CS256 — Homework 3

January 27, 2016

Due: Wednesday, February 10, 2016 before midnight (100 points)

Description

For this homework assignment, you may work with a partner. If you choose to do this, follow the instructions in the submission section to make sure both people have proper access to the same code repository.

We will be writing an interpreter/compiler for our own programming language. However, this won't be like a normal programming language. Our language will only have 8 different symbols, each corresponding to a different command (or instruction). We'll call this the Crazy programming language, because anyone who tries to write a program using it must be crazy.

A program in the Crazy language consists of a sequence of our 8 possible symbols. Any other symbol encountered will simply be ignored. The program terminates when it has passed the final command in the sequence.

The language will execute these commands on an array of 1000 `int` cells all initialized to zero that acts like a read/write tape. We'll call this array `tape`. There will be a movable *data pointer* that starts pointing at index 0 of the array that we'll call `dp`.

The commands of our language will work as follows:

Symbol	Meaning
`+`	Increment the value pointed to by `dp`
`-`	Decrement the value pointed to by `dp`
`>`	Move `dp` to the right
`<`	Move `dp` to the left
`:`	Print out the `int` stored in the location pointed to by `dp`
`.`	Print out the value stored in the location pointed to by `dp` as if it were a `char`
{	If the value pointed to by `dp` is zero, instead of moving to the next instruction, jump forward to the instruction *after* the corresponding }
}	Jump back to the corresponding {

Your interpreter will have three different modes distinguished by passing command line arguments:

1. If the user passes no command line arguments, your interpreter will be in *interactive mode*. In this mode, it will run a loop that prints out the current tape contents around `dp` and `dp`'s current location, asks the user to supply some Crazy commands, then runs those commands. You can have it exit if the user types in "exit".
2. If the user passes two command line arguments, they must be done in one of the following ways:
   (a) Pass `-c` then a file name. Your interpreter will then read the contents of the file passed as a Crazy program, compile it to C++, and output it via `cout`.

(b) Pass `-e` then a file name. Your interpreter will then read the contents of the file passed as a Crazy program and execute it directly, then quit. No output should be generated unless the program itself contains `.` or `:` commands.

You should implement the compile mode first because it is the easiest, then try the other two modes.

I would recommend writing separate functions to print out the tape contents, executing commands, etc.

For executing the commands, you should use recursion to handle loops. When you need to execute a loop, there are a few steps:

1. Determine the position of the end of loop character.
2. Decide if you should execute the loop. If not, move forward to the position found above.
3. If yes, then determine the position of the first character of the loop body.
4. Using the two positions you found, recursively execute those as a Crazy program, then reset your execution to the opening brace of the loop to repeat the process. As you can see, by rescursively executing the loop body as a subprogram, it will also handle any nested loops appropriately.

Include comments at the beginning of your source code file that contain your name, the homework assignment number, and the date that you completed the assignment. Include both names if working with a partner. For example, my submission's comments might look like this:

```
// Nick Pantic and Second Person
// Homework 4
// Completed 2/20/2016
```

Submission

1. If you are working with a partner, choose one person to create the project on `codebank.xyz`. The second person should not create a second project. The one creating the project should follow the instructions to create the project below then the second person should follow the remaining instructions for getting access to the project.
2. Create a project on https://codebank.xyz named `CS256-HW3`. Follow this naming convention precisely including case.
3. On the project page, click *Settings* to the left.
4. From the *Settings* page, click *Members* to the left.
5. On the Members page, click *Add Members*.
6. Add the second partner of your group with at least Developer permission.
7. After creating the project, the second partner should clone the repository with the following command:

   ```
   git clone https://codebank.xyz/firstpartner/CS256-HW4.git
   ```

   so that they have a folder referencing the same repository.
8. Now, whenever someone *pushes* changes to the server, the other person can *pull* them using:

   ```
   git pull origin master
   ```
9. It's a good idea to run the `pull` command whenever you start working on the project to make sure you get the other person's changes.

10. Make sure to stay organized, don't have both people modify the same code at the same time or you will have to resolve merge conflicts because `git` won't know which code to keep.
11. You and your partner can use the features on `codebank.xyz` to help communicate with each other such as issue tracking and commit comments.