ITP 365: Managing Data in C++

Homework 2: itpPhone — Due: 9/15/2017 @ 11:59PM

Goal

You are the lead programmer for a revolutionary new smart phone called the itpPhone. You have tasked one of your junior programmers with the job of implementing the calculator and phone dialer app. However, the junior programmer in question is a graduate of UCLA, so unsurprisingly he is unable to complete the task. He has implemented the user interface, but he could not finish the backend due to a lack of understanding of data structures. You will leverage your knowledge of how to use Stacks and Maps to implement the required functionality.

Lab Setup

- Download and extract **HW2Start.zip** from Blackboard
- Unlike in Homework 1, the starting project has several code files in it already.
  - You will only need to edit **Calculator.h/cpp** and **PhoneSystem.h/cpp**
  - **Do not edit any other files**
- If you run the starting code, you will see a main menu with "Calculator" and "Phone" which correspond to the two parts of the homework
- Note that on Mac, there is occasionally an issue where all the buttons do not draw on screen. It's unclear what causes this. If you encounter the issue, simply stop and start your program in Xcode.
- At the top of the four files you will edit, make sure you add comments in the following format:

```
// ITP 365 Fall 2017
// HW2 – itpPhone
// Name: Tommy Trojan
// Email: ttrojan@usc.edu
// Platform: Windows
```

Part 1: Calculator class

- In this part, you will implement a RPN calculator
- In **Calculator.h**, you must add a member variable that is a stack of integers. We will use this stack to implement the RPN functionality
- In **Calculator.cpp**, you must implement the following member functions:
  - `pushNum` – Pushes the provided number onto the stack
  - `peekTop` – If the stack is not empty, peeks the top value on the stack
  - `doCalc` – Given the provided `CalcType`, pops the two values from the top of the stack, performs the calculation, and pushes this result back onto the stack. Note that if there are less than two values on the stack, `doCalc` should simply do nothing.
- To test your `Calculator` implementation, try the following input sequences:
  - `[1] [0] [ENTER] [1] [2] [ENTER] [5] [*] [+]` should yield `70`
  - `[5] [ENTER] [1] [0] [+] [7] [ENTER] [8] [-] [*]` should yield `-15`

- `[1] [0] [ENTER] [5] [/] [7] [ENTER] [3] [-] [2] [ENTER] [5] [+] [*] [*]` should yield 56

Part 2: PhoneSystem class

- In this part, you will implement phone number lookups using a pair of maps
- As this part is more complex than Part 1, it is worth more points
- In **PhoneSystem.h**, you must add three member variables:
  - Two maps where the key and value type will be std::string. The first map will be a map of area codes to their locations, and the second map will be a map of phone numbers to their contacts.
  - A std::string to keep track of the current phone number the user is dialing.
- In **PhoneSystem.cpp**, you must implement the following member functions:
  - Default Constructor
    1. First, read in the **areacodes.txt** file (which is in the same HW2 folder as all the .h/cpp files) and populate the map of area codes from this file. You must validate that the file exists – if it does not, emit an error using the Stanford error function.
    2. Next, read in the **contacts.txt** file and populate the map of contacts from this file. As with areacodes.txt, you must validate that the file exists.
  - `addDigit` – Given the provided digit, concatenate it to the current phone number. Note that you must take into account adding hyphens at the appropriate point in the number, in the standard `XXX-XXX-XXXX` format. Do not allow the phone number to contain more than 10 digits.
  - `backspace` – Removes the rightmost digit from the number. Conveniently, there is a `pop_back` member function for `std::string` that removes the last character from a string. Be sure to also remove any hyphens as appropriate.
  - `getNumber` – Returns the current phone number (so that it can be displayed)
  - `lookupNumber` – Returns one of three values:
    1. If the number is a known contact, return the name of the contact
    2. Otherwise, if the number begins with a known area code, return the location of the area code
    3. Otherwise, return `"UNKNOWN"`
- To test your `PhoneSystem` implementation, try the following input sequences:
  - `[8][1][9][2][9][8][2][5][7][0]` should yield Alan Mathis
  - `[3][1][0][8][6][7][5][3][0][9]` should yield Jenny
  - `[2][4][6][1][2][3][4][5][6][7]` should yield BARBADOS
  - `[1][2][3][4][5][6][7][8][9][0]` should yield UNKNOWN

A Note on Style

Style will be a small portion of the overall homework grade (5%). If you write spaghetti code, don't provide comments for function declarations, and so on, you will lose some points. We will similarly make sure you use the separate files correctly.

Deliverables

1. You must submit a ZIP file to Blackboard called **HW2Submit.zip**, using the designated Homework 2 submission link in the "Assignments" section.
2. Follow the same procedure to create the ZIP file as in Homework 1.

Grading

Item	Points
Calculator::pushNum	5
Calculator::peekTop	5
Calculator::doCalc	20
PhoneSystem::PhoneSystem	20
PhoneSystem::addDigit	10
PhoneSystem::backspace	10
PhoneSystem::getNumber	5
PhoneSystem::lookupNumber	20
Comments, style, and correct use of separate files	5
Total	**100**