San Francisco Bay University

CS360L - Programming in C and C++ Lab
Lab Assignment #5

Due day: 4/5/2024

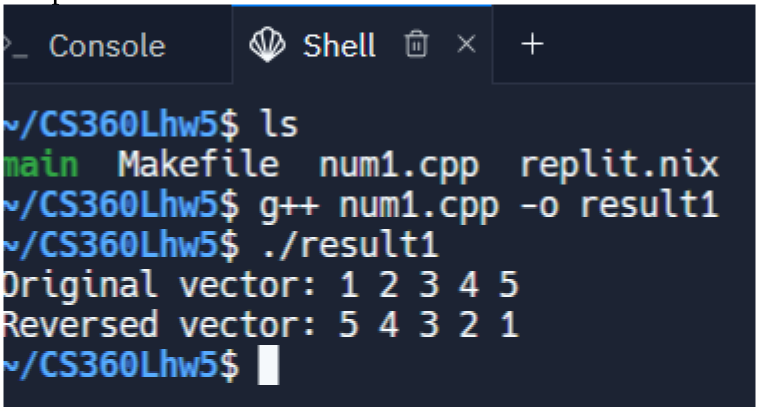
Kritika Regmi
ID:19702

1. Write a function that takes a vector of integers as argument and reverses its elements.

```
void rvrs(Vector<int>& vct){
      //Complete your program
}
```

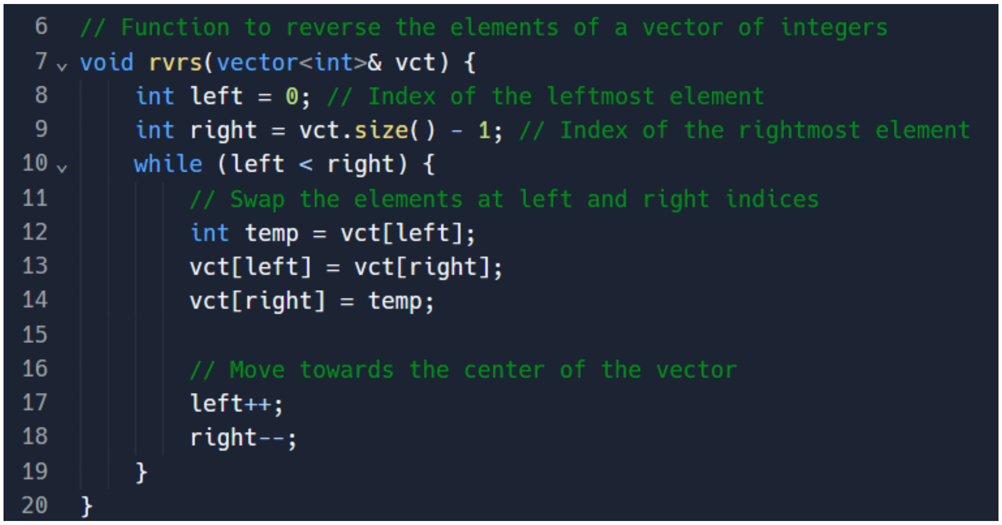
ANS:

Output:

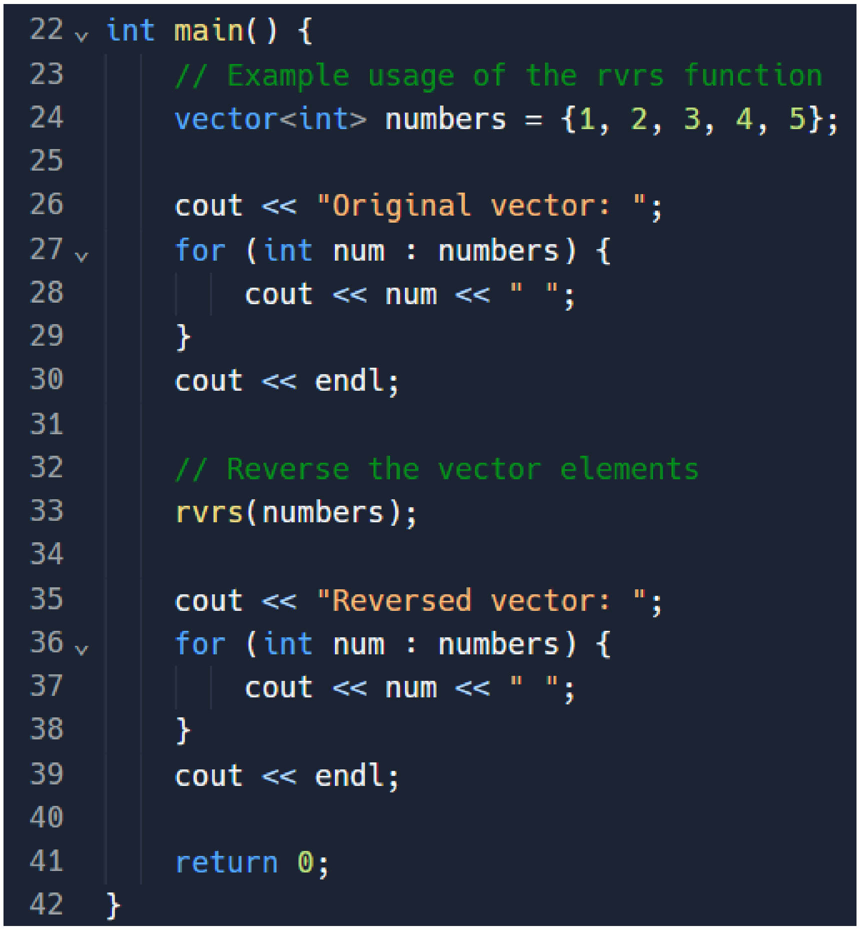


```
~/CS360Lhw5$ ls
main  Makefile  num1.cpp  replit.nix
~/CS360Lhw5$ g++ num1.cpp -o result1
~/CS360Lhw5$ ./result1
Original vector: 1 2 3 4 5
Reversed vector: 5 4 3 2 1
~/CS360Lhw5$ 
```

Code:



```cpp
// Function to reverse the elements of a vector of integers
void rvrs(vector<int>& vct) {
    int left = 0; // Index of the leftmost element
    int right = vct.size() - 1; // Index of the rightmost element
    while (left < right) {
        // Swap the elements at left and right indices
        int temp = vct[left];
        vct[left] = vct[right];
        vct[right] = temp;

        // Move towards the center of the vector
        left++;
        right--;
    }
}
```

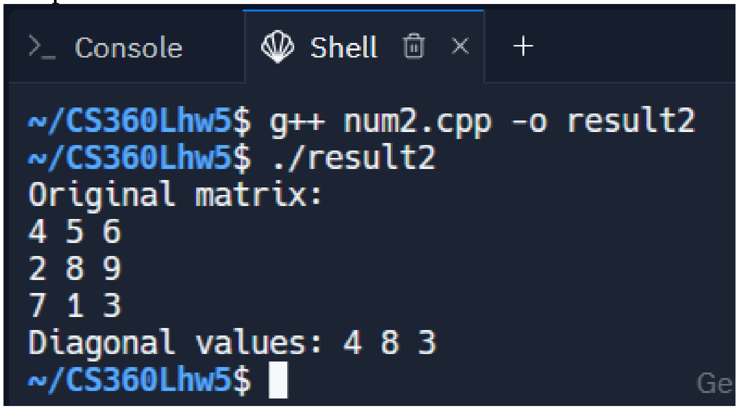


```cpp
int main() {
    // Example usage of the rvrs function
    vector<int> numbers = {1, 2, 3, 4, 5};

    cout << "Original vector: ";
    for (int num : numbers) {
        cout << num << " ";
    }
    cout << endl;

    // Reverse the vector elements
    rvrs(numbers);

    cout << "Reversed vector: ";
    for (int num : numbers) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

2. Find a function with one argument, vector of vectors named *vals*, for coordinates of one of its elements in *row* and *col* to print the values that lie on the lower-left to upper-right diagonal of *vals*. After that, verify it in *main* function.

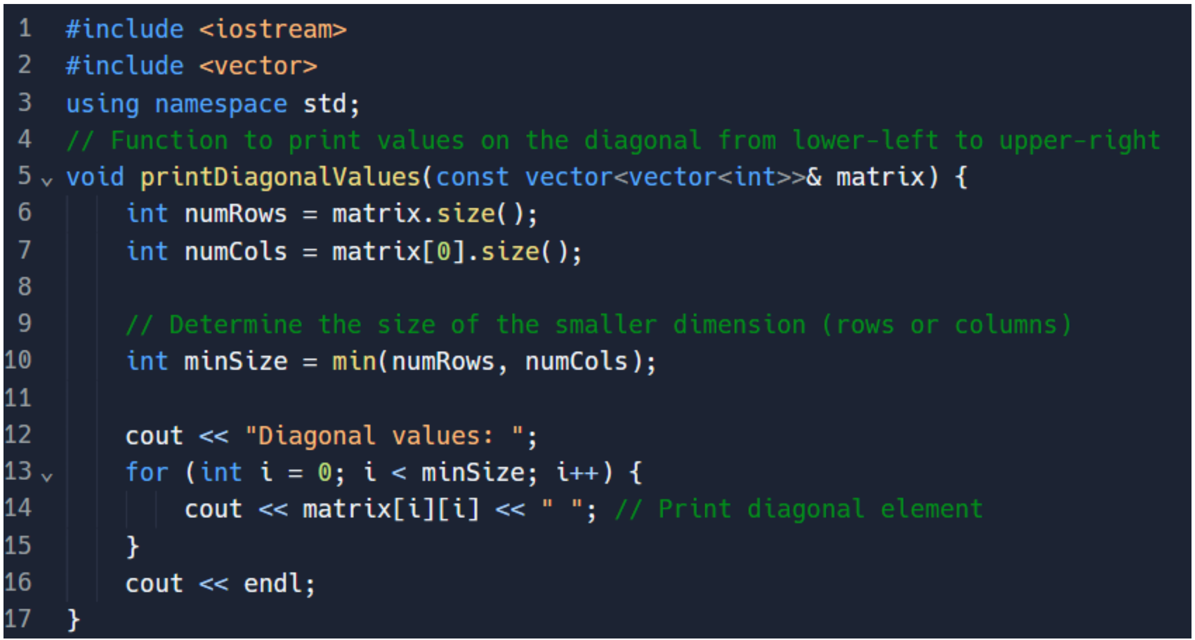
ANS:

Output:

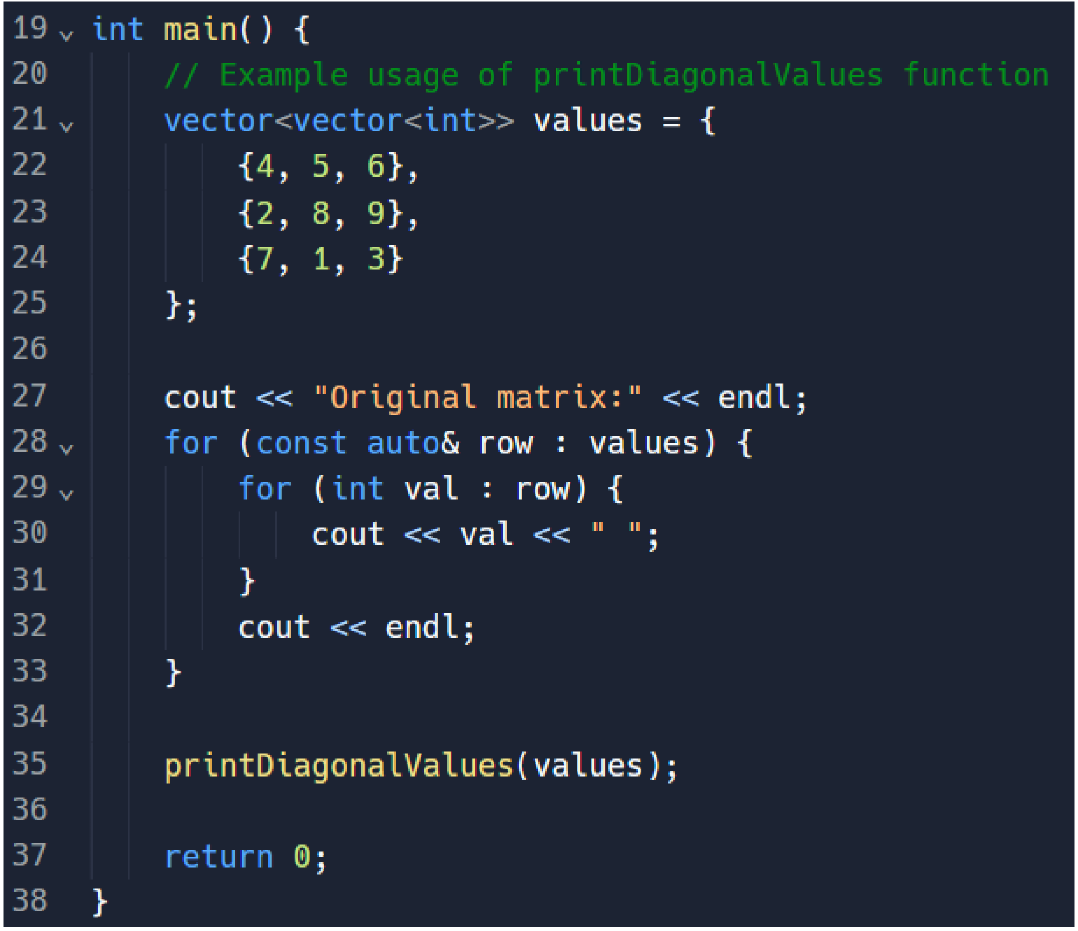


```
~/CS360Lhw5$ g++ num2.cpp -o result2
~/CS360Lhw5$ ./result2
Original matrix:
4 5 6
2 8 9
7 1 3
Diagonal values: 4 8 3
~/CS360Lhw5$
```

Code:



```cpp
#include <iostream>
#include <vector>
using namespace std;
// Function to print values on the diagonal from lower-left to upper-right
void printDiagonalValues(const vector<vector<int>>& matrix) {
    int numRows = matrix.size();
    int numCols = matrix[0].size();

    // Determine the size of the smaller dimension (rows or columns)
    int minSize = min(numRows, numCols);

    cout << "Diagonal values: ";
    for (int i = 0; i < minSize; i++) {
        cout << matrix[i][i] << " "; // Print diagonal element
    }
    cout << endl;
}
```

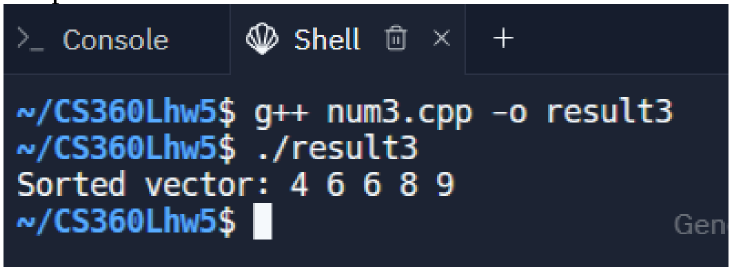


```cpp
int main() {
    // Example usage of printDiagonalValues function
    vector<vector<int>> values = {
        {4, 5, 6},
        {2, 8, 9},
        {7, 1, 3}
    };

    cout << "Original matrix:" << endl;
    for (const auto& row : values) {
        for (int val : row) {
            cout << val << " ";
        }
        cout << endl;
    }

    printDiagonalValues(values);

    return 0;
}
```

3. Create a class *Tensor* with a method *sort* to sort a vector input argument and print it out. Please verify this correctness in *main* function.

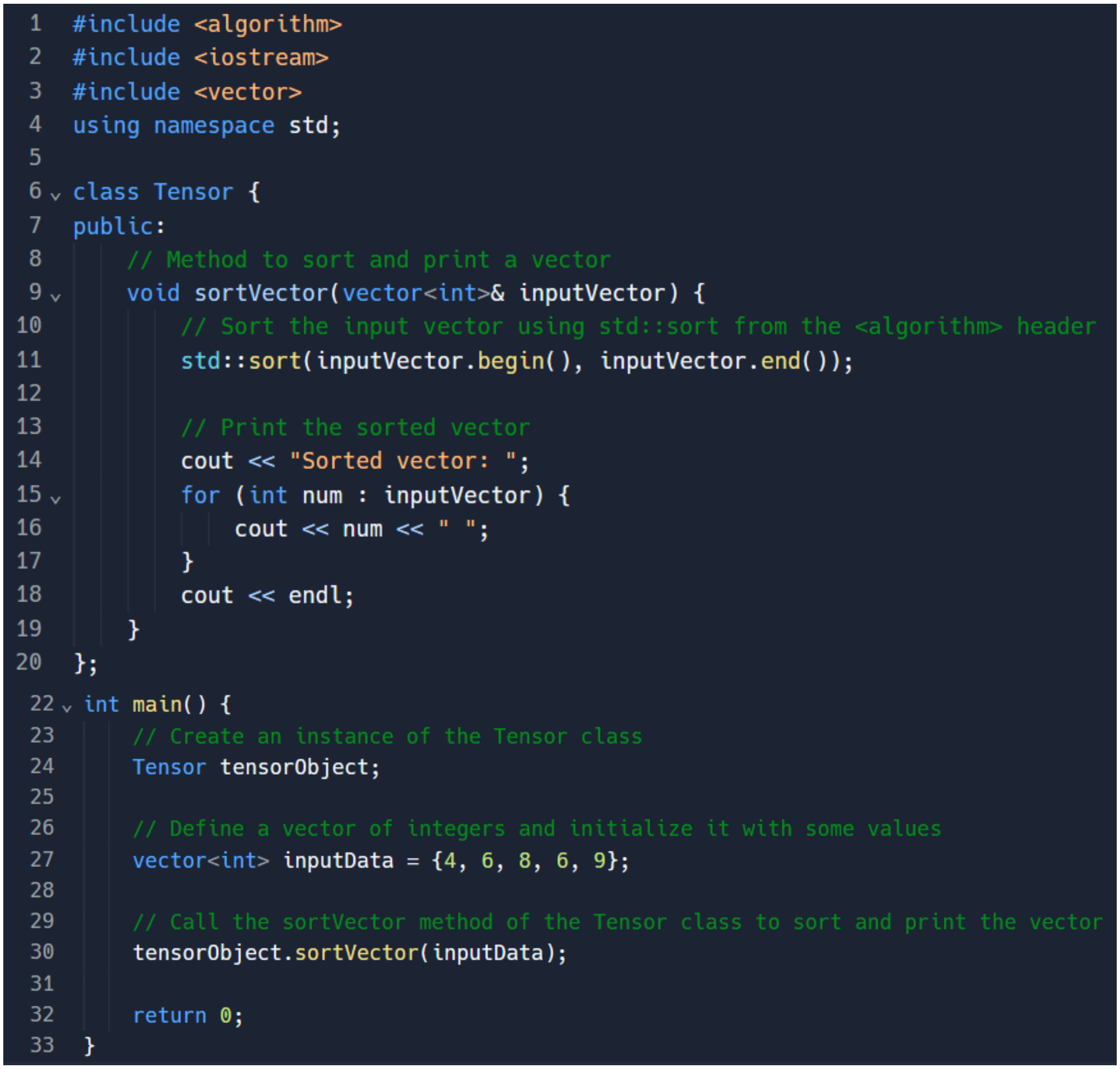
ANS:

Output:



```
~/CS360Lhw5$ g++ num3.cpp -o result3
~/CS360Lhw5$ ./result3
Sorted vector: 4 6 6 8 9
~/CS360Lhw5$ 
```

CODE:

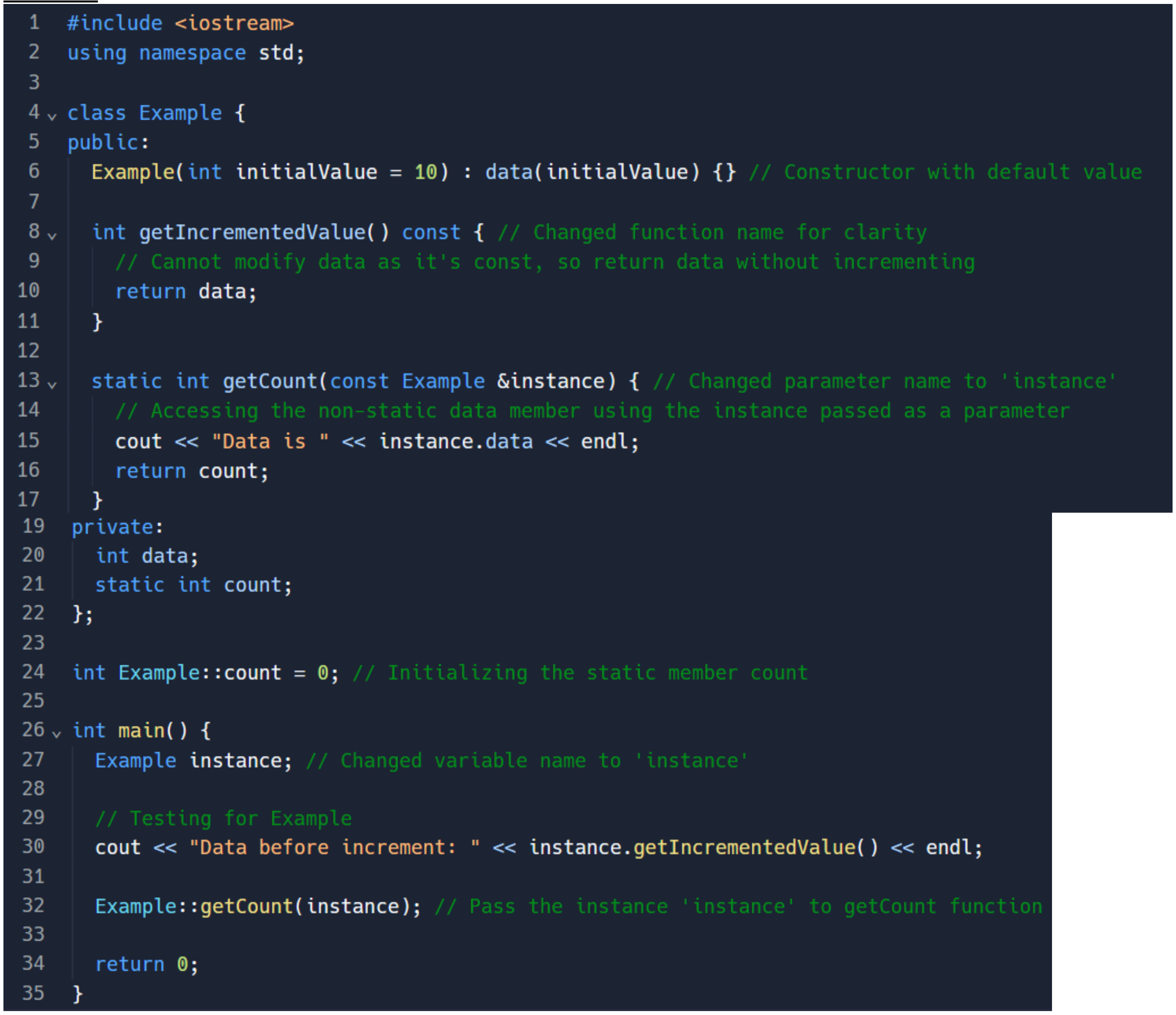


```cpp
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

class Tensor {
public:
    // Method to sort and print a vector
    void sortVector(vector<int>& inputVector) {
        // Sort the input vector using std::sort from the <algorithm> header
        std::sort(inputVector.begin(), inputVector.end());

        // Print the sorted vector
        cout << "Sorted vector: ";
        for (int num : inputVector) {
            cout << num << " ";
        }
        cout << endl;
    }
};
int main() {
    // Create an instance of the Tensor class
    Tensor tensorObject;

    // Define a vector of integers and initialize it with some values
    vector<int> inputData = {4, 6, 8, 6, 9};

    // Call the sortVector method of the Tensor class to sort and print the vector
    tensorObject.sortVector(inputData);

    return 0;
}
```

4. Find the errors in the following class and explain how to correct them. Please test it in main function.

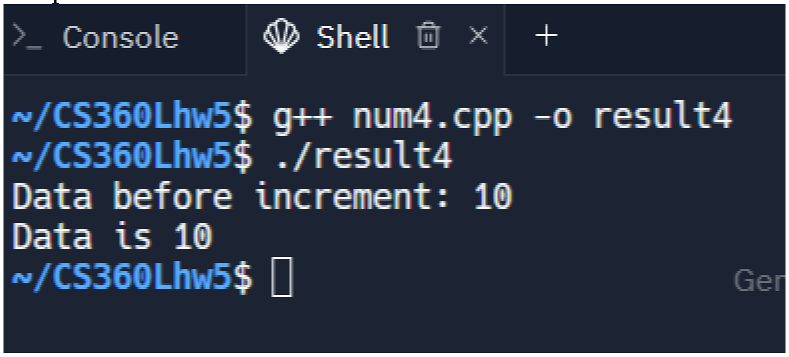
```
class Example{
  public:
    Example( int y = 10 ): data( y ){
      // empty body
    } // end Example constructor
    int getIncrementedData() const{
      return data++;
    } // end function getIncrementedData
    static int getCount(){
      cout << "Data is " << data << endl;
      return count;
    } // end function getCount
  private:
    int data;
    static int count;
}; // end class Example
```

CODE:



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

class Example {
public:
  Example(int initialValue = 10) : data(initialValue) {} // Constructor with default value

  int getIncrementedValue() const { // Changed function name for clarity
    // Cannot modify data as it's const, so return data without incrementing
    return data;
  }

  static int getCount(const Example &instance) { // Changed parameter name to 'instance'
    // Accessing the non-static data member using the instance passed as a parameter
    cout << "Data is " << instance.data << endl;
    return count;
  }
private:
  int data;
  static int count;
};

int Example::count = 0; // Initializing the static member count

int main() {
  Example instance; // Changed variable name to 'instance'

  // Testing for Example
  cout << "Data before increment: " << instance.getIncrementedValue() << endl;

  Example::getCount(instance); // Pass the instance 'instance' to getCount function

  return 0;
}
```

Output:



```
~/CS360Lhw5$ g++ num4.cpp -o result4
~/CS360Lhw5$ ./result4
Data before increment: 10
Data is 10
~/CS360Lhw5$
```