Process

The final submission will include a stack server (component I) implementing multi process shared memory and locking routines

1. Do you still have the stack from ex4? the stack should support:

a. PUSH <text>\n

b. POP\n

c. TOP\n

2. The expected behaviour

a. PUSH will push <text> into the stack.

b. POP will pop text from the stack

c. TOP will display (output) the stack top.

d. Your output should begin with OUTPUT: prefix. These lines are the only lines that will be checked by us. (Your only command that generate output is the TOP command)

e. For the sake of clarity and simplicity. You may assume that all commands are only in uppercase. Text includes characters and can be capped at 1024 bytes. You may add additional commands for operation or debugging. (Such as command for quitting, displaying stack contents or clearing the stack.) Additionally you may assume that all commands and text end with a \n (ASCII 10) and that this character will not be part of the text.

f. You may assume all input is legal and that TOP or POP are only given when the stack is in non-empty state but encouraged to give error message if you find illegal command. Use ERROR: <cause> prefix for errors.

g. We will ignore any other prefix for output. However, you are encouraged to use DEBUG: for debug output.

3. Merge the beej multi process server with the stack implemented in 2.

a. You now serve multiple client that transmit the stack command (instead of standard input)

4. You should implement locking now as you may receive multiple commands to the stack on different connections.
use fcntl locking.

5. Practice: implement new methods to manipulate memory (malloc, free) in multi processes environment. Do not use libc primitives. use mmap MAP_SHARED

Process

shared memory to implement malloc.

6. Final submission: Merge the server implemented in 4 with the components developed in section 5. **You should submit both the server and client.**

7. Bonus (2 pts): implement Unix domain sockets (DGRAM) interface in addition to the TCP sockets.

8. Bonus (2 pts):code quality bonus - for good implementation (OOAD) 2 points bonus will be awarded (no appeals)

9. Additional instructions:

this exercise (like all other exercises) will be checked using moss for cheating