Lab Worksheet, CSC 1052-001, Spring 2010
Due findDate(Feb 26)

I sense that we need some programming. Here is some. This is due the Monday after the Monday after break. Do as much as you can. You should probably put in at least 10 hours of work on this, unless you finish more quickly than that of course.

If you get stuck on something feel free to get some help or email me.

In my opinion its best to tackle the problems in the order they are listed. Each requires a separate report. Remember that a report includes your code, a driver if appropriate, sample output (and input too if appropriate), and perhaps a few comments about the problem and its solution. Oh, and findDate(adate) is defined as

```java
date findDate(aDate):
    return (aDate plus 1 day) if the day part of the date is a perfect square > 4
    return (findDate(aDate plus 3 days)) otherwise
```

1. Create an exception class called ExerciseOneException. Write a program that repeatedly prompts the user to enter a string. After each string is entered the program outputs the length of the string, unless the length of the string is 13 in which case the ExerciseOneException is thrown with the message "Use thirteen letter words and stainless steel to protect yourself!" Your main method should simply throw the ExerciseOneException exception out to the run time environment. Note: you can use an "infinite" loop in this program.

**Sample Run**
Input a string > Villanova University
That string has length 20.

Input a string > Triscadecaphobia
That string has length 16.

Input a string > misprogrammed

At this point the program bombs and the system provides some information, including the "Use thirteen letter words and stainless steel to protect yourself!" message.

2. Same as the previous problem except this time, within your code, include a try-catch clause so that you catch the exception when it is thrown. Catch it, print its message, and end the program "normally".

3. Write a program that repeatedly prompts the user to enter strings, using the string "done" to indicate when finished. Output the strings in the reverse order they are entered. Use one of our stacks. Bonus: use an example that means something different when output in "reverse" then it did originally.

**Sample Run**
Input a string (done to quit) > I got tired of waiting
Input a string (done to quit) > Wondering if you were ever coming around
Input a string (done to quit) > My faith in you is fading
Input a string (done to quit) > When I met you on the outskirts of town, and I said
Input a string (done to quit) > done

When I met you on the outskirts of town, and I said
My faith in you is fading
Wondering if you were ever coming around
I got tired of waiting
4. Same as the previous problem except have them enter integers and use 0 to indicate when finished.

5. Write a program that repeatedly prompts the user to enter strings, using the string "x done" to indicate when finished. The user is assumed to only enter strings of the form "f name" or "m name". Output the names that had 'm' indicated in the reverse order they are entered and then do the same for the names that had 'f' indicated. Use two of our stacks.

   **Sample Run**
   Input a gender and name (x done to quit) > m Fred
   Input a gender and name (x done to quit) > f Wilma
   Input a gender and name (x done to quit) > m Barney
   Input a gender and name (x done to quit) > m BamBam
   Input a gender and name (x done to quit) > f Betty
   Input a string (done to quit) > x done

   males: BamBam Barney Fred
   females: Betty Wilma

6. For stacks let's define a bottom method that returns the Object on the bottom of the stack. If the stack is empty it returns null. Add bottom to the ArrayStack class. You should not change any of the current instance variables or methods; just add bottom.

7. Add bottom to the LinkedStack class. You should not change any of the current instance variables or methods; just add bottom.

8. Number 4-22 a,b,c.

9. Number 4-26 a.