Goals of this assignment

- Practice incremental development of a project
- Practice the use of libraries
- Practice the use of loops and conditionals
- Learn how to write a makefile

You may use Pair Programming for this assignment. Instructions are on the website (under Assignments)

Part 1

Turtle Graphics

This problem is adapted from Exercise 7.23 of the textbook.

Imagine a mechanical turtle that walks around the room under the control of a C++ program. The turtle holds a pen which can be in one of two positions, UP or DOWN. If the pen is DOWN, the turtle traces out shapes as it moves; while the pen is up, the turtle can still move but doesn’t draw anything.

In this problem, you will simulate the operation of the turtle and create a computerized sketchpad.

The drawing board (floor) where the turtle moves and draws is implemented as a square grid of cells that can be FULL (of color) or EMPTY. It is assumed that the grid is $20 \times 20$. The cell in the upper left corner has coordinates $(0,0)$ and the one in the lower right corner has coordinates $(19, 19)$ A DrawingBoard class has been defined and will be made available to you. It should be used to initialize the board, and provides you with functions that allow you to fill or erase a cell and to render the drawing. You must use the class and you should not modify the provided files in any way.

Implementation

You will need to define at least one more class that will represent a graphic/drawing. It will keep track of the current position of the pen, whether it is up or down, the direction the turtle is facing, and of course the drawing board.

The turtle should always start at position $(0,0)$, facing down, and holding the pen UP.

Your graphic class should have functions for raising and lowering the pen, changing direction, drawing the sketch and anything else you deem necessary.
You should also write a driver with a main() that will allow the user to enter “turtle commands” to create a drawing. These commands are based on the numeric keypad (for easier memorization):

<table>
<thead>
<tr>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Face down (south)</td>
</tr>
<tr>
<td>8</td>
<td>Face up (north)</td>
</tr>
<tr>
<td>4</td>
<td>Face left (west)</td>
</tr>
<tr>
<td>6</td>
<td>Face right (east)</td>
</tr>
<tr>
<td>5 x</td>
<td>Move x spaces in the direction you are facing. x is any non-negative integer</td>
</tr>
<tr>
<td>9</td>
<td>Raise the pen</td>
</tr>
<tr>
<td>3</td>
<td>Lower the pen</td>
</tr>
<tr>
<td>7</td>
<td>Render</td>
</tr>
<tr>
<td>1</td>
<td>End the program</td>
</tr>
</tbody>
</table>

You may add more commands if you like, but you should not change the key mapping of the commands mentioned above (otherwise we won’t be able to test your program easily).

Your main() function should be short and delegate its work to other, well-defined functions.

Finally, you must provide a makefile for this project. It should be able to allow separate compilation as needed. Use what you learned in Tuesday’s (1/17) lab. You may also refer to the make tutorial found at the Links section of the class website.

Provided files

The files containing the DrawingBoard class interface and implementation are located on T-Lab, in the path ~b11/PA2/DISTR. Login to your Tlab account and create a directory hierarchy as shown below:

```
mkdir PA2
cd PA2
mkdir part1
mkdir part2

cd to part1 and copy all the necessary files over from the b11 directory:

cp ~b11/PA2/DISTR/* .
```

(Note the dot at the end of the command line)

Testing

~b11/PA2/TESTS on T-lab contains a number of test cases that you can use to try out your program. Feel free to write additional test cases and post them to the newsgroup.

~b11/PA2/BIN contains a sample executable that you can use to see how the program is expected to behave.

Part 2

Problem description

Build the “times table” drill-and-practice program described in Exercises 6.35, 6.36, and 6.37 (page 320). The program should run until the student has answered 10 questions correctly.

Feel free to embellish and extend the program. You can use different response texts than those given. You can use more than four different responses, etc.

Write modular object-oriented code. The main() function should be quite short, like the main() function in the GradeBook program in Figure 6.5. Most of the code should be in a TimesTable class. The TimesTable class should have:

- A constructor that takes two integer arguments:
  - How many problems to generate, e.g., 10, and
  - how big the numbers in the problem can be, e.g., 9.
- A method that runs the drill
- A method that prints the final results
- Any other methods you think would be useful

Provided files

There are no starter files for this part. However, all files that you write as part of the answer should be in the part2 directory you created earlier. You only need to submit the solution to Exercise 6.37.

Commenting guidelines

Follow the rules described in PA1. You may also have a look at the provided code which is commented according to these guidelines.

Submission guidelines

When you finish the project, remove any executables (rm a.out) and then cd to the PA1 directory that you created. Type the following to archive your files:

tar cvf your_name.tar part1 part2

E-mail this tar file to b11@cs.northwestern.edu AND to yourself (so you can verify that the file was attached correctly).

You may resubmit your code as many times as you want, within the deadline (plus any late hours that you wish to use). If you do, write RESUBMIT in the subject line of your email. You should expect a confirmation within 12 hours of submission.

Start your work early and take advantage of the newsgroup and office hours. This project is longer than PA1.