CS106B Winter 07-08

Handout #15 January 17, 2008

Section Handout #2

Problem 1: Vectors (AKA C++ ArrayLists…)
Say we are writing the next version of a nifty new mail reading program and want to use a Vector (interface in reader appendix) to store all the data. The following structure is used to hold the data of an email message:

```
struct eMailMsg {
  string to;        // i.e. "professor@stanford.edu"
  string from;      // i.e. "student@stanford.edu"
  string message;   // body of message
  string subject;   // i.e. "CS106 Rocks!"
  int date;         // date email was sent
  int time;         // time email was sent
};
```

a) How would you declare a `Vector` that stores `eMailMsg`s?

b) Write a function `RemoveSpam` that takes a vector containing elements of type `eMailMsg` and removes all elements whose subject begins with the string "SPAM".

c) How could you modify the `to` field of the `eMailMsg` structure so that it can hold the email addresses of an arbitrary number of recipients of an email? With the modification in place, given an `eMailMsg email`, how would you access the last address listed in the `to` field?

Problem 2: Queues
Write a function

```
void ReverseQueue(Queue<int> & q);
```

that reverses the elements in the passed in queue. (Hint: Is there another class that could make doing this a lot easier?)

Problem 3: Using the Scanner and Stack classes

```
<html><b><i>CS106 rules!</i></b></html>
```

Web browsers use stacks to track html tags such as `<b>`, `<i>` or `<html>`. Every html tag must be matched by an equivalent closing tag -- `</b>`, `</i>` or `</html>`. Mozilla is looking for programmers to help implement this feature in the next version of Firefox and you, armed with your newly acquired knowledge of classes, decide to volunteer for the job. Using the Scanner class and the Stack class, write the following function:

```
bool IsCorrectlyNested(string htmlStr);
```

You can assume that all the tags in the html string will be correctly formed. That is, once you see an angle bracket, it will be followed by the remainder of a complete and well-formed tag (So, nothing like "`<<html>`").

Problem 4: Map Warm-up
Write a function:

```
char MostFrequentCharacter(ifstream &if, int &numOccurrences);
```

that given an input file stream, returns the character that occurs the most frequently and stores the number of times it occurs in the reference parameter `numOccurrences`. To write this function, first start by scanning through the file stream, analyzing each character and storing an updated count in a map. Then, after you've built this table, iterate over it to find the character that occurred the most often.

Problem 5: Minesweeper
In the game of Minesweeper, a player searches for hidden bombs on a rectangular grid. The game board is represented by a grid of booleans marking bomb locations. A grid value is true if there is bomb at that location, false otherwise. Here is an example grid:

(0,0)

T	F	F	F	F	T
F	F	F	F	F	T
T	T	F	T	F	T
T	F	F	F	F	F
F	F	T	F	F	F
F	F	F	F	F	F

Given such a grid of bomb locations, the function MakeGridOfCounts constructs a new grid of integers storing the count of bombs in each neighborhood. The neighborhood for a location includes the location itself and its eight adjacent locations. In the returned grid, each value will be a number from 0 to 9. If passed the boolean grid above, MakeGridOfCounts returns:

(0,0)

1	1	0	0	2	2
3	3	2	1	4	3
3	3	2	1	3	2
3	4	3	2	2	1
1	2	1	1	0	0
0	1	1	1	0	0

The function is passed the grid by reference, and returns an int grid created as described.

```
Grid<int> MakeGridOfCounts(Grid<bool> & bombLocations)
```