CS246 lab Notes #7, preprocessor directives

- Prototypes reviewed
  - Recall a function prototype is the function declaration statement without the actual code for the function.
  - Prototypes tell the compiler that a function exists but will be defined later.
  - Why do we have prototypes?
    - gcc is a "one-pass compiler"
    - It doesn't go back and figure out what to do with a function they didn't know would exist later
    - Thus functions would have to be written in a dependency based order
    - Prototypes are used to tell the compiler that a function WILL exist, but it hasn't been defined yet.
- Preprocessor
  - What is the preprocessor?
    - There are advantages to certain things being done before compiling begins. For example, giving the compiler prototypes for functions like "printf" so that assembly can be written properly.
    - The preprocessor does many convenience jobs, rewrites the source file without doing any actual compiling.
  - #include
    - Adds library functions to do specific tasks
  - #define
    - Syntax: #define <replace-ee> <replace-or>
    - Example:
      - #define BUFFERSIZE 100
      - Then you might have code like:
      - int buffer[BUFFERSIZE]
      - for(i = 0; i < BUFFERSIZE; i++)
      - This makes code more readable, and easier to modify because you change the BUFFERSIZE in one place.
    - Example:
      - #define TEST_EXISTS
      - then you could have code
      - #ifdef TEST_EXISTS
      - …
      - #endif
    - #define vs. const
      - #define is a preprocessor operation, changes text, but is not part of the actual compiler
      - Affects all subsequent code, regardless of scope
      - #define can change more than just variables, can represent functions, etc. but this is not recommended.

- #if, #ifdef, #ifndef, #else, #elif, #endif
  - The C preprocessor contains a simplified conditional system.
  - Works very similarly to "if-then" statements, but more efficient in the compiled code because the code removed never even gets compiled.
  - Usage
    - #if X == 2 (if X is #defined to be 2)
    - #ifdef X (if X is #defined at all)
    - #ifndef X (if X is not #defined)
    - #else – as expected
    - #elif X == 2– "else if X is #defined to be 2"
    - #endif – ends the if block
  - Often used for cross-platform code
    - #ifdef WINDOWS, #ifdef UNIX
    - #ifdef LITTLE_ENDIAN
  - Dependency checking
    - #ifdef is very useful in header files…
- gcc can define preprocessor variables
  - The –Dmacro[=def] flag of g++
    - g++ –DTEST lab07.cpp –o lab07
      - #define TEST (without putting it in code)
    - g++ –DTEST=1 lab07.cpp –o lab07
      - #define TEST 1 (without putting it in code)
    - g++ –DTEST=0 lab07.cpp –o lab07
      - #define TEST 0 (without putting it in code)

- Exercise: write a program that has 4 different function definitions that print different lines to the console based on conditional compilation, and one main function that calls the defined function.
- Create a makefile that compiles each version separately using the –D option and runs each of the programs.