CT248: Introduction to Modeling

Assignment 2: Sub-functions and Function Handles

The aim of this assignment is explore how sub-functions can be made available from a function file, via functional handles. The aim is to implement a simple stack in a file (mystack.m), where the main function passes back the three stack manipulation functions. A stack is a "last in first out" queue. The function file does not maintain any stack state, therefore the stack itself is passed in to functions, and then, when it's modified, the updated stack is returned.

The file **mystack.m** contains the following functions:

Function Name	**Function Type**	**Inputs**	**Outputs**	**Description**
mystack	Main function for the file.	None	push pop peek	Main function that returns 3 handles to the stack functions
mystack_push	Sub-function	stack value	stack	Pushes value onto the stack (location 1)
mystack_pop	Sub-function	stack	stack	Pops value off the stack (i.e. value in array location 1). Should return an empty stack if there is only one element
mystack_peek	Sub-function	stack	value	Returns the top value from the stack (array location 1). If the stack is empty, it should not throw an error.

The code below shows a test case. The first call sets up the three stack functions so that they can be called. The variable stack is an array variable that holds the data. It is passed into functions, and also its modified value returned (from two of the functions).

Implement a script that contains the following test code.

```
[push, pop, peek] = mystack();

stack = []
stack = push(stack,100)
>> stack = 100

stack = push(stack,200)
>> stack = 200   100

peek(stack)
>> ans = 200

stack = pop(stack)
>> stack = 100

peek(stack)
>> ans = 100
```