COSC 101 Homework 1 Fall 2014

Due date: **Wednesday, September 10, 11:55pm**

Picobot

This homework is designed to give you your first exposure to designing algorithms and writing a program. You may find this assignment challenging, so we encourage you to start early and seek help if you get stuck.

This first assignment does not involve any Python. Instead, you are writing in the language of Picobot. The Picobot language was introduced in class; this homework also includes a reference that describes the language (see the file named `picobot_reference.pdf`).

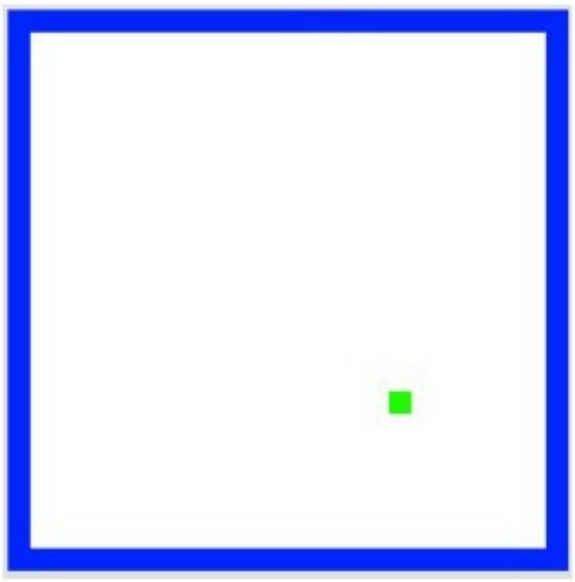
The Picobot language is much simpler than Python. On the one hand, this means that you won't get bogged down learning language syntax. On the other hand, the language is so simple that it can be challenging to express what you want to do in such simple terms. This is one of the central challenges of programming, even in Python, and while you may find it frustrating at times, we hope that you also experience the sense of satisfaction when you figure out how to get that 'bot to zoom all around the room!

To complete this homework, you will need to try out your programs on a Picobot simulator, which can be found here:

http://cs.colgate.edu/~mhay/picobot/

Warning— The simulator works for Firefox and Chrome browsers. It is not compatible with Microsoft's Internet Explorer, even though it might at first appear to work. **Do not use IE.**

Overview and Setup



Your task is to write two Picobot programs. The first program should successfully navigate the empty room (shown on left). The second program should successfully navigate the maze room (shown on right). Both of these rooms are available in the Picobot simulator: click the arrow buttons on either side of the word "MAP".

When you have completed both programs, you have to submit them on Moodle using a file in **plain text format**. Use the following setup.

- Find the file `hw1.txt` that has been provided for you.
  - `hw1.txt` is included with this homework files.
  - `hw1.txt` is a file in *plain text format*, hence the `.txt` file suffix.
- Open `hw1.txt` using
  - Notepad on Windows or
  - TextEdit on Mac or any other simple text editor.
- Save your Picobot program in your `hw1.txt` file as you are working with the online simulator.

To fulfill the assignment, complete the following two programs. Once you have terminated the two required programs try the optional **Challenge Problem** described below.

Empty Room

Write a Picobot program for the **empty room**.

1. Build on the activities we did in class think about a strategy (i.e., an *algorithm*) for completing the empty room. Draw on a diagram on paper to help your thoughts.
2. Once you have a rough idea of your strategy, start translating this idea into a Picobot program on paper. Write rules to move Picobot. Consult the `picobot_reference.pdf` that comes with this homework to use the correct syntax of a Picobot program.
3. Once you have a set of rules try them inside the Picobot simulator. Is Picobot navigating through the entire room using your program? Understand the relation between the rules you designed and Picobot motion in the simulator. (Remember that *the simulator does not save your work*: if the browser is closed or crashes, your work will be lost. Be sure to save a copy of your work in `hw1.txt` as you go.)
4. When you think you have a working program for the empty room, run your program several times in the simulator. Click the "Reset" button to have Picobot start in a new randomly selected location. Remember that your program *must* work regardless of Picobot's starting location.
5. Add some comments to your program (see **Grading** below).

Maze

Write a program to navigate the **maze**.

1. The recommended strategy for solving the maze is something called "the right hand rule." This is explained in detail in the `picobot_reference.pdf` (see the section titled **Solving the maze**). Make sure you understand the right hand rule before attempting to translate it into Picobot program.
2. Translate the right hand rule into a Picobot program. Try it on a piece of paper and don't hesitate to check your program incrementally.
3. Once you think you have a solution be sure to try your maze program several times in the simulator.
4. Add some helpful comments to your program (see **Grading** below).

Submission

In the file `hw1.txt`

- write each program in the appropriate place and
- fill in the "header" at the top of the file.

Upload `hw1.txt` to Moodle before the deadline.

Grading

Your assignment will be graded on two criteria:

1. Correctness: each program should cover the room regardless of start location. [80% = 30+50]
2. Program design and style: when we get to Python, style and program design become more significant. For this assignment, the only style requirement is that you include *comments* that explain what each *state* means. Write exactly **one comment per state** that explains at a high level what you are trying to accomplish in that state. [20%]

   Here is an example. This program has two states and therefore has two comments (Please do **not** write a comment for every single rule!):

```
# State 0: go N as far as possible, then switch to state 1
0 x*** -> N 0
0 N*** -> X 1

# State 1: go S as far as possible, then switch to state 0
1 ***x -> S 1
1 ***S -> X 0
```

Challenge Problem (OPTIONAL)

Challenge problems are entirely optional extensions to the homework. If you complete them successfully, you are rewarded with a sense of accomplishment and a small number of extra points on the homework. They are intended for students who want to explore a little further; only pursue the challenge problem after you have successfully completed the homework.

At heart, CS fundamentally tries to answer questions of complexity: to show that problems are easier than initially thought, or, sometimes, to prove that they can't be handled with fewer resources.

You might think about how efficient your solutions are, both in terms of the number of states used and in terms of the number of rules. There are other ways to measure efficiency as well (e.g., speed).

The challenge problem is to create as efficient a solution as possible for each room:

- for the empty room, see if you can use only 6 rules
- for the maze, see if you can use only 12 rules

Just for fun, you might try to write a program for the diamond-shaped room (available at the Picobot simulator). You do not need to turn in this program as it is not part of the challenge problem.