Shazaib Dawood
CS 211
09/16/2024

Outline

- What are the differences between memory on the stack vs. the heap? How is each region of memory utilized?
    - Stack
        - Used for static memory allocation
        - Fixed size
        - Fast access
        - Includes local variables, and function arguments
    - Heap
        - Used for dynamic memory allocation
        - Variables stay in memory until they are deallocated
        - Size changes depending the amount of memory being allocated/deallocated
        - Can be accessed globally as long as a pointer to the memory exists
        - Susceptible to memory leaks
- What is malloc() used for? What is the proper syntax for calling malloc()? Specifically, describe every component of a sample call, such as int* ptr = (int*)malloc(sizeof(int)*100);
    - malloc() allocates memory from the heap
    - int* = declares a pointer to an integer
    - (int*) typecasts the memory address to an integer pointer
    - sizeof(int) * 100 = the size of 1 int times 100 so it gives the size of 100 ints
    - malloc() = allocates the specified amount of memory from the heap and returns a pointer to the memory address
- What is realloc() used for? What is the proper syntax for calling realloc()? How does realloc() differ from malloc()?
    - realloc() adjusts the size of the memory that has already been allocated without changing the data that is stored

    - ptr = (int*)realloc(ptr, sizeof(int) * 200);
        - Ptr = the pointer to memory that has already been dynamically allocated from the heap
        - (int*) casts the memory address to an int pointer
        - realloc(ptr, sizeof(int) * 200) = resizes the amount of dynamically allocated memory to the size of 200 ints
    - Difference between malloc() and realloc()
        - malloc() allocates a new block of memory
        - realloc() resizes a block of memory that was already allocated without changing the data that was stored inside
- What is free() used for? When should free() be called? What are the potential issues if free() is not used properly?
    - free() deallocates the memory to avoid memory leaks
    - free() should be called when the data stored in the memory block is no longer needed
    - Potential issues
        - Memory leaks
            - Memory isn't being deallocated and all free memory ends up being used causing the system to crash
        - Deallocated pointers
            - Using free() on a pointer that has already been deallocated can cause unpredictable behavior