University of Massachusetts - Boston **Ramin Dehghanpoor**
Programming in C **CS 240 – Summer 2018**

Homework Assignment

HW4: Multi-module programs and MAKE

Assigned: 22 June 2018 **Due:** 2 July 2018 by class time

1. Compile and link multiple modules together into a single executable file named "calcit".

Copy the makefile from my hw4 directory(/courses/cs240/sum18/ramin/GROUP/hw4) to your hw4 subdirectory. Then, create source code files for a MODIFIED version of the calcit program described in K&R, Section 4.3. You will use modules named: main.c, getop.c, stack.c, calc.h, and getch.c.

Your MODIFIED version of the calcit program will operate on integers instead of floating point variables and support BIT OPERATIONS: ^, %, ~, | and &. In order to do this, change the variable types and returns of functions such as "pop" so that type "int" is used in calculations. Then, add the operations ^, %, ~, | and & to support these binary operations. For example, if the operation is ^, then it is programmed as:

```
push(pop()^ pop());
```

Note that because we no longer allow double type returns, we can deal only with integers and must use the atoi() function instead of atof(). Look for atoi() in the index, also try "man atoi"--this is a C library function, and you are not required to code it. But you need to be careful when you are converting a negative number since the bit representation will be quite different from the positive number.

NOTE THAT YOU MAY USE THE LIBRARY FUNCTIONS USED IN THE BOOK FOR THIS PROGRAM, BUT YOU MUST BE ABLE TO PROGRAM THESE LIBRARY FUNCTIONS YOURSELF ON AN EXAM.

Use the infix string ~(((292 %16) +(292/16)*16) ^ 292) as the script test case. Your program need not understand infix arithmetic though -- you must convert the test expression given above to reverse polish before feeding it to your program (just create a file named “calc.in” with the proper sequence and use it as stdin).

For this problem, turn in a hardcopy of a “typescript1” file as usual.

Experiment with make and touch.

Prepare another file "makefile2" that is a copy of makefile except remove the command lists for the .o rules (we want to prove that they are superfluous). Leave the command list in the rule for calcit. In this case, make will try to build the .o files according to some built-in implicit rules which includes $(CC) and $(CFLAGS) (see https://www.gnu.org/software/make/manual/html_node/Implicit-Rules.html for some details).

Use "script typescript2" to create a file that shows you doing the following:
a. "make clean –f makefile2" to clear out old *.o files
b. "make –f makefile2" to do a full rebuild of calcit.
c. modify getop.c to add in a comment line with your name.

d. "make –f makefile2" to see just getop.c recompiled, plus the loadable calcit.
e. "touch calc.h" to simulate an edit to calc.h.
f. "make –f makefile2" to see two compiles based on dependencies on calc.h, plus the loadable calcit.
g. Use the “diff makefile makefile2“ command to show how the two makefiles are different.
h. Use exit to save the typescript2 file.

For this problem, turn in a hardcopy of a “typescript2” file as usual.