San Francisco Bay University

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

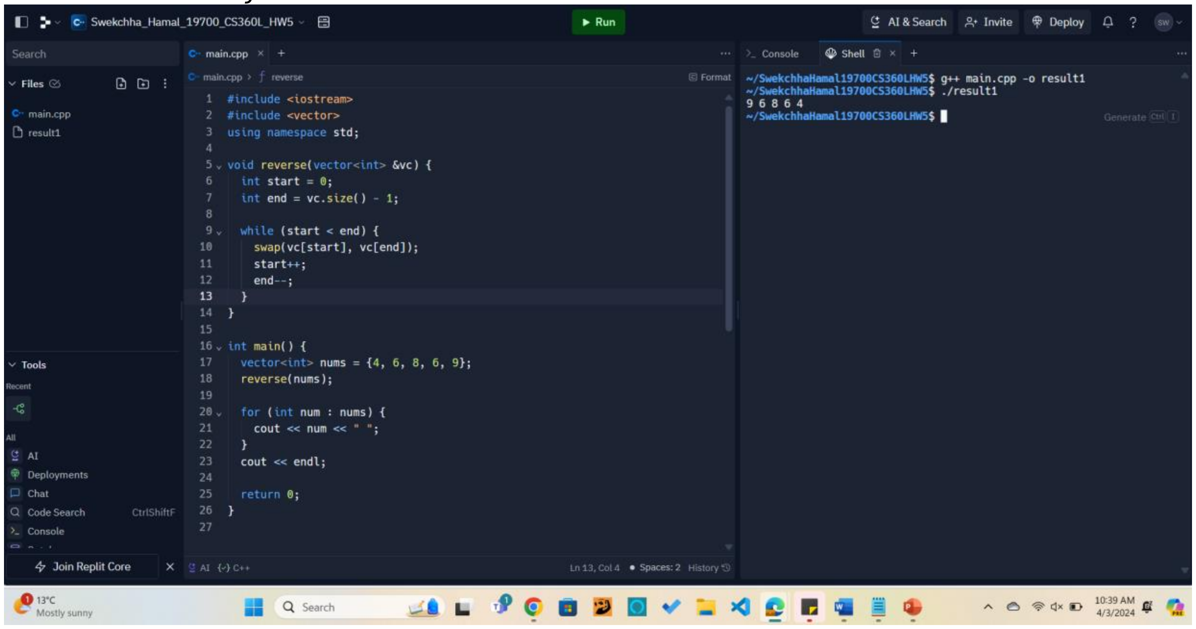
Due day: 4/5/2024

Instruction:

1. Push the answer sheets/source code to Github
2. Please follow the code style rule like programs on handout.
3. Overdue lab assignment submission can't be accepted.
4. Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)

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

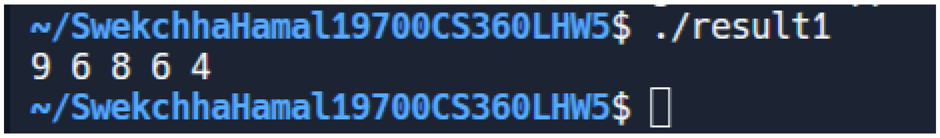
```
void rvrs(Vector<int>& vct){
        //Complete your program
}
```



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

void reverse(vector<int> &vc) {
  int start = 0;
  int end = vc.size() - 1;

  while (start < end) {
    swap(vc[start], vc[end]);
    start++;
    end--;
  }
}

int main() {
  vector<int> nums = {4, 6, 8, 6, 9};
  reverse(nums);

  for (int num : nums) {
    cout << num << " ";
  }
  cout << endl;

  return 0;
}
```

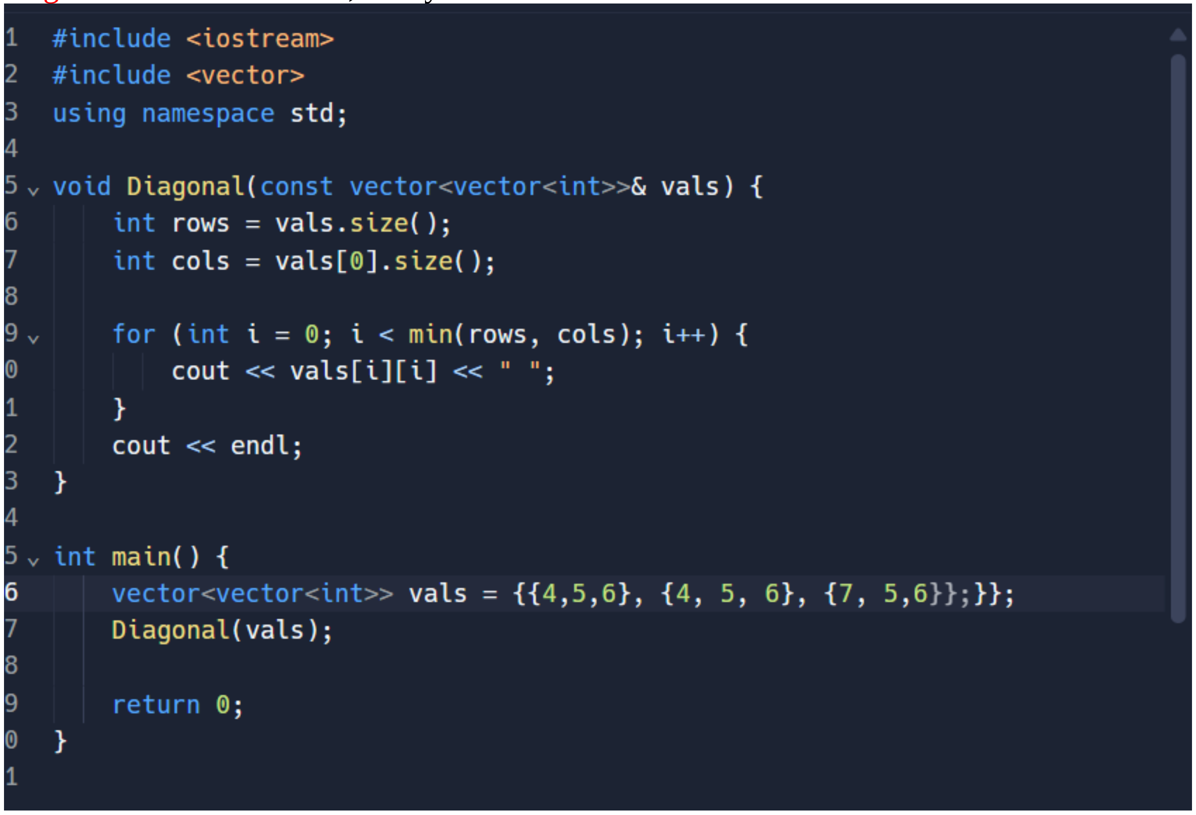
```
~/SwekchhaHamal19700CS360LHW5$ g++ main.cpp -o result1
~/SwekchhaHamal19700CS360LHW5$ ./result1
9 6 8 6 4
~/SwekchhaHamal19700CS360LHW5$
```

Output:



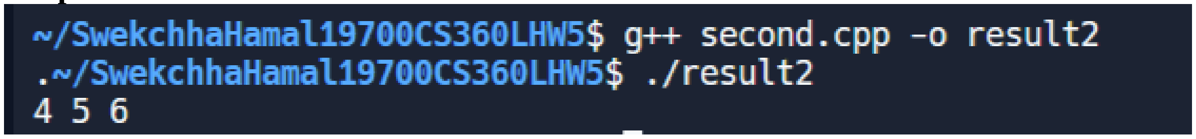
```
~/SwekchhaHamal19700CS360LHW5$ ./result1
9 6 8 6 4
~/SwekchhaHamal19700CS360LHW5$
```

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.



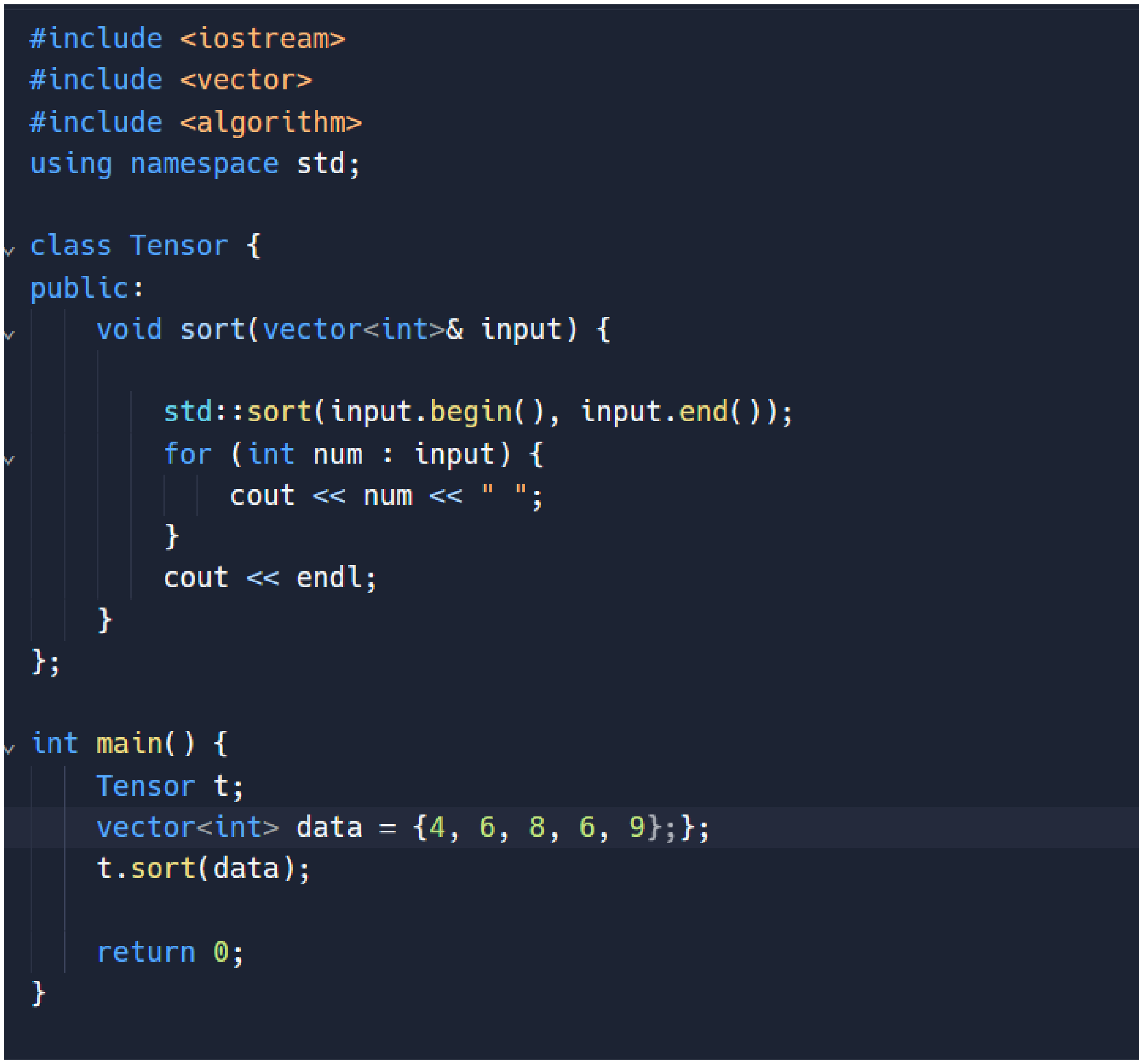
```cpp
#include <iostream>
#include <vector>
using namespace std;

void Diagonal(const vector<vector<int>>& vals) {
    int rows = vals.size();
    int cols = vals[0].size();

    for (int i = 0; i < min(rows, cols); i++) {
        cout << vals[i][i] << " ";
    }
    cout << endl;
}

int main() {
    vector<vector<int>> vals = {{4,5,6}, {4, 5, 6}, {7, 5,6}};}};
    Diagonal(vals);

    return 0;
}

```

Output:



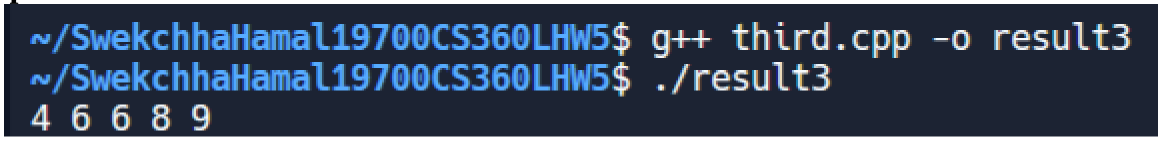
```
~/SwekchhaHamal19700CS360LHW5$ g++ second.cpp -o result2
.~/SwekchhaHamal19700CS360LHW5$ ./result2
4 5 6
```

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



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

class Tensor {
public:
    void sort(vector<int>& input) {

        std::sort(input.begin(), input.end());
        for (int num : input) {
            cout << num << " ";
        }
        cout << endl;
    }
};

int main() {
    Tensor t;
    vector<int> data = {4, 6, 8, 6, 9};};
    t.sort(data);

    return 0;
}
```

Output :

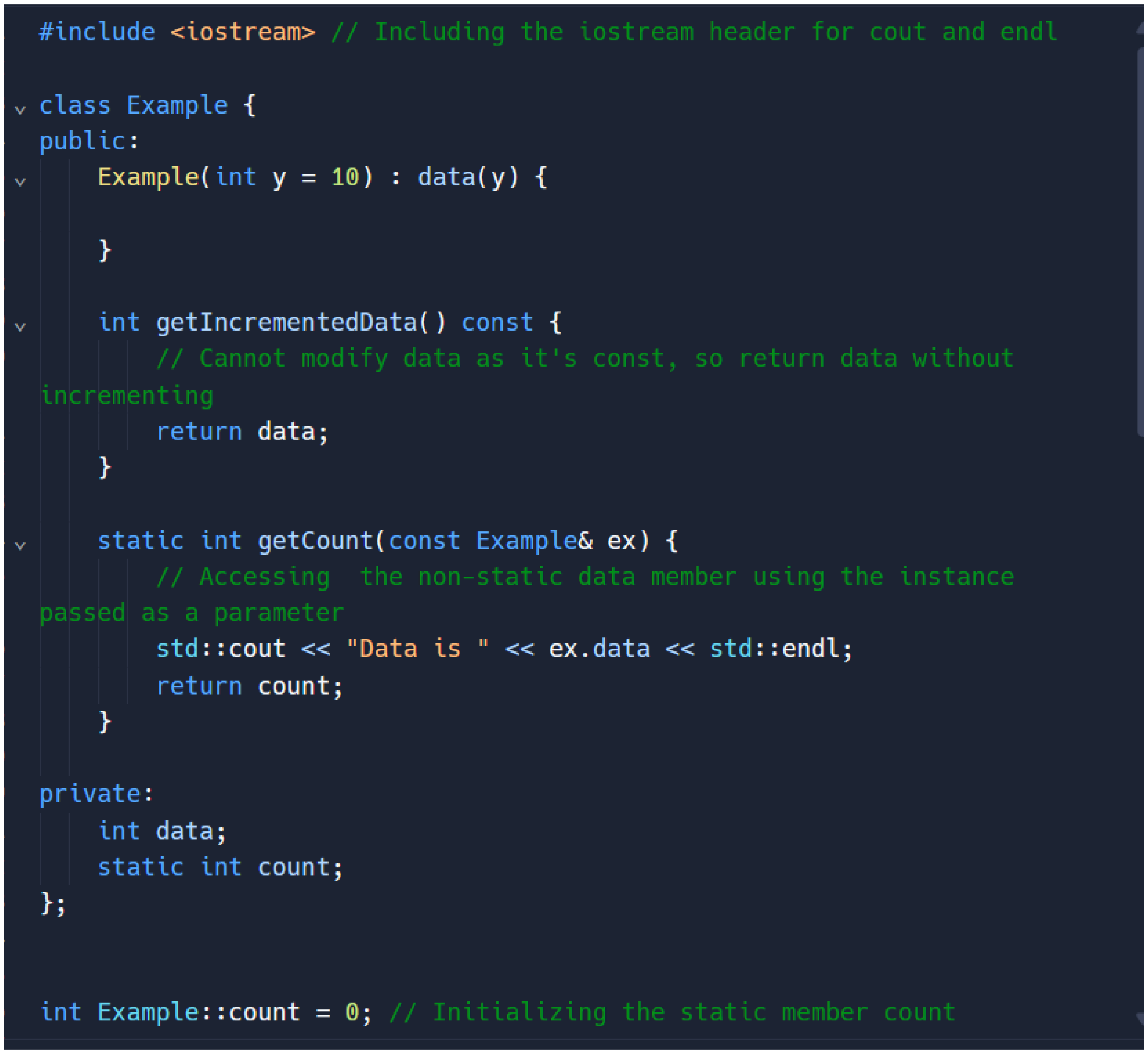


```
~/SwekchhaHamal19700CS360LHW5$ g++ third.cpp -o result3
~/SwekchhaHamal19700CS360LHW5$ ./result3
4 6 6 8 9
```

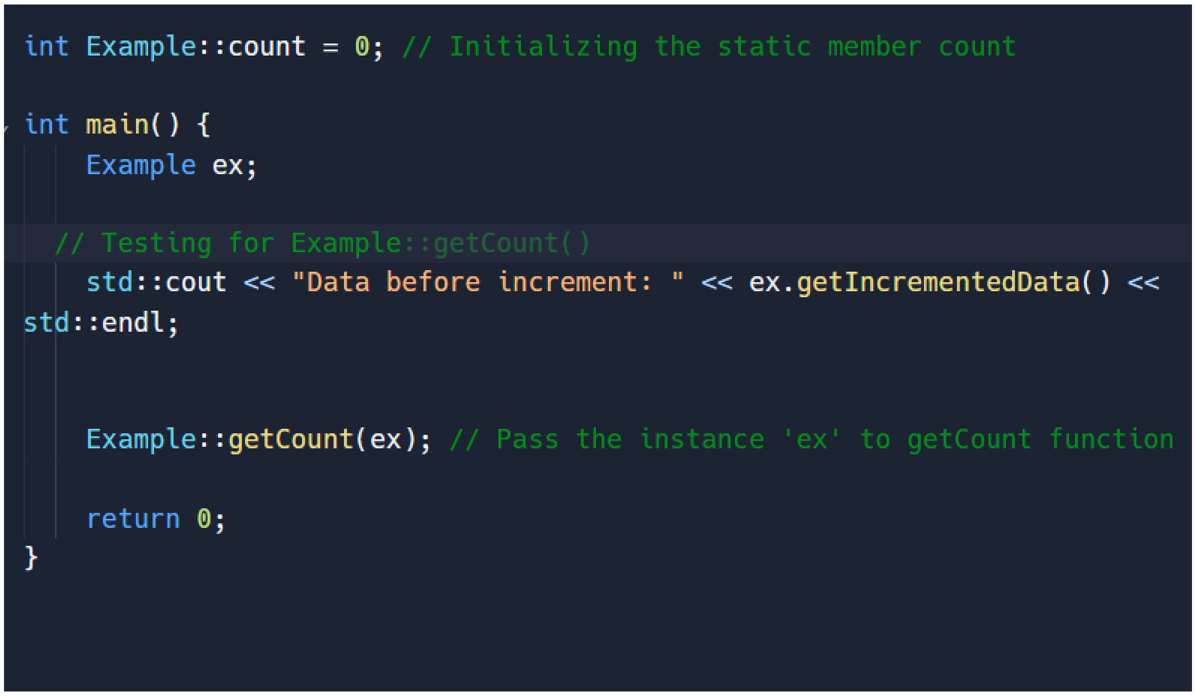
4. Find the errors in the following class and explain how to correct them. Please test it in main function

```cpp
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
```

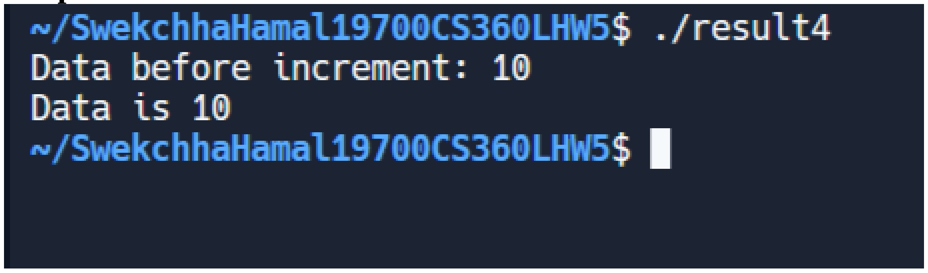


```
#include <iostream> // Including the iostream header for cout and endl

class Example {
public:
    Example(int y = 10) : data(y) {

    }

    int getIncrementedData() const {
        // Cannot modify data as it's const, so return data without
incrementing
        return data;
    }

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

private:
    int data;
    static int count;
};


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



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

int main() {
    Example ex;

  // Testing for Example::getCount()
    std::cout << "Data before increment: " << ex.getIncrementedData() << 
std::endl;


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

    return 0;
}
```

Output:



```
~/SwekchhaHamal19700CS360LHW5$ ./result4
Data before increment: 10
Data is 10
~/SwekchhaHamal19700CS360LHW5$ 
```