CS106B Handout

Winter 2016 Feb 7th, 2016

Practice Midterm #2

This is an open-note, open-reader exam. You can refer to any course handouts, textbooks, handwritten lecture notes, and printouts of any code relevant to any CS106B assignment. You may not use any laptops, cell phones, or handheld devices of any sort. You will be graded on functionality—but good style helps graders understand what you were attempting. You do not need to include any libraries and you do not need to forward declare any functions. You have 2 hours. We hope this exam is an exciting journey ☺.

Last Name: _____

First Name: _____

Section Leader: _____

I accept the letter and spirit of the honor code. I've neither given nor received aid on this exam. I pledge to write more neatly than I ever have in my entire life.

(signed) _____

		Score	Grader
1. Mystery	[10]	_____	_____
2. Blamo	[15]	_____	_____
3. Templatize	[15]	_____	_____
4. Recursive Read	[10]	_____	_____
5. Palindromes	[15]	_____	_____
Total	**[65]**	**_____**	

Problem 1: Tracing C++ programs and big-O (10 points)

Assume that the functions **`Mystery`** and **`Enigma`** have been defined as follows:

```
int enigma(int n) {
    int sum = 0;
    for(int i = 0; i < 2 * n; i++) {
        sum += mystery(i);
    }
    return sum;
}

int mystery(int n) {
    int sum = 0;
    for(int i = n; i >= 1; i /= 2) {
        sum += i;
    }
    return sum;
}
```

(a) [3 points] What is the value of **`mystery(16)`**?

(a) [3 points] What is the value of **`enigma(5)`**?

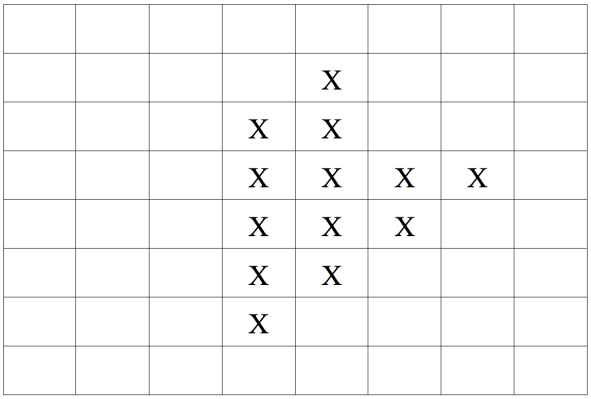
(c) [4 points] What is the worst case computational complexity of the **`enigma`** function expressed in terms of big-O notation, where *N* is the value of the argument **`n`**? In this problem, you may assume that **`n`** is always a nonnegative integer.

Problem 2: Grids, Vectors, Stacks, and Queues (15 points)

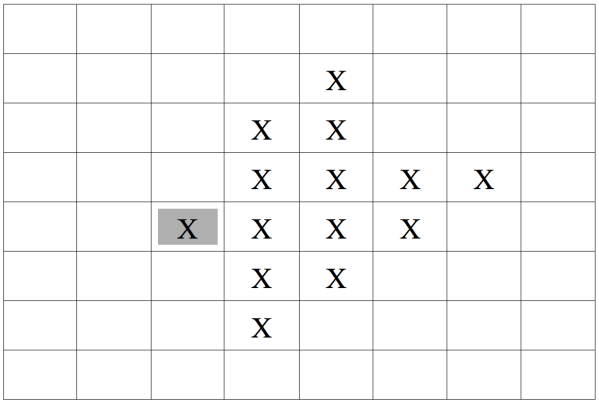
Blamo is a board game in which players take turns placing tiles on a grid and getting points based on the length of horizontal and vertical chains formed by each move. Your job is to write the following function, which calculates the score associated with a valid move:

```
int computeScore(Grid &board, int row, int col);
```

This function takes as input **board** which represents the state of a Blamo board, and **row** and **col** which corresponds to the location of the move on board. The function computeScore calculates the score of a Blamo move consisting of placing a single tile on the board at the position (row,col). For instance, the following is a possible state of the Blamo board (where X indicates the location of a tile):

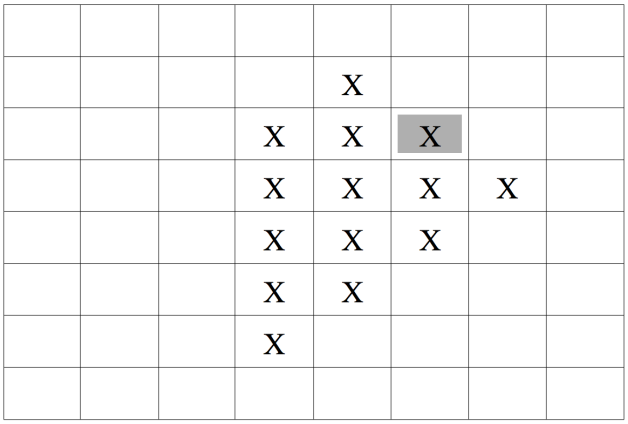


A move involves placing a single tile on the board. The score of that move is equal to the sum of the lengths of the horizontal and the vertical chains of tiles that are adjacent to the tile placed. For example, consider a move consisting of placing a tile in the shaded cell of the board below



This move would be worth 5 points: 4 from the horizontal chain of 4 tiles, and 1 from the vertical chain of 1 tile.

A better move to make would be this:



This move is worth 6 points: 3 for the vertical chain, and 3 for the horizontal chain.

The input board represents the state of the Blamo board such that: board[y][x] is true if there is a tile at row y and column x of board and false if there is a not a tile there. row and col represent the position on board that a tile was placed. You may assume row and col are within the bounds of board.

(space for the answer to problem 2 appears on the next page)

Answer to problem 2:

```
int computeScore(Grid &board, int row, int col) {
```

Problem 3: Maps and Sets (15 points)

A templatized string is a string that is used as a template for dynamically generated content. The idea is that given a template, you "fill in" some information to customize the string for a given context. Your job is to write a function

```
string customizeTemplate(string templateStr, Map &tokens);
```

that accepts as input string **templateStr** which is a string of text with 0 or more tokens which need to be replaced with their corresponding values in tokens. Tokens in **templateStr** are represented using the syntax (key), where parentheses are used to indicate that the text inside of the them is a token that needs to be replaced. For example, consider the following input:

```
templateStr = "(friend) is now (verb) you on (websiteName)"
tokens =
```

Key	Value
"friend"	"Megan"
"verb"	"following"
"websiteName"	"Google Plus"

Given this input, **customizeTemplate** should return the following string:

```
"Megan is now following you on Google Plus"
```

For this question you may make the following assumptions:

1. The only use of parentheses in templateStr is to denote a token. For instance, you don't need to worry about the following inputs:
    a. "))))) I am a string with random parentheses ((((((“
    b. "I am a string with a ((TokenSurroundedByParentheses))"
2. You may assume that the map tokens contains every token that appears in templateStr
3. You may assume that tokens in templateStr only consist of letters and numbers
4. The string you return must be identical to templateStr except for the replaced tokens. This means, for example, that you should not add or remove any whitespace.

(space for the answer to problem 3 appears on the next page)

Answer to problem 3:

```
string customizeTemplate(string templateStr, Map &tokens) {
```

Problem 4: Recursive Read (10 points)

For each call to the following recursive function, write the output that is produced as it would appear on the console. Recall that relational operators like < and > compare strings by alphabetical order; for example, "a" is less than "b".

```
void recursionMystery7(string s) {
   if (s.length() <= 1) {
      cout << s;
   } else {
      string first = s.substr(0, 1);
      string last = s.substr(s.length() - 1, 1);
      string mid = s.substr(1, s.length() - 2);
      if (first < last) {
         recursionMystery7(mid);
         cout << last << toUpperCase(first);
      } else {
         cout << "[" << first << "]";
         recursionMystery7(mid);
         cout << last;
      }
   }
}
```

Call	Output
`recursionMystery7("abcd");`	
`recursionMystery7("leonard");`	
`recursionMystery7("breakfast");`	

Problem 5: Recursive Exploration (15 points)

A palindrome is a word that reads the same from left to right as right to left. For example, the following are palindromes:

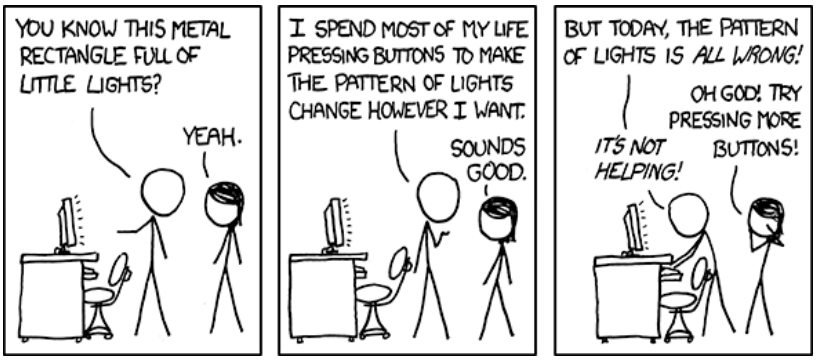
- Madam
- Civic
- Racecar

Your job is to write the function **`printAllPalidromes`** which recursively prints out all palindromes consisting of the letters A through Z of length at most **`numChars`**. You may assume that **`numChars`** is positive.

```
void printAllPalindromes(Lexicon & lex, int numChars){
```

Problem 6: Optional Fun Problem (0 points)

Make your own paneled style stick figure comic pontificating or commenting about CS, computers, life, etc. Here's one for reference:



This problem isn't required and there are no points associated with doing any work here, but any great gems will be very much appreciated by the course staff :).