Task 01

1 Basic Stack [5 marks]

Implement a stack in C that holds integer values. It needs to support three main operations: **top**, **pop**, and **push**, which inspect the top element, take the top element off the stack and put a new element on the stack, respectively.

To make the task more challenging, we have a few constraints on the implementation:

1. There shall be no need to explicitly call any initialisation routine for the stack.
2. You may assume that the size of the stack is fixed. The default stack size is given through a macro `DEFAULT_STACK_SIZE`.
3. None of the three functions is allowed to ever access non-allocated memory.
4. There needs to be a static global variable named `stackTop` which at all time points to the top element. In case of the stack being empty it needs to point to a value 0.
5. You are **not** allowed to modify / extend the given implementation of **top**.
6. Try to implement **push** and **pop** as efficiently as you can.
7. The functions **push** and **pop** both return the number of elements they successfully popped / pushed. **pop** returns the top element via its argument.

Use the boiler-plate provided through vision as a starting point. It contains a module `stack` with a fixed interface in `stack.h` whose functionality needs to be implemented in `stack.c`.

A simple test program is provided in `stacktest.c`. A `Makefile` enables convenient compilation of both, the module `stack` and the test program.

Make sure that you carefully implement, test, and debug your implementation.

2 Resizing the Stack [3 marks]

Implement three more functions as already declared in `stack.[ch]`: **getStackSize**, **setStackSize**, and **deleteStack**. They are supposed to modify, inspect, or delete the current stack. Make sure that **setStackSize** never leads to a loss of elements on the stack and that **deleteStack** resets the stack into the default state.

3 Infinity Stacks [2 marks]

Create a new module `infstack` which implements a stack without any given maximum size. In contrast to the module `stack`, this module imposes no fixed size on the stack. The functions **top**, **pop**, and **push** should behave as before albeit without a limit on the growth of the stack. Try to come up with an as (runtime) efficient program as possible.