Purpose of this handout:

1) C Syntax / Semantics
2) Style requirements (comments, variable naming, indentation) { placement)

Basic C Program 1:

```
/* helloworld.c
  * purpose: Show basic C program structure and printing
  */
#include<stdio.h>
#include <stdlib.h>

int main()
{
      printf("Hello World\n");
 }
```

****

To compile the code (go from human-readable code to an executable for the computer):

The executable will be named a.out: `$ clang helloworld.c`

To run: `$ ./a.out`

To make an executable with the name myprogram `$ clang helloworld –o myprogram`

To run: `$ ./myprogram`

****

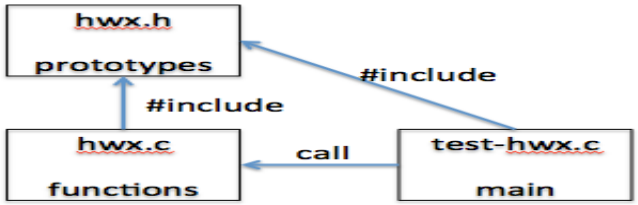
Basic C Program 2:

```
/*  singlemaintemp.c
  * purpose: Show variable declaration, use, reading from user, printing.
  */
#include<stdio.h>
#include <stdlib.h>

int main()
{
      double fahr, celsius;      // declaration of variable named fahr to be type double

      // print out user information
      printf("This program converts from Celsius to Fahrenheit\n");

      // get user input
      printf("Enter a temperature in Celsius > ");
      scanf( "%lf", &celsius);

      // calculate the temperature
      fahr =  (celsius*9.0)/5.0+32;

      // output the result
      printf("%lfC == %lfF\n", celsius, fahr);

      // return success
      return (0);

 }
```

Basic Program 3:

```
/*  functiontemp.c
  * purpose: Show how to implement, use functions.
  */
#include<stdio.h>
#include <stdlib.h>

/* convert_celsius_to_fahrenheit
 * purpose: Converts a temperature given in Celsius into Fahrenheit.
 *  input parameters:
 *   float – the temperature in Celsius
 *  return value:
 *   float – the temperature in Fahrenheit
 */
float convert_celsius_to_fahrenheit(float cel)
{
      return (cel*9.0)/5.0+32;            // must explicitly return the calculated number
}

int main()
{
      double fahr, celsius;

      // print out user information
      printf("This program converts from Celsius to Fahrenheit\n");

      // get user input
      printf("Enter a temperature in Celsius > ");
      scanf( "%lf", &celsius);

      // calculate the temperature
      fahr =  convert_celsius_to_fahrenheit(celsius);

      // output the result
      printf("%lfC == %lfF\n", celsius, fahr);

      // return success
      return (0);

 }
```

Basic Program 4: Showing how to place code in multiple files
All programs in this class will be placed in a minimum of 3 files.
test-hwx.c – contains main code that starts execution. Contains function *calls*.
hwx.c – contains function *implementations* that are called by main
hwx.h – contains function *prototypes* of all functions implemented in hwx.c



********** hwx.h **********

```
#ifndef HWX_H                   // always at top of hwx.h. Compiler reads file only 1x.
#define HWX_H                   // this makes prior statement false next time read.
/* convert_celsius_to_fahrenheit        // this is the required function header
 * purpose: Converts a temperature given in Celsius into Fahrenheit.
 *  input parameters:
 *   float – the temperature in Celsius
 *  return value:
 *   float – the temperature in Fahrenheit
 */
float convert_celsius_to_fahrenheit(float);                     // function prototype
#endif                          // matches #ifndef in 1st line
```

********** hwx.c **********

```
#include <stdio.h> // these are the first two lines of all .c files
#include <stdlib.h>
#include "hwx.h"
/* convert_celsius_to_fahrenheit        // this function header goes again in
implementation
 * purpose: Converts a temperature given in Celsius into Fahrenheit.
 *  input parameters:
 *   float – the temperature in Celsius
 *  return value:
 *   float – the temperature in Fahrenheit
 */
float convert_celsius_to_fahrenheit(float celsius)          // function implementation
{
      return (celsius*9.0)/5.0+32;
}
```

****

To compile the code (go from human-readable code to an executable for the computer):

The executable will be named a.out:	`$ clang hwx.c test-hwx.c`
To run:	`$ ./a.out`
To make an executable with the name blah	`$ clang hwx.c test-hwx.c –o blah`
To run:	`$ ./blah`

********** test-hwx.c **********

```
/* test-hwx.c
 * purpose: illustrate how all homework will be structured (3 files)
 * same code as basic program 3, just different file structure and testing
 */

#include <stdio.h> // these are the first two lines of all .c files
#include <stdlib.h>
#include "hwx.h"    // this allows us to call functions from hwx.c

/* This is where we will put testing helper functions (later) */
void test_convert(double celsius, double expected)
{
      double fahr;
      // calculate the temperature
      fahr =  convert_celsius_to_fahrenheit(celsius);         // function call
      // output the result
      printf("Test celsius %lfF, expected %lfC, calculated %lfC.\n",
                  celsius, expected, fahr);
}

/* this is where the program starts */
int main()
{
      double fahr, celsius;

      // print out user information
      printf("This program converts from Celsius to Fahrenheit\n");

      // call a series of tests
      test_convert(     ,     );              // fill these in with well thought-out tests
      test_convert(     ,     );
      test_convert(     ,     );
      test_convert(     ,     );

      // return success
      return (0);

 }
```

****

To automate the compilation process, unix has the program "make" that uses a file named makefile
Contents of makefile for this program:

```
myexe: test-hwx.c hwx.c hwx.h
      clang test-hwx.c hwx.c –o myexe
```

Notes:

1) Must be a tab before "clang", not spaces.
2) myexe: is the name of the target. You may have multiple targets in one makefile.
3) Typing make compiles the 1st target in file.
4) If you have multiple targets, use "make myexe" to compile
5) test-hwx.c hwx.c hwx.h on first line after ':' means that if any of those files have changed, recompile. List all files there for which it should compile in case of change.