CS 100 Assignment 2: Conditions, Loops, and MATLAB
Summer 2001
Due in lecture, Friday July 6

Every now and then go away, have a little relaxation,
for when you come back to your work your judgment will be surer. – Leonardo da Vinci

1. Objectives

Completing all tasks in this assignment will help you:
- Spot programming mistakes (bugs)
- Practice declaring, assigning, and using variables
- Write conditional statements and loops in Java
- Plot and manipulate data in MATLAB

First, skim the whole assignment. Then carefully read all instructions before starting. You must use a **fixed-width** font (e.g., Courier or Monaco) for your homework and you must **staple** your assignment together. We will be strict about these things from now on. If you are not sure whether you are using a fixed-width font, ask a consultant or TA to check it for you (sometime *before* the assignment is due!).

2. Identifying Coding Errors (25 points)

Create a new text file and include identifying information at the top (name, CUid, date, assignment).

The following program was written by a very careless programmer. Read through the program carefully and **write a short paragraph** describing what you think the program is supposed to do. (5 points)

```
public class BadProgram
{
  Public static void main(String args[])
  {
    int i2, i3, I3;

    // Prompt user for max number
    System.out.println("How many?");
    int n = SavitchIn.readDouble();

    // Nested loop
    for (i2=1; i2 < n; i2++)
    {
      for (I3=1; i3 < n; i3++);
      {
        // Print stuff
        if (i2 % i3 = 0) System.out.print("1");
        else System.out.print(0");
      }
      System.out.println();
    }
  }
}
```

Now, list **five mistakes** that the programmer has made, by copying the relevant line of code into your file and explaining what error it contains. For each one, provide **a corrected version** of the erroneous line of code. A "mistake" is an error in the program that will

cause it to behave incorrectly (e.g., fail to compile, compute an incorrect value, provide the wrong output). How do you know when the program is doing "the wrong thing"? The program should behave as advertised in any provided comments. If it does not, that behavior qualifies as a mistake. (10 points)

Example (you should not include this in your list):
1. Original code: **`Public static void main(String args[])`**
   Problem: Java is case-sensitive, so "Public" should be "public".
   Fixed code: **`public static void main(String args[])`**

As discussed in lecture, the compiler can catch several errors, but there are some errors that it will not detect. You, as a human, should be able to identify several mistakes *in addition to* what a compiler would notice.

In addition, the author of the program has not used very good programming style. Identify **five examples of bad programming style** (according to what we have discussed in lecture and what has been covered in the assigned reading) and **describe how you would fix them**. (10 points)

Example (you should not include this in your list):
1. Style problem: The "nested loop" comment is uninformative. What does the loop do? Why is it nested?
   To fix it: Change the comment to describe what the nested loop does (include information about what the two variables stand for). [Note: your answers should be more precise than this. In this case, you should include the corrected version of the comment. We have omitted it here so that we don't give away too much about what the program does.]

You should turn in a **printout of a text file** that contains the answers to these questions as well as your identifying information (name, CUid, date, assignment number).

3. Conditional statements (10 points correctness, 10 points style)

Our goal is to come up with a program that asks the user to input a month by number (e.g., January is 1, February is 2 …) and then tells the user how many days are in that month. (Note that we have to ask the user for some more information if the input is month 2. Ask if the year is a leap year; the user should enter either 'y' for yes or 'n' for no).

1. First, **write an algorithm** that you can follow to solve this problem (hint: working some examples (of different input values, and what the output should be) on paper is a good way to start). **Number** the algorithm steps, and specify the **INPUT**, **PROCESS**, and **OUTPUT** portions of your algorithm. Of course, the program should also check to make sure the input is valid; if not, it should exit with an appropriate error message.
2. Download **NumberOfDays.java** from the course website (and create a new CodeWarrior project for it).
3. Add a **comment block** to the top of NumberOfDays.java with the following information (yes, you will lose points if you forget to do this):
   Name: (your name)
   ID: (your Cornell ID)
   Date: (current date)
   Assignment: 2, NumberOfDays.java
4. Put your algorithm in NumberOfDays.java in the form of a set of comments (at the point indicated in the file by "Algorithm").
5. Now go back and **fill in the missing code** to implement the **PROCESS** part of your algorithm (you will note that the **INPUT** and **OUTPUT** parts have already been written).

(Hint: you should make use of both conditional statements we've covered.) Be sure to follow good style (commenting, variable naming, structure, etc.).
6. Print your NumberOfDays.java and turn it in at lecture.

4. Loops (15 points correctness, 15 points style)

For this problem, we will write a program that asks the user for a number (maxNum) and then reports, for every number from 1 to maxNum, whether or not the number is prime. The program will also write out its results to a text file, which you will use in the next section of this assignment.

1. Download **PrimeNumbers.java** from the course website (and create a new CodeWarrior project for it).
2. Add a **comment block** to the top of PrimeNumbers.java with the following information:
   Name: (your name)
   ID: (your Cornell ID)
   Date: (current date)
   Assignment: 2, PrimeNumbers.java
7. This time, the algorithm for PrimeNumbers.java has been provided for you. Read it carefully so you know what should be implemented.
8. Now go back and **fill in the missing code** to implement the **PROCESS** part of the algorithm. Be sure to follow good style (commenting, variable naming, structure, etc.).
9. It is important to test your program with different inputs. What would be good values to choose as test input for this program? Do the tests with your program. If any of the tests fail, go back and correct your program. (You don't need to turn anything in for this part.)
10. Print your PrimeNumbers.java and turn it in at lecture.

5. MATLAB (25 points)

MATLAB is a wonderful tool for analyzing large amounts of data. Not only can it easily compute a wide range of statistics, but, as they say, "A picture is worth a thousand words."[1] MATLAB has powerful support for plotting data in a variety of ways.

1. After running PrimeNumbers.java, you'll notice a file has been created in the same directory with the name "primes.txt". You can view this file with any text editor. Run the program with "50" as the input, then copy primes.txt to a new file named **primes-50.txt**.
2. Run MATLAB. Read the contents of primes-50.txt into a variable called **p50**. **Plot p50 using a histogram**. This will show peaks where the prime numbers occur. Add an informative title to the graph that includes your name and CUid, label the axes appropriately, and then **print** it.
3. Do the same thing with "1000" as the input. (Run PrimeNumbers.java, copy primes.txt to a new file named primes-1000.txt, and plot it with a histogram in MATLAB. Then print the graph.)
4. Create a new text file that includes the list of MATLAB commands you executed to create the two graphs. Print this file as well.
5. Turn in the two graphs and the text file of MATLAB instructions at lecture.

[1] And the two most important words in MATLAB are **doc** and **help**.