Writing Functions

Panel 1

Prior to Le12

Panel 2

Writing Functions
- passing parameters
- return types
- prototypes
Writing a function together for `addTwoNumbers`
Writing a function by yourself for `integerExp`

Panel 3

Passing parameters to a function

```c
// Function: void sample(void)
// Input Variables:
// Output return:
// Overview: You change this template of a function if needed
void sampleFunction(){
    // Some function that does a specific task
}
```

Panel 4

Passing parameters to a function

```c
void sampleFunction(int a){
    a = 55;
}

void sampleFunction(char a){
    a = 55;
}
```
Passing parameters to a function

```
void sampleFunction(int a){
    a = 55;
}

void sampleFunction(char a){
    a = 55;
}
```

Usually you use the variable to control other things

```
void sampleFunction(char b){
    int y;
    y = 5*b;

    // Do stuff with y or b within function
}
```

Return types from a function

```
int sampleFunction(){
    int y;
    y = 22;

    return y;
}
```

```
char sampleFunction(){
    char y;
    y = 22;

    return y;
}
```

Can only return 1 variable!

```
long float sampleFunction(){
    char y;
    float f;
    y = 22;
    f = 1.25;

    return y, f;
}
```

Example: `rand()` was a function that returned a value

```
rand
Function:      Generate a pseudo-random integer.
Include:       stdlib.h
Prototype:     int rand( void );
Remarks:       Calls to this function return pseudo-random integer values in the range [0,32767]. To use this function effectively, you must seed the random number generator using the srand() function. This function will always return the same sequence of integers when identical seed values are used.
Return Value:  A pseudo-random integer value.
File Name:     rand.asm
```
Panel 9

Just cause a function has a return value, you're not required to store it

```
printf
Function: Formatted string output to stdout.
Include: stdio.h
 Prototype: int printf (const char *fmt, ...);
Remarks: The printf function formats output, passing the characters to
 stdout via the putc function. The format string is processed as
 described for the fprintf function.
Return Value: printf returns EOF if an error occurs, otherwise returns the number
 of characters output.
Filename: printf.c
Code Example: #include <stdio.h>
void main (void)
{
   /* will output via stdout (_H_USART by default) */
   printf ("Hello, World\n");
}
```

Panel 10

Function prototypes

```
/*** Local Function Prototypes ******************************************/
void sampleFunction(void);

/* Additional Helper Functions */

/*================================================================*/
void sampleFunction(void)
{
   // Some function that does a specific task
}
```
Your turn `addTwoNumbers`

Your function should be called `addTwoNumbers`
Receives two integers
Returns an integer
The return value is simply the sum of the two inputs

Make sure to add the function prototype as well!!!
Panel 17

Simplified syntax

```c
void main (void)
{
    int Result;
    Result = addTwoNumbers(2,3);
}
```

```c
int addTwoNumbers(int a, int b){
    return a+b;
}
```

Panel 18

Your turn integerExponent

Your function should be called integerExponent
- Receives two inputs
  - int base = The number for the base in base^x
  - char exp = The exponent in base^exp
- Returns a long
  - long = The result of base^exp

The return value is simply the expected result if integers were used in exponentiation. Note that ^ for integers becomes exclusive OR. So we made our own function!

Hint:
- Make sure to add the function prototype
- Make sure it works for the 0th power a^0 = 1
- Try using a well constructed for loop!

Panel 19

This question will be #1 on the quiz
Use the function to load an array with 14 elements
The elements should be exponents[i] = 5^i

```c
void main(void)
{
    int a, b, Result, m;
    char i;
    int expPow(32);
    int base = 1;
    b = 5;
    Result = addTwoNumbers(b, b);
    for(i = 0; i < 14;)
    {
        array[i] = integersExp(i);
    }
    while (i < 10)
    {
        // This array loops forever
    }
}
```

Panel 20

Multiple .c and .h files

- You can have as many .c and .h files as you like within a project.
- However you can only have 1 function named main.
- All the .c and .h files should be in the same folder with the project.

Some unfamiliar syntax, but easy to use a template

In this class you're not required to use multiple files but I wanted to make you aware that you could and big project almost ALWAYS use different files for different tasks.
Sample image from a BIGGER project (lots of files)

The files get added, the functions and prototypes move but Main looks the same

Supporting header file

Supporting source file
Add the #include file using "" instead of < > since it’s in the folder not the path

Modular Programming using a Supporting file:

Download the blank template
Supporting.c and Supporting.h
files off the website Calendar under today’s date

Cut your integerExp from file with main in it
and put it into Supporting.c with a prototype in Supporting.h

Add the #include "Supporting.h"

Add the extern prototype in Supporting.h (with comments)

If you want make a few other functions in Supporting

Sending an int and getting back the ASCII character for each digit:
- char charHundredsDigit(int);
- char charTensDigit(int);
- char charOnesDigit(int);