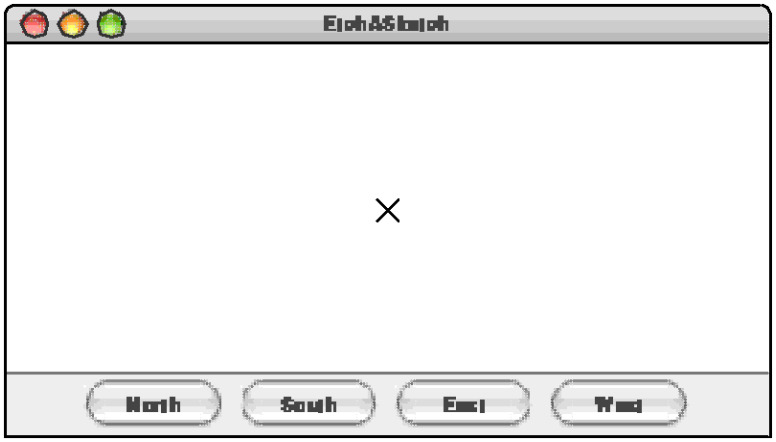
Extra Practice Final Problems

This handout is intended to give you practice solving problems that are comparable in format and difficulty to those which will appear on the final examination. A solution set to these practice problems will be published online on Thursday the 11th. Please try to take these in test conditions!

Problem —Graphics and interactivity (15 points)

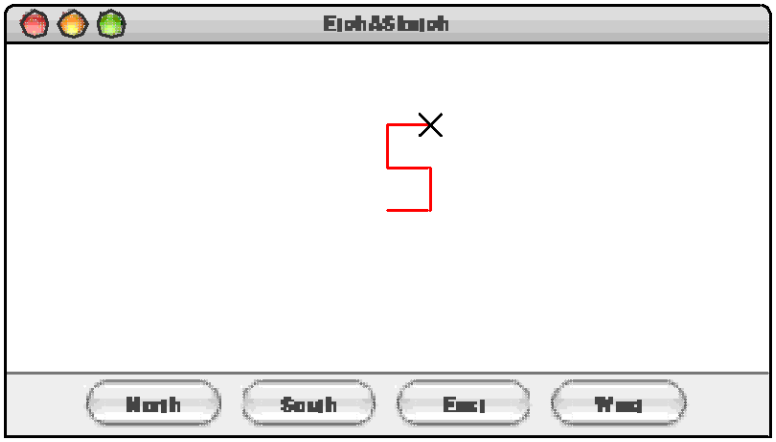
Write a **`GraphicsProgram`** that does the following:

1. Add buttons to the south region labeled **`"North"`**, **`"South"`**, **`"East"`**, and **`"West"`**.
2. Create an X-shaped cross ten pixels wide and ten pixels high.
3. Adds the cross so that its center is at the center of the graphics canvas. Once you have completed these steps, the display should look like this:



4. Implement the actions for the button so that clicking on any of these buttons moves the cross 20 pixels in the specified direction. At the same time, your code should add a red **`GLine`** that connects the old and new locations of the pen.

Keep in mind that each button click adds a new **`GLine`** that starts where the previous one left off. The result is therefore a line that charts the path of the cross as it moves in response to the buttons. For example, if you clicked **`East`**, **`North`**, **`West`**, **`North`**, and **`East`** in that order, the screen would show a Stanford "S" like this:



Problem —Strings (15 points)

There is no gene for the human spirit.

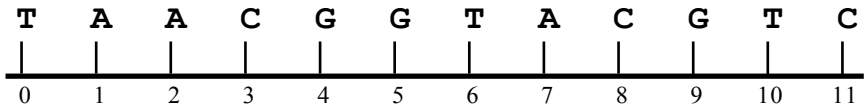
—Tagline for the 1997 film *Gattaca*

The genetic code for all living organisms is carried in its DNA—a molecule with the remarkable capacity to replicate its own structure The DNA molecule itself consists of a long strand of chemical bases wound together with a similar strand in a double helix. DNA's ability to replicate comes from the fact that its four constituent bases—adenosine, cytosine, guanine, and thymine—combine with each other only in the following ways:

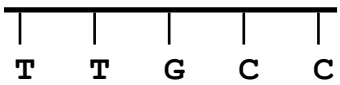
- Adenosine on one strand links only with thymine on the other, and vice versa.
- Cytosine links only with guanine, and vice versa.

Typically, biologists abbreviate the names of the bases so that each is represented by its initial letter: `A`, `C`, `G`, or `T`.

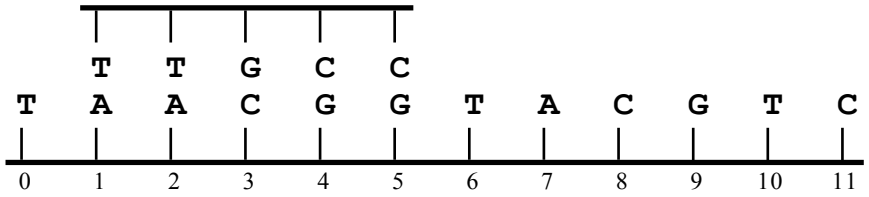
Inside the cell, a DNA strand acts as a template to which other DNA strands can attach themselves. As an example, suppose that you have the following DNA strand, in which the position of each base has been numbered as it would be in a Java string:



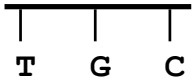
Your mission in this problem is to determine where a shorter DNA strand can attach itself to the longer one. If, for example, you were trying to find a match for the strand



the rules for DNA dictate that this strand can bind to the longer one only at position 1:



By contrast, the strand



matches at position 2 or position 7.

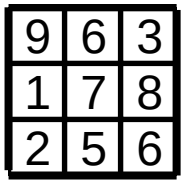
Write a method `findFirstMatchingPosition(shortDNA, longDNA)` that takes two strings of letters representing the bases in DNA strands. The method should return the first index position at which the first DNA strand would bind onto the second, or –1 if no matching position exists.

Thus, calling `findFirstMatchingPosition("TTGCC", "TAACGGTACGTC")` should return 1, `findFirstMatchingPosition("TGC", "TAACGGTACGTC")` should return 2 (the first match), and `findFirstMatchingPosition("CCC", "TAACGGTACGTC")` should return –1.

Problem —Arrays (10 points)

In the last several years, a new logical puzzle from Japan called *Sudoku* has become quite a hit in Europe and the United States. In Sudoku, you start with a 9×9 grid of numbers in which some of the cells have been filled in with digits between 1 and 9, as shown in the upper diagram to the right. Your job in the puzzle is to fill in each of the empty spaces with a digit between 1 and 9 so that each digit appears exactly once in each row, each column, and each of the smaller 3×3 squares. Each Sudoku puzzle is carefully constructed so that there is only one solution. In this case, the unique solution is shown in the lower diagram.

By the end of CS 106B, you could write a program to solve Sudoku puzzles. In CS 106A, we have to be content with a simpler task. Suppose that you wanted to write a program to check whether a proposed solution was in fact correct. Because even that task is too hard for a ten-minute array problem, your job here is simply to check whether the upper-left 3×3 square contains each of the digits 1 through 9. In the completed example shown at the right, the 3×3 square in the upper left contains exactly one instance of each digit and is therefore legal. If, however, you had made a mistake filling in the puzzle and come up with



instead, the solution would be invalid because this square contains two instances of the value 6 (and no instances of the value 4).

Your task in this problem is to write a method

```
private boolean checkUpperLeftCorner(int[][] matrix)
```

which looks at the 3×3 square in the upper left corner of the matrix and returns **`true`** if it contains one instance of each of the digits from 1 to 9. If it contains an integer outside of that range or contains duplicated values, **`checkUpperLeftCorner`** should return **`false`**.

In writing your solution, you may assume that the variable passed in as the **`matrix`** parameter has already been initialized as a 9×9 array of **`int`**s. You are also completely free to ignore the values outside of the 3×3 square in the upper left. Those values will presumably be checked by other code in the program.

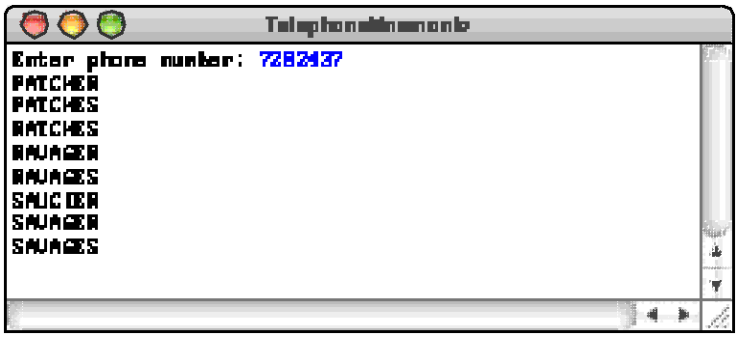
Problem —Java programming (15 points)

On a standard telephone, each of the digits except for 1 and 0 is associated with a group of three or four letters as shown in the following diagram:

For certain telephone numbers, it is possible to find words that correspond to the digits in that number. For example, at one time in history it was possible to get the time of day in the Boston area by dialing **`"NERVOUS"`** (6378687). Since these words are usually easier to remember than the number, they serve as **mnemonics** for the number.

Suppose that you have a text file named **`words7.txt`** that contains a list of all English seven-letter words, one word per line, entirely in uppercase. Write a program that accepts a string of seven digits and prints out every seven-letter word that corresponds to that number. For example, your program should be able to duplicate this sample run:

If more than one word can be formed from a given string of digits, your program should list them all. In terms of the number of mnemonics it generates, the most prolific phone number is 7282437, which corresponds to eight English words:



Caution: Don't try to go for the most efficient solution here. The simplest strategy is to read the file line by line and display every line that matches the supplied phone number.