Lab 03: C Programing Part two:

Lab Theory:

This Lab will be focusing on C memory, specifically pointers, the use of the commands malloc and free and general memory management.

Memory:
Memory to C is a row of bytes and variables are reserved in the specific memory, and variables are assigned to a particular memory address and their position is remembered.

How C allocates Memory:
Using the standard library functions, the two main dynamic memory functions are `malloc();` and `free();`
`malloc()` allocates heap memory, taking a number of bytes you want as a number, allocate them on a heap and return a pointer to those bytes. If more memory is used than what you have, it will cause a memory leak and it will return Null, or crash the system etc.
`free()` is vital for memory de-allocation, it frees up space from `malloc()` which is important, to avoid `malloc()` to return Null.
These commands are used with `#include <stdlib.h>`

How `malloc()` is usually used:
```
int //(This is a pointer ->)// *arr = malloc([Array size here] * sizeof(int));
//int stores 4 bytes per 1 array
```
This declares a pointer arr that will point to an integer, as it has * indicating a its pointer.

```
One Array:
With 4 bytes (32 bits), the range is:
Minimum value:  -2,147,483,648
Maximum value:  2,147,483,647
```
This is for a signed integer.

If its unassigned, then the value can only go into min: 0 and max 4,294967295

```
All log files and code are attached here:
https://github.com/WillGusackov/SecurePrograming.git
```

Exercise one:

Building static version of a calculator
In order to code a static version of a calculator, the first steps is to include all necessary libraries, in this code `<stdio.h> <string.h> <stdlib.h>` are included.

Memory storage is vital for this program, a pointer was implemented, to create an array with custom sizing.
The following code was used: int* arr(int size){
return malloc(size * sizeof(int));
Once the storage place is created, the main void should include the following ints:
i, highNumber = 0;, then a pointer array = arr(5) for the 5 spaces for the user input values.
scanf is called in an if statment, if(scanf("%d ... , &array[0]) This is for the user input, int separated by commas.

For extra input, char extraChar is added, then scanning the char "%c" , extraChar and if its avalible it will give a warning, clear the extra characters from the buffer and print the highest number from the first five values, using a for loop to check each intager.

```
for(i=1; i <5; i++){
if(array[i] > highestNumber){
highestNumber = array[i];
}
```

Once this is completed, using the command free(array); and return 0; will free up the memory to avoid memory leak or other issues.
The code is attached as lab03.o lab03.c lab03.log