CSCI 4160 Project1

Due: see class calendar

Goal:

This assignment serves several purposes:
to be familiar with COOL: classroom object-oriented language. COOL documents, including manual, examples, and interpreters for different platforms (Windows and Linux), can be found in the Github repository. To be familiar with COOL, please read chapters 3-11 in the manual.

Description:

This assignment asks you to write a short Cool program. The purpose is to acquaint you with the Cool language and to give you experience with some of the tools used in the course.

A machine with only a single stack for storage is a stack machine. Consider the following very primitive language for programming a stack machine:

Command	*Meaning*
int	push the integer int on the stack
+	push a '+' on the stack
e	evaluate the top of the stack (see below)
d	display contents of the stack
x	stop

The 'd' command simply prints out the contents of the stack, one element per line, beginning with the top of the stack. The behavior of the 'e' command depends on the contents of the stack when 'e' is issued:

- If '+' is on the top of the stack, then the '+' is popped off the stack, the following two integers are popped and added, and the result is pushed back on the stack.
- If an integer is on top of the stack or the stack is empty, the stack is left unchanged.

The following examples show the effect of the 'e' command in various situations; the top of the stack is on the left:

Stack before	*stack after*
+ 1 2 5 . . .	3 5 . . .
1 + 3 . . .	1 + 3 . . .

You are to implement an interpreter for this language in Cool. Input to the program is a series of commands, one command per line. Your interpreter should prompt for commands with >. Your program need not do any error checking: you may assume that all commands are valid and that the appropriate number and type of arguments are on the stack for evaluation. You may also assume that the input integers are unsigned. Your interpreter should exit gracefully; do not call abort() after receiving an x.

You are free to implement this program in any style you choose. If you wish, you may copy A2I class defined in atoi.cl and List and Cons classes in list.cl to your solution file so that you can use them to perform string to integer conversion and manipulate lists. Both files could be found in example folders in class repository.

Sample session
The following is a sample execution of your solution. To run the COOL interpreter, open a command line window (terminal in Linux) and type the following command:
%cool_interpreter stack.cl

```
>10
>2
>+
>d
+
2
10
>e
>e
>d
12
>x
```

COOL program successfully executed.

How to submit: Submit your source file to D2L dropbox of project1.

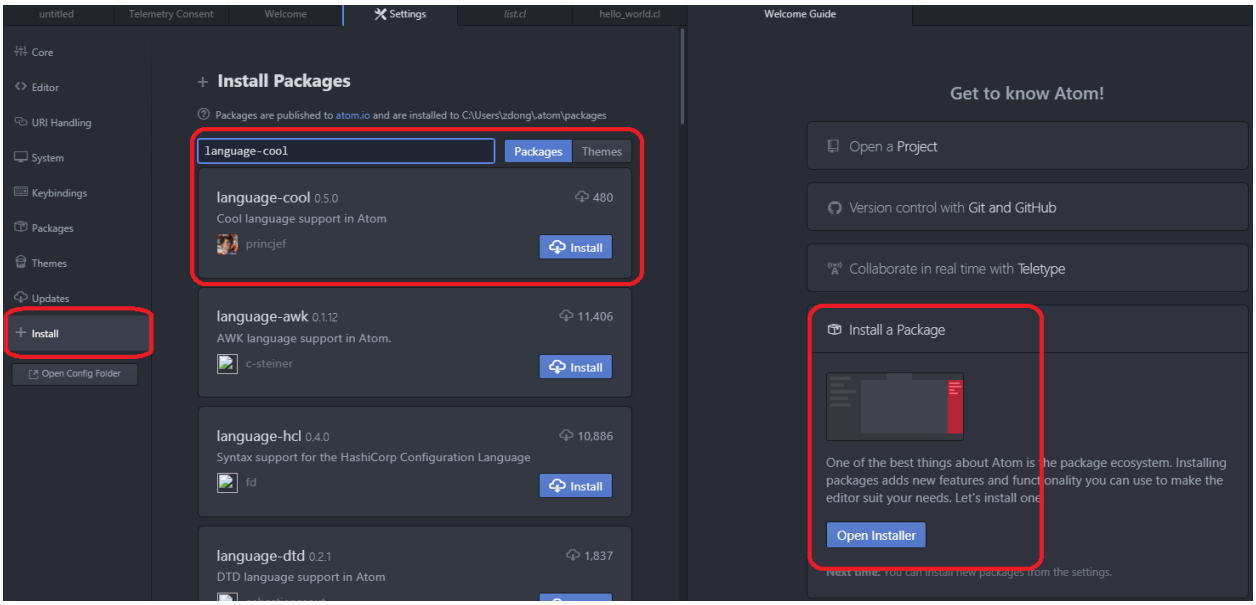
Where to download files for this project?
There are two ways to download the repository that contains the COOL documents as well as projects.

- Open a web browser to access the following webpage: https://gitfront.io/r/zdong-mtsu/cxBi197e9nLJ/CSCI-4160/
- You can use the following command to access the github repository:

  Git clone https://gitfront.io/r/zdong-mtsu/cxBi197e9nLJ/CSCI-4160.git

  It will copy all repository files to a "CSCI-4160" folder. In the future, if a new project is assigned, what you need to do is to enter the folder "CSCI-4160" and type command: "git pull" to retrieve latest version from the github repository to your local folder. To use this approach, you need to make sure **git** is installed in your computer.

An optional tool that may be of use to you
You can use ATOM editor which has a language-cool package adding syntax highlighting to Atom text for the Cool (Classroom Object Oriented Language) language. Please go here: https://atom.io/ to download the version for your own platform. To install the language-cool package, open ATOM, click Help | Welcome Guide and choose Install a package on the popup panel. Type "language-cool" to search the package and install it.



After installation, your COOL program will look like the following:

```
class Cons inherits List {

   car : Int; -- The element in this list cell

   cdr : List;  -- The rest of the list

   isNil() : Bool { false };

   head()  : Int { car };

   tail()  : List { cdr };

   init(i : Int, rest : List) : List {
      {
   car <- i;
   cdr <- rest;
   self;
      }
   };

};
```