Programming Exam 1

CMPE 126 Section 06 Lab Version 1
Date: 04/19/2024 9:00am-11:50am

This version of the exam is ONLY for students whose SJSU ID end in 0, 2, 4, 6, and 8.

General guidelines

- This exam will be for 2.5 hours only. Please use the last 20 minutes to complete your submission.
- Please submit your best work BEFORE 11:50am
- You may use external sources **with citation**

Academic Dishonesty

The following will be considered actions of academic dishonesty and will be reported without further discussion.

- Copying code:
  - You are NOT permitted to share code or reuse someone else's work.
- Discussing with others:
  - You are NOT permitted to discuss with your classmates or anyone else about the lab.

Remember to submit your OWN work and **cite all your sources**.

Expected outcome from this exam

The exam will test your skills and understanding of the following concepts:

- C++ programming skills
- Problem-solving skills
- Linked Lists
- Recursion
- Stacks

Exam in the next page…

Problem 1

1. Write the pseudocode using recursion to reverse a number digit by digit. E.g., if the input number is 1234, the output needs to be 4321. Put your pseudocode, **including all your sources**[1], in a file called `recursiveReverseNumber.pdf`.

2. Implement your pseudocode. You are provided a file called `recursiveReverseNumber.cpp` with a main function and the function declaration. You will need to include your implementation in this file. The main function includes some test cases.

Problem 2

In this problem, you will implement undo and redo functionality for a calculator that performs only add or subtract operations. A user's actions may perform add or subtract operation on the last output of the calculator. After each action, the user may perform an undo or redo operation until such an operation is possible. Undo operation invalidates the last operation performed, and redo performs the last undone operation. E.g., the user has already performed the actions as follows (calculator starts with a value 0):

i. Add 1
ii. Add 14
iii. Subtract 6
iv. Add 4

At this point, the result of the calculation is 13. Now, an undo operation undoes the most recent operation, i.e., "Add 4" is undone, and the calculator output becomes 9. Another undo operation undoes the operation before it, and the calculator output becomes 15. A redo operation redoes the last operation that was undone, i.e., calculator output becomes 9. Another redo operation makes the output go back to 13. At this point no more redo operations are possible.

You are provided with two files `Calculator.h` and `main.cpp`. `Calculator.h` consists of the declaration of the calculator class. `main.cpp` consists of test cases and expected behavior of the program. An important operation you will need to support is the ability to print all the operations performed (the history of all operations). You may choose to store this information in an array or a linked list. You may use any STL data structures or other libraries to store the operations.

You will need to create a file called `Calculator.cpp`, in which you will implement the functions declared in `Calculator.h` that supports the operations provided in `main.cpp`. Implement undo and redo operations using stacks. You may use the STL stack, or your own implementation of stacks.

For an A-level grade, implement a Calculator Set class consisting of a list of calculators by implementing the functions defined in `CalculatorSet.h` that is provided to you. You may choose to change the data structure used to store the calculators. Uncomment the last try catch block in `main.cpp` to test your code.

In the file called `Calculator.pdf`, include your ideas on the algorithm and implementation. Also **include all your sources in this file**.

[1] You risk losing all the points for the question if you do not list references

Rubric

1. [Core] Problem 1 (pseudocode)
2. [Advanced] Problem 1 (recursive function implementation)
3. [Core] Problem 1 – Coding skills (files compile and run fine)
4. [Core] Problem 2 – Basic functionalities of Calculator (add and subtract)
5. [Core] Problem 2 – Undo operation implemented correctly
6. [Advanced] Problem 2 – Redo operation implemented correctly
7. [Core] Problem 2 – History
8. [Core] Problem 2 – Coding skills (files compile and run fine)
9. [Advanced – required for A level] Problem 2 – Implemented CalculatorSet class

Passing criteria: 3/6 core tasks completed
C-level grades: Upto 6/6 core tasks completed
B-level grades: 2/3 advanced tasks completed
A-level grades: All advanced tasks completed

Submission instructions

Submit the following files in a zip file:

- `recursiveReverseNumber.pdf`
- `recursiveReverseNumber.cpp`
- `Calculator.h`
- `Calculator.cpp`
- `main.cpp`
- `Calculator.pdf`
- `CalculatorSet.cpp` (if implemented)
- `CalculatorSet.h` (if used)

Name your zip file ProgrammingExam2_XXXYYYXXX.zip, where XXXYYYXXX is your SJSU ID. E.g., if your SJSU ID is 111000111, your submission will be called ProgrammingExam2_111000111.zip.

Warning – If you reuse code from the Internet as-is without proper citation, you risk losing all points for this exam. Even if you refer to code on the Internet, copying entire functions as-is is not acceptable. Make sure your submission is your original work.