Lecture 4 Functions

* An executable program is loaded into main memory at execution
* Memory space is divided into four regions for the execution of the program
  + Text region (stack, system stack): hold functions
  + Data region: hold global, static variable data
  + Stack region: hold runtime function, local variable data
  + Heap region: hold runtime dynamic data

**Call stack of function calls**

1. When a function is called the argument and local variables of the functions are pushed onto the call stack. Namely the instances of local variables and argument variables of the function are allocation in the stack region
2. After that the argument and local vars are instantiated and have absolute addresses. Parameter values are copied to the argument variable locations. Both argument and local vars are accessed by the statements of the function
3. After the function call is done, the argument and local vars of the function are popped off from the call stack. The memory space in the call stack will be reused by other function calls.
4. If function f1 calls function f2, the argument and local variables of f2 are pushed to call stack to top of f1’s argument and local variables. This results in the growing of the call stack.

Passing by reference

* If you put a \* in front of a function parameter you’re really passing the address in memory
* Similarly, if you put & in front of a variable it refers to the address of that variable
  + &n is called “the reference of n” it represents the memory location of n at runtime

**What are &, \***

* & - reference operator to get the address of a variable
* \* - dereference operator to get the value stored at the memory location of the address

**Swapping values**

* It is necessary to swap using memory locations

Void swap(int \*first, int \*second){

Int temp = \*first;

\*first = \*second;

\*second = temp;

}

**Defining constants (macros)**

* Use the #define to define a constant

Ex. #define PI 3.1415926

#define PROMPT “Please enter a character \n”

#define getmax(a,b) a>b?a:b //this is not a function, it’s a macro

* #ifdef and #endif can have a line in between them that will run if a function is defined