**#Question-1:**

#include <iostream>

#include <vector>

using namespace std;

void rvrs(vector<int>& vct) {

int n = vct.size();

for (int i = 0; i < n / 2; ++i) {

int temp = vct[i];

vct[i] = vct[n - i - 1];

vct[n - i - 1] = temp;

}

}

int main() {

vector<int> numbers = {1, 2, 3, 4, 5};

cout << "Original vector: ";

for (int num : numbers) {

cout << num << " ";

}

cout << endl;

rvrs(numbers);

cout << "Reversed vector: ";

for (int num : numbers) {

cout << num << " ";

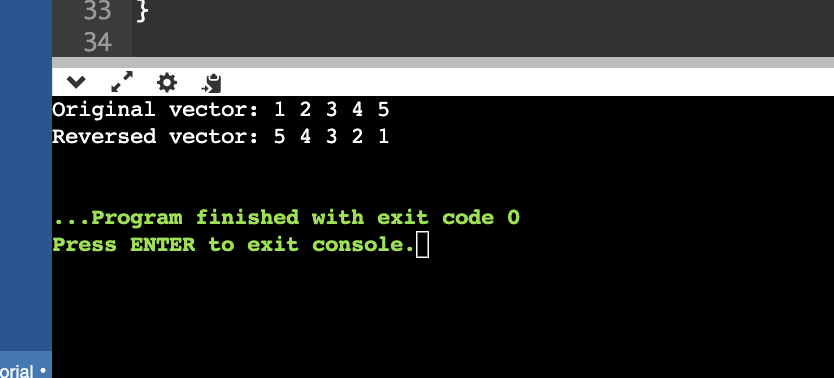
}

cout << endl;

return 0;

}

**Output:**



**#Question-2:**

#include <iostream>

#include <vector>

using namespace std;

void printDiagonal(const vector<vector<int>>& vals) {

int n = vals.size();

cout << "Given Matrix:" << endl;

for (int i = 0; i < n; ++i) {

for (int j = 0; j < n; ++j) {

cout << vals[i][j] << " ";

}

cout << endl;

}

cout << "Diagonal elements: ";

for (int i = 0; i < n; ++i) {

cout << vals[i][i] << " ";

}

cout << endl;

}

int main() {

vector<vector<int>> vals = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

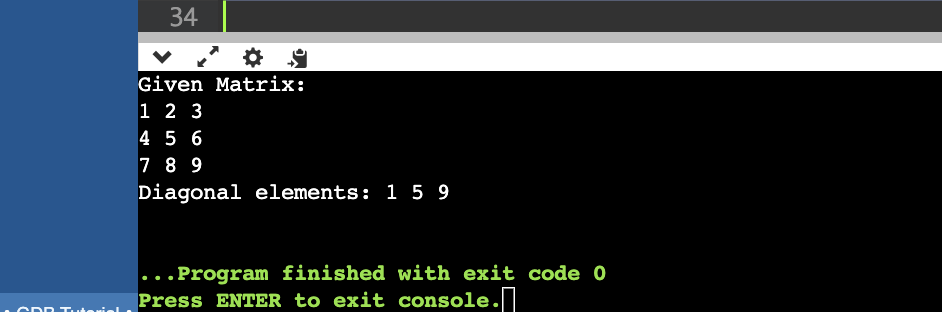
};

printDiagonal(vals);

return 0;

}

**Output:**



**#Question-3:**

#include <vector>

#include <algorithm>

#include <iostream>

class Tensor {

public:

void sort(std::vector<int>& v) {

std::cout << "Given array: ";

for(const auto& i : v) {

std::cout << i << " ";

}

std::cout << std::endl;

std::sort(v.begin(), v.end());

std::cout << "Sorted array: ";

for(const auto& i : v) {

std::cout << i << " ";

}

std::cout << std::endl;

}

};

int main() {

Tensor tensor;

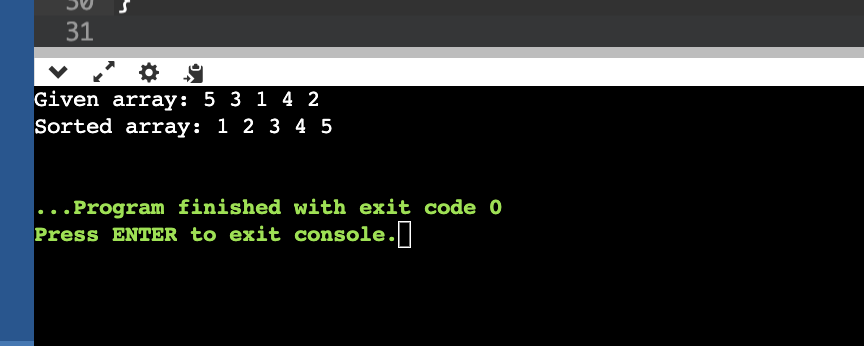
std::vector<int> v = {5, 3, 1, 4, 2};

tensor.sort(v);

return 0;

}

**Output:**



**#Question-4:**

There are several errors here:

Error 1: Missing #include <iostream> Directive. The code lacks the necessary #include <iostream> directive, which is essential for using input/output stream functions like cout. To fix this, add #include <iostream> at the beginning of the code.

Error 2: Incorrect Initialization of data Member Variable. In the constructor, the member variable data is initialized with the value of y. However, the increment operation in the getIncrementedData() function suggests that data should be incremented. To correct this, initialize data to y without the increment operation in the constructor.

Error 3: Incorrect Implementation of getIncrementedData() Function.

The getIncrementedData() function currently returns the value of data after incrementing it. However, the function is declared as const, which means it should not modify the object’s state. To fix this, remove the increment operation and return the incremented value

Error 4: Incorrect Implementation of getCount() Function.

The getCount() function attempts to print the value of data, which is not a static member. Additionally, the count variable is declared but not used. To correct this, remove the attempt to print data and return the actual count (which should be a static member)

**Corrected Code:**

#include <iostream>

class Example {

public:

Example(int y = 10) : data(y) { ++count; }

~Example() { --count; }

int getIncrementedData() { return data++; } // Changed to post-increment operator

static int getCount() { std::cout << "Count is " << count << std::endl; return count; }

private:

int data;

static int count;

};

int Example::count = 0;

int main() {

Example ex;

int originalValue = ex.getIncrementedData(); // Get the original value

int incrementedValue = originalValue + 1; // Increment the original value

std::cout << "Original value of data: " << originalValue << std::endl;

std::cout << "Incremented value of data: " << incrementedValue << std::endl;

Example::getCount();

return 0;

}

**Output:**

