## Sheet #1 (Stack)

In this sheet you will explore the Stack data structure at the implementation level. Please follow the instructions described below carefully. Note that every function in the implementation level assumes some pre-conditions. It is the responsibility of the user, at the application level, to check for these pre-conditions. For example, the function pop assumes that the stack is not empty. Similar pre-conditions apply to other functions; make sure that you do not overlook these pre-conditions throughout this exercise.

1. Write the implementation level of the array-based stack we discussed in the lecture in a separate file and name it stack.c. Compose the header file stack.h that consists of the prototypes of the functions along with the definition of the stack element type stack_entry. Define stack_entry to be a character. Will you need to include stack.h in stack.c? Why? Compile stack.c and make sure that the compiler generated the object code file stack.obj.

2. Write a test program, test.c that calls the stack functions. The program should display and carry out the appropriate actions based on the following menu:

(a) Read an element then Push it.
(b) Pop an element then displays it.
(c) Exit.

At this point, will you be able to build test.c? Why? Build your project; this is the step in which the "Linker" links the functions from stack.obj to test.obj to create one executable file test.exe. Run your program.

3. In stack.c, write the function stack_size that returns the size of the stack. Write the pre- and post- conditions for this function. Add this option to the menu of the main program defined in number 2.

4. Assume that you purchased the stack library stack.obj, along with its header stack.h, from the author, i.e., you do not know any of the details of stack.c. Write a function to print on the screen the contents of a stack without changing the stack. Why shouldn't this function be written at the implementation level?

5. Write a function that returns the first element entered to a stack. (Implementation level)

6. Write a function that returns a copy from the last element in a stack. (Implementation level)

7. Write a function to destroy a stack. (Implementation level)

الممسوحة ضوئيا بـ CamScanner

8. Write a function to copy a stack to another. (Implementation level)

9. Write a function to return the size of a stack (Implementation level)

10. Write a function that returns the first element entered to a stack. (User level)

11. Write a function that returns a copy from the last element in a stack. (User level)

12. Write a function to destroy a stack. (User level)

13. Write a function to copy a stack to another. (User level)

14. Write a function to return the size of a stack (User level)

15. Write a function to print on the screen the contents of a stack without changing the stack (user level). Why should this function not be written at the implementation level?

16. Use a stack structure to check the balance and ordering between various parentheses.

الممسوحة ضوئيا بـ CamScanner