CS320 Assignment #1

Purpose

This assignment is designed to familiarize you with GIT and Virtual Machines and C programming.

Requirements

This assignment consists of one major requirements.

1) Developing solutions to the problems below.
2) Your assignments will be checked, submitted and graded electronically.

Problem

1) I fully expect that you will **RTFM** (Read the friendly manual). There is documentation for lots of neat stuff on this thing you may have heard about called the interwebs. Many times you will be able to find answers to your questions on the magic box you are reading this very document on!

   Demonstration: https://lmgtfy.com/?q=Can+I+haz+programming%3F

   If you have trouble: http://lmgtfy.com/?q=how+to+get+better+at+searching+google

   No… I am serious about this. I'll even start you off with your first cat. If this isn't your first, then you can add it to your collection.

   For some of you this will be difficult. Here is a chipmunk:

   Good luck.

2) PLEASE READ INSTRUCTIONS BEFORE MAKING ACCOUNT!
   You must signup for an account at gitlab.com:
   IMPORTANT
   You will be sharing your username with me. Please make it school appropriate.

   http://www.gitlab.com

   We are going to use gitlab for this class because it is both a standard fully functional git management webportal and has unlimited private repositories for you store your assignments. We could use the more standard github.com, however you would not be able to make private repositories (and this makes your code available for all students). All of your projects/repositories for this class should be set of private when you create them. Please note that any members you add to the repository is tracked and recorded (thus if you give another student access to a repository I will know about it and it will be recorded as cheating).

3) Setup a Debian[1] based linux system. If you do not have an additional computer or do not want to permanently install linux on one of your machines you can setup a virtual computer for your linux system to run on. I recommend VirtualBox[2]. You can find a number of tutorials online with the correct search terms. If you wish to use another linux distribution or virtualization software you may. I will be testing on a Debian distribution with default packages.
4) Setup a project named "CS320Assignment1" under your account. Then you can add a new project.
5) Clone[3] this repository to your computer (or your virtual machine, or both). I recommend the software SourceTree for a graphical user interface on windows or osx. In your linux installation on virtualbox you will need to learn how to use the command line interface (Learning the git command line tools is essential to being developer with hireable skills in the future). After this repository is cloned to your local machine (you should probably do this in your virtualbox), create a file called README.md and put your Full Name and Email (each on their own line, as they appear in blackboard) in the file. Commit the file with an appropriate commit message (don't forget to push the commit to the server). You will also complete the readme to contain the information required by the assignment later.

---

[1] Recommended. Mint: https://www.linuxmint.com/ or Ubuntu: http://www.ubuntu.com/

[2] VirtualBox: https://www.virtualbox.org/

[3] How to use GIT: https://rogerdudler.github.io/git-guide/ We will be creating the repositories on the server through the web interface, so you do not need to do that step on your local computer. If you are using sourcetree I recommend: https://github.com/GSoft-SharePoint/Dynamite/wiki/Getting-started-with-SourceTree,-Git-and-git-flow

6) Create the following C programs/libraries. Your programs will follow the naming scheme specified: progX_Y.zzz Where X is the assignment number, Y is the program/library number for the assignment and zzz is the file extension for the particular language we will be writing that code in. For example, the first program will be in the file prog1_1.c (Capitalization matters!). Every driver you write for this class will start by printing a standardized header. The header will look like this:

```
Assignment #X-Y, <NAME>, <EMAIL>
```

Only programs that have entry points will print the header. In this assignment part 1 (prog1_1.c will print a header, but prog1_2.c/prog1_2.h will not, prog1_3.c will).

Your program must use the strings used in the examples provided. You do not have any creative leeway in the prompts or the responses. Most of the grading in this course is done automatically and the autograder is *extremely* unforgiving.

Please ensure that you commit and push your submission files as well.

Not all of your programs will be a driver (have an entry point). Some of the programs you write will be libraries for other programs.

prog1_1

Create a C program that prompts the user to enter their name. Your program will then take in their full name (You do not know how many names they will enter) and print a greeting to them using their full name.

Example compilation:

```
gcc prog1_1.c -o prog1_1
```

Example execution:

```
./prog1_1
```

Example run (User input to STDIN highlighted with yellow):

```
Assignment #1-1, Scott Lindeneau, slindeneau@sdsu.edu
What is your name?
Firsty McLasty
Hello Firsty McLasty!
```

prog1_2

Create a header file (prog1_2.h) which will contain the following function prototypes and typedefs.

```
typdef struct stack{
      int* data;
      int size;
      int capacity;
} STACK;

STACK* MakeStack(int initialCapacity);
void Push(STACK *stackPtr,int data);
int  Pop(STACK *stackPtr);
void Grow(STACK *stackPtr);
```

Create a source file (prog1_2.c) that will implement the `MakeStack`, `Push`, `Pop` and `Grow` functionality. The `STACK` that is produced by the `MakeStack` function should be able to hold `initialCapacity` integers. The `Grow` function should double a stacks capacity without changing any of the values held by the `STACK`. The `Push` and `Pop` functions should work as expected for a stack. The `Push` function should `Grow` the stack if there isn't enough capacity to hold the pushed data. The `Pop` function should return -1 if there is no data in the `STACK`. Yes, this is probably incorrect behavior.

prog1_3

Create a driver program that takes a single command line argument N (in addition to the name of the program being executed). The command line argument N will be an integer number. If there is not a single command line argument your program should print an error message and quit. For valid inputs, your program will print a single right carrot take N lines of input from STDIN (entire line inputs). Your program will split each of these lines on a space delimiter. Your program should ignore multiple space characters (i.e. treat as many spaces in a row as a single space. It should also ignore leading and trailing spaces. For example
`"    Hello         World    "`
should only result in two tokens, "Hello" and "World". Neither token should have spaces in the string. If there are exactly two tokens **AND** the first token is the string "`push`" then your program should convert the second token to an integer and push the result onto your `STACK`. If there is exactly one token **AND** the token is the string "`pop`" then your program should `Pop` a value off of the `STACK` and print it to `STDOUT` on its own line. All other input should be ignored. Your program should **only** print the assignment header, the error message, the carrot at the beginning of the input line or the result of a `Pop`. Your program can assume that all STDIN inputs will be less than 256 characters, the command line argument will be an integer (if it exists) and the second token in a two token input that has "`push`" as the first token will be an integer.

Example compilation:

```
gcc prog1_3.c prog1_2.c -o prog1_3
```

Example execution:

```
./prog1_3 8
```

Example run (User input to STDIN highlighted with yellow):

```
Assignment #1-3, Scott Lindeneau, slindeneau@sdsu.edu
> Push 4
> push 2
> abs
> pop
2
> pop
-1
> push 3 more
> push 2
> pop
2
```

Example execution:
```
./prog1_3
```

Example run (User input to STDIN highlighted with yellow):
```
Assignment #1-3, Scott Lindeneau, slindeneau@sdsu.edu

This program expects a single command line argument.
```

Example execution:
```
./prog1_3 5 data
```

Example run (User input to STDIN highlighted with yellow):
```
Assignment #1-3, Scott Lindeneau, slindeneau@sdsu.edu

This program expects a single command line argument.
```

7) After you have committed your files you will have to grant me developer access to your repository. To do this you must open the project on gitlab.com, and open the Members page that is under the settings menu on the right hand side of the page (It looks like a gear). Add me to your project by typing in my username: slindeneau and make sure that the project access is set to developer. When I grade your project I will push a new branch called "Grade" to your repository with the autograder output and any notes associated with your grade.

   You will need to do this for <u>every</u> assignment.

8) The last step involves verifying that everything is working correctly. Please go to: http://cs320.lindeneau.com
   Input your first and last name (as it appears on blackboard), your gitlab username and select the assignment you would like to grade. Verify that all of the parts of your program work as expected.

<u>DO NOT ENTER ANY PASSWORDS</u>

Passwords are not required for cs320.lindeneau.com to function. If your project does not get graded, you have not given correct access to the correct developer account on gitlab.

Additional Details

- You should be able to investigate any issues you have on your own and spend the time necessary to understand what is happening overall.

Assignment #1
Due 2/1 11:59pm

Late Policy

Late programs will be accepted with a penalty of 5% per day for seven days after due date. Turn in time will be determined by either date of rubric submission OR timestamp on graded files, whichever is later.

Cheating Policy

There is a zero tolerance policy on cheating in this course. You are expected to complete all programming assignments on your own. Collaboration with other students in the course is not permitted. You may discuss ideas or solutions in general terms with other students, but you must not exchange code. (Remember that you can get help from me. This is not cheating, but is in fact encouraged.) I will examine your code carefully. Anyone caught cheating on a programming assignment or on an exam will result in a zero for the class or assignment.