IC210 AY16 6-Week Practicum Exam

Practicum Rules

- This is a multi section exam that will be given to different midshipmen at different times. As per USNAINST 1531.53A, you may NOT communicate about this exam with anyone using any medium until your instructor tells you that you can.
- You may use your written notes, your own code stored on your CS Department home directory, the course website, and online resources that are linked directly from the course website.
- You may not communicate with anyone other than your instructor while taking this exam.
- You may submit as many times as you like, your most recently submitted solution for a given part will be the one graded.
- Your most correct submitted solution will be graded out of 60 points. Your second most correct will be graded out of 25 points, your third most correct out of 15 points. So focus first on getting one problem completely correct. Programs should function correctly. Partial credit for incorrect solutions will be limited. Expect to get a zero for code that doesn't compile.

How to submit a solution

You will be using the submit script to submit your practicum solutions.

1. To submit your first solution give the command:

   ```
   submit p1p6wk p1p6wk.cpp
   ```

2. To submit your second solution give the command:

   ```
   submit p2p6wk p2p6wk.cpp
   ```

3. To submit your third solution give the command:

   ```
   submit p3p6wk p3p6wk.cpp
   ```

Problem 1

Write a program with source file named `p1p6wk.cpp` that provides a simple tip calculator. It offers two basic commands:

- `calc` for calculating how much to pay given the bill amount, (optionally) the number of people to split the bill amongst, and the tip percentage; and
- `test` for determining the tip percentage given the bill amount and the total amount to pay.

You may assume you are only given input formatted in the manner shown below - i.e. no error checking needed!

`~/$ ./p1p6wk`
`calc $1134.25 split 12 ways tip 18.5%`
`pay $112.007`
	The bill is $1134.25. We want to add an 18.5% tip then split the result 12 ways.
	(1134.25 + 1134.25 * 0.185)/12 = 112.007
	... and the answer is $112.007.

`~/$ ./p1p6wk`
`calc $87.39 tip 16.0%`
`pay $101.372`
	The bill is $87.39. We want to add an 16.0% tip (no splitting).
	(87.39 + 87.39 * 0.16) = 101.372
	... and the answer is $101.372.

`~/% ./p1p6wk`
`test $78.50 pay $90.00`
`tip 14.6497%`
	Paying $90.00 on a $78.50 check. The tip is the difference, so
	(90.0 - 78.50)/78.50 = 0.146497
	... so we'd be giving a 14.6497% tip.

Problem 2

Write a program with source file named `p2p6wk.cpp` that calculates the taxes you owe. You will read from the user a filename and an income amount (in whole dollars). The file contains tax bracket information for a specific year (we provide two examples: tableA.txt and tableB.txt). Your program will print output for each row — either the rate followed by "This is not you!" or the rate followed by "This is you! You owe $XXXXXX", as appropriate. Here are some sample runs:

```
10%     $0 to $9075
15%     $9076 to $36900
25%     $36901 to $89350
28%     $89351 to $186350
33%     $186351 to $405100
35%     $405101 to $406750
39.6%   $406751 to $9999999999
```

tableA.txt

```
~/$ ./p2p6wk
tableA.txt $80278
10% This is not you!
15% This is not you!
25% This is you! You owe $20069.5
28% This is not you!
33% This is not you!
35% This is not you!
39.6% This is not you!
```

The 25% bracket applies because the income ($80278) is between $36901 and $89350 (from tableA.txt). Our simple tax calculation is that 25% of $80278 is $20069.5.

```
~/$ ./p2p6wk
inXqB.txt $80278
File not found!
```

```
~/$ ./p2p6wk
tableB.txt $80278
10% This is not you!
15% This is not you!
25% This is not you!
28% This is you! You owe $22477.8
33% This is not you!
35% This is not you!
```

The 28% bracket applies because the income ($80278) is between $$78851 and $164550 (from tableB.txt). Our simple tax calculation is that 28% of $80278 is $22477.8.

1. You may assume the user inputs an income (in whole dollars) between $0 and $9999999999.
2. You may assume that the user's input is properly formatted, but not that the file they name exists in the current directory. If the file is not found, you must print the message "`File not found!`", and nothing more.
3. If the file exists, you may assume it is properly formatted, but your program must deal with *any* file formatted in the same way as the two example files.

Problem 3

```
width = 10
@X.........@
@......X...@
@....X....X@
@..X.......@
@........X.@
@.....X....@
```

file boardZ.txt

Write a program with source file named `p3p6wk.cpp` that plays a simple game. The user/player enters the name of a file containing a "board" for the game (we provide board files boardX.txt, boardY.txt, and boardZ.txt). The board is a rectangle representing the playing area. In between the @'s are *width* spaces. The .'s are safe spots, the X's are death traps. The player chooses a column number from 1 to *width*, and the game marches the player from the top of the board at that column position down until he either dies in a death trap, or makes it through safely. If the player dies, the game reports what step he died at: step 1 is the first row, step two the second, and so on. Otherwise it reports that the player survived. If the file named by the user does not exist, print the message "File not found!" and exit. If the user enters a column number outside the range 1..*width*, print the message "Invalid position!" and exit.

```
~/$ ./p3p6wk
foo.txt
File not found!
```

```
~/$ ./p3p6wk
boardZ.txt
Enter position between 1 and 10: 15
Invalid position!
```

```
~/$ ./p3p6wk
boardZ.txt
Enter position between 1 and 10: 7
You died on step 2
```

```
~/$ ./p3p6wk
boardZ.txt
Enter position between 1 and 10: 2
You survived!
```

```
~/$ ./p3p6wk
boardY.txt
Enter position between 1 and 30: 22
You died on step 17
```