cs3157 – Advanced Programming
Summer 2006, lab #3, 20 Points
June 12, 2006

A lab in C...Hopefully you are already done with the first homework, so here is a very easy lab, we will start it in class together. Follow these step-by-step instructions. This lab must be submitted electronically (see instructions at the end of the lab) by **Wednesday** , June 14, 5 pm.

You need to include a README file with information about each file submitted and any other comments you want the TA to see.

Create a cs3157/lab3 subdirectory for this lab.

- Remember to follow the naming instructions!
- Remember to read through an entire step before you try to do it.
- Have fun!!!

Comments. (3 points)

Your code MUST BE COMMENTED! Include a HEADER COMMENT at the top of every file, containing your name, cs username, lab number, due date and name of the file, plus a short description of what the program does. Put additional comments in the code to explain what the code is doing. Comments are worth 0.1 points for each program!

Step 1. (4 points)

The first step is to write a C program named **step1.c** that prints out a four-line poem. This can be any poem you want. Make something up or look something up on the internet. Just make sure that the output contains four lines.

The text of the poem should be in a character array, and you should use a character pointer to refer to the poem for printing purposes.

First print it forward, then print it backwards (line by line).

Compile and test it to make sure the program works :-)

Remember from Monday's class:

```
compile: bash# gcc -o step1 step1.c
run: bash# ./step1
```

Step 2 (3 points):

The make command is a special tool (installed on local machines) which executes instructions on how to build your program. Those instructions are kept in a file named "Makefile". Please take a quick look at the format of the files: http://vertigo.hsrl.rutgers.edu/ug/make_help.html
Create a Makefile file with instructions on building and executing all the steps in the lab (also later steps). For example, typing in
make step1
will compile and execute the step1 program.
Hint: Modify a Makefile to do this.
Sample very simple Makefile:

```
step1:
        gcc -o step1 step1.c
```

Next create a rule to clean up after yourself.
make clean
will erase all temporary build files and executables. Be careful not to delete your .c files from your assignment here.
Hint: object files which end with .o are created from compile to linker, but then not used in execution

Step 3. (4 points)

Write a program called **step3.c** that asks for your name and then asks for your birthday and then prints out a welcome message and your birthday day (1-31) and month (1-12) in the format:
MM/DD. Create two int variables – one for the day and one for the month. Read in the numbers using the scanf function. Then use a printf statement to print nicely like this:
Hello Shlomo Hershkop
your birthday is 01/01.
The month and day are equal

Use the format string to display leading zeros if your birthday day or month is a single-digit number. Then compare the day and month numbers and print out a message saying which one is bigger or if they are the same. You should check for illegal input and print out an error message Compile and test it to make sure the program works.

Hint/Tip : regarding for loops in C
for loops in C are just like they are in Java, EXCEPT that you cannot declare the loop control variable inside the loop statement. In Java, you can do this:

```
for ( int i=0; i<10; i++ ) {
System.out.println( "i="+i );
}
```

That's **NOT ALLOWED** in C!!! In C, you have to do this:

```
int i; /*at the start of the code block*/
….
for ( i=0; i<10; i++ ) {
printf( "i=%d\n",i );
}
```

Also note that mathematical operators in C are the same as they are in Java.

Step 4: (Using the debugger)2 points

We will be using the c debugger. It is called gdb. Please do a quick search on the internet on how to use the debugger, and try to run any of your previous programs through it….I will be asking you questions about this on the final. For this step, write a paragraph or two describing your experience on getting it working/debugging any of the steps.

Step 5 (4 points)

Write a program in C called **step5.c** as follows:

1. Declare a constant called ARRSIZE using a #define command and sets it equal to 10.
2. Declare an array of ARRSIZE integers, as a local variable inside the main() function.
3. In C, functions are a 2 step process, you need to define a function signature, which tells the compiler what the function will look like, and then actually define the function. (more in next class).

   Define a function whose prototype is:
   void init(int arr[ ], int len, int x);

   This tells C that you are going to have a function which returns nothing (void) and takes an integer array as the first argument, and an integer as the second and third arguments. You could have declared the function also in the following manner:

   ```
   void init(int [],int,int);
   ```

   (NOTE: you need a semicolon after a function declaration).
   This function should initialize the contents of the array (arr), setting every entry in the array to the value of x. Note that len contains the length of the array. For example, if we called init(myarray,10,0), then myarray would contain 10 zeros (0) when the function exits.

4. Define a function whose prototype is:

   ```
   void print( int arr[ ], int len );
   ```

   The function should print out the contents of arr to the screen. Precede the output with the "[" character and follow it with the "]" character.

Complete the **main()** function by having it call the **init()** function followed by a call to the **print()** function.

Submit your lab.

See instructions online