#include <iostream>
#include <cmath>

class Triangle {
private:
    double side1, side2, side3;

public:
    // Constructor to initialize sides
    Triangle(double s1, double s2, double s3) : side1(s1), side2(s2), side3(s3) {}

    // Function to calculate and print the area
    void calculateArea() {
        double s = (side1 + side2 + side3) / 2.0;
        double area = sqrt(s * (s - side1) * (s - side2) * (s - side3));
        std::cout << "Area of the triangle: " << area << " square meters" << std::endl;
    }

    // Function to calculate and print the perimeter
    void calculatePerimeter() {
        double perimeter = side1 + side2 + side3;
        std::cout << "Perimeter of the triangle: " << perimeter << " meters" << std::endl;
    }
};

int main() {
    // Create a Triangle object with sides 3, 4, and 5
    Triangle myTriangle(3.0, 4.0, 5.0);

    // Calculate and print area
    myTriangle.calculateArea();

    // Calculate and print perimeter
    myTriangle.calculatePerimeter();

    return 0;
}
```

Output:

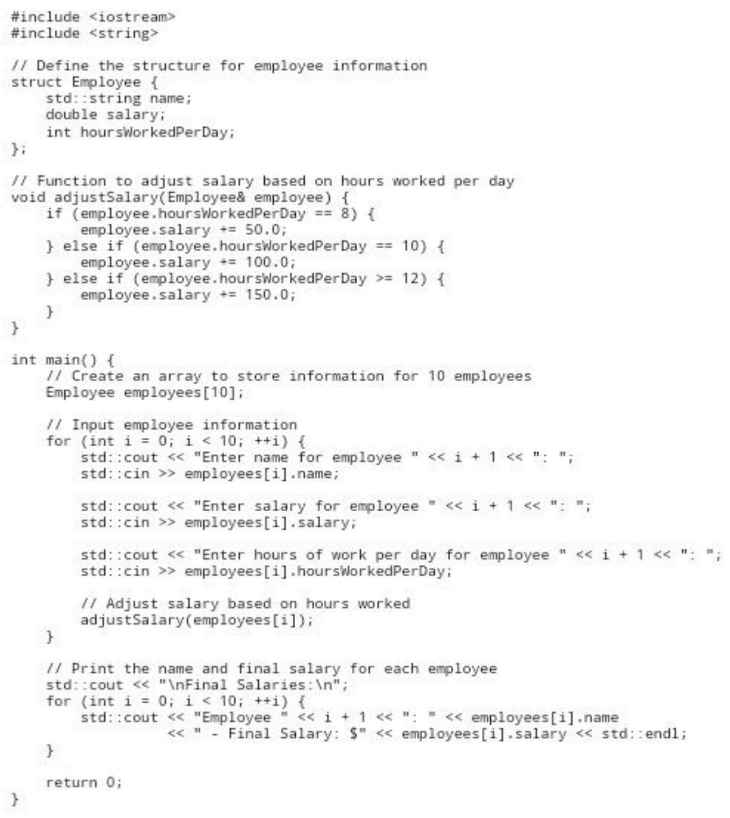


```
Area of the triangle: 6 square
 meters
Perimeter of the triangle: 12
meters


...Program finished with exit
code 0
Press ENTER to exit console.
```

National University of Science and Technology, NUST
School of Mechanical and Manufacturing, SMME

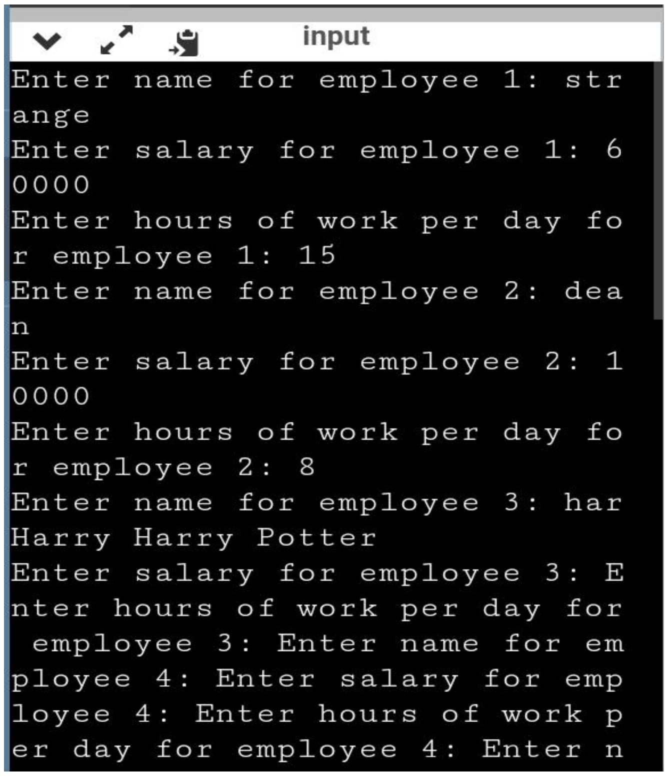
Task 4:

Input:



```
#include <iostream>
#include <string>

// Define the structure for employee information
struct Employee {
    std::string name;
    double salary;
    int hoursWorkedPerDay;
};

// Function to adjust salary based on hours worked per day
void adjustSalary(Employee& employee) {
    if (employee.hoursWorkedPerDay == 8) {
        employee.salary += 50.0;
    } else if (employee.hoursWorkedPerDay == 10) {
        employee.salary += 100.0;
    } else if (employee.hoursWorkedPerDay >= 12) {
        employee.salary += 150.0;
    }
}

int main() {
    // Create an array to store information for 10 employees
    Employee employees[10];

    // Input employee information
    for (int i = 0; i < 10; ++i) {
        std::cout << "Enter name for employee " << i + 1 << ": ";
        std::cin >> employees[i].name;

        std::cout << "Enter salary for employee " << i + 1 << ": ";
        std::cin >> employees[i].salary;

        std::cout << "Enter hours of work per day for employee " << i + 1 << ": ";
        std::cin >> employees[i].hoursWorkedPerDay;

        // Adjust salary based on hours worked
        adjustSalary(employees[i]);
    }

    // Print the name and final salary for each employee
    std::cout << "\nFinal Salaries:\n";
    for (int i = 0; i < 10; ++i) {
        std::cout << "Employee " << i + 1 << ": " << employees[i].name
                  << " - Final Salary: $" << employees[i].salary << std::endl;
    }

    return 0;
}
```

Output:



```
Enter name for employee 1: str
ange
Enter salary for employee 1: 6
0000
Enter hours of work per day fo
r employee 1: 15
Enter name for employee 2: dea
n
Enter salary for employee 2: 1
0000
Enter hours of work per day fo
r employee 2: 8
Enter name for employee 3: har
Harry Harry Potter
Enter salary for employee 3: E
nter hours of work per day for
 employee 3: Enter name for em
ployee 4: Enter salary for emp
loyee 4: Enter hours of work p
er day for employee 4: Enter n
```