Creative Software Design, Assignment 14-2

Deadline: 2024-12-03 23:59 (No score for late submission)

- Submit your homework by uploading your zip file to the LMS assignment section. Below is an example.

```
13178_Assignment1-1_2024123456.zip
├─ 1.cc
├─ 2.cc
├─ 3.cc
└─ ...
```

- Your zip file name should follow this format:
  13178_Assignment[Assignment-number]_[Student-ID].zip
  - Ex. 13178_Assignment1-1_2024123456.zip

- Source files should be named as **<filename>.cc** *or* **<filename>.cpp**

- **You must submit your solution in the zip file before the deadline.**

1. Using Lambda Functions for Data Processing

A. Task Overview

Write a C++ program that reads a list of integers from the user and stores them in a std::vector<int>. The program should use lambda functions to calculate and print the sum of all elements, find and print all even numbers in the vector, and sort the vector in descending order and print the sorted list.

B. Requirements

1. Reads a list of integers from the user and stores them in a std::vector<int>.
2. Uses lambda functions to:
   i. **Calculate and print the sum** of all elements.
   ii. **Find and print all even numbers** in the vector.
   iii. **Sort** the vector in **descending order** and print the sorted list.
3. **Do not use** any named helper functions; all logic should be within lambdas.
4. Use std::for_each with a lambda to print elements of a container.

C. Hints

1. Utilize std::accumulate, std::copy_if, and std::sort with lambda functions.
2. Remember to capture variables appropriately in your lambdas.

D. Example output of your program (Bold text indicates user input):

```
Enter integers (type 'q' to finish): 5 2 8 1 3 7 4 q
Sum of all elements: 30
Even numbers: 2 8 4
Sorted list in descending order: 8 7 5 4 3 2 1
```

E. Submission file: one C++ source file (File name: 1.**cc** or 1.**cpp**)

2. Implementing Move Semantics with a String Wrapper Class

A. Task Overview

You need to create a class StringWrapper that manages a dynamically allocated C-style string. Demonstrate move semantics by creating instances of StringWrapper, moving one instance to another using std::move, and showing that the source object is left in a valid but empty state after the move.

B. Create a StringWrapper class

1. This class manages a dynamically allocated C-style string
2. Private members:
   i. char* str
3. Public members:
   i. Default constructor: the default value of str is nullptr.
   ii. StringWrapper(const char* s): Parameterized constructor that takes a const char*.
   iii. Copy constructor.
   iv. Move constructor.
   v. Copy assignment operator.
   vi. Move assignment operator.
   vii. Destructor
4. Overload the << operator to output the string content.
   i. Prints "null" if the str is nullptr.

C. Requirements

1. Use std::move to facilitate move operations.
2. Ensure that after a move, the source object's internal pointer is set to nullptr.

D. Hints

1. Use std::strlen and std::strcpy for string operations.
2. **Copy constructor** and **copy assignment operator** create deep copies of the string.
3. **Move constructor** and **move assignment operator** transfer ownership of the string.

E. **Example of the `main()` function:**

```
int main() {
    StringWrapper sw1("Hello World");
    std::cout << "sw1: " << sw1 << std::endl;

    // Copy constructor
    StringWrapper sw2 = sw1;
    std::cout << "sw2 (after copy): " << sw2 << std::endl;

    // Move constructor
    StringWrapper sw3 = std::move(sw1);
    std::cout << "sw1 (after move): " << sw1 << std::endl;
    std::cout << "sw3 (after move): " << sw3 << std::endl;

    // Copy assignment operator
    StringWrapper sw4;
    sw4 = sw2;
    std::cout << "sw4 (after copy assignment): " << sw4 << std::endl;

    // Move assignment operator
    StringWrapper sw5;
    sw5 = std::move(sw2);
    std::cout << "sw2 (after move assignment): " << sw2 << std::endl;
    std::cout << "sw5 (after move assignment): " << sw5 << std::endl;

    return 0;
}
```

F. Example output of your program (Bold text indicates user input):

```
sw1: Hello World
Copy constructor called.
sw2 (after copy): Hello World
Move constructor called.
sw1 (after move): null
sw3 (after move): Hello World
Copy assignment operator called.
sw4 (after copy assignment): Hello World
Move assignment operator called.
sw2 (after move assignment): null
sw5 (after move assignment): Hello World
```

G. Submission file: one C++ source file (File name: 2.**cc** <u>or</u> 2.**cpp**)