4.2

1. main process before appl1:
Base of the stack : Address - 0x0EFD8FFC, Value - 0xFFFFFFA9
Top of the stack : Address - 0x0EFE8FFC, Value - 0x00000055

2. after appl1() is crated before fun1() is called:
Base of the stack : Address - 0x0FDEFFFC, Value - 0xFFFFFFA9
Top of the stack : Address - 0x0FDF07FC, Value - 0x00000000

3. after appl() calls fun1() and before fun1() returns:
Base of the stack : Address - 0x0EFC8FFC, Value - 0xFFFFFFA9
Top of the stack : Address - 0x0EFC97CC, Value - 0xFFFFFFB6

4. after appl1() calls fun1() and after fun1() has returned:
Base of the stack : Address - 0x0EFC8FFC, Value - 0xFFFFFFA9
Top of the stack : Address - 0x0EFC97CC, Value - 0xFFFFFFB6

I noticed that the address changed between 1 and 2 but the value at the top of the stack didn't change when appl1 was called. The base of the stack value was consistent value wise, but the address changed when appl1 was called. When fun1 was called, it seems that none of the addresses or values changed because it was ran as a process instead of a normal function.

4.2

My approach for the stack smashing was to run a recursive method that called itself to the ceiling value of int, overflowing the stack with return addresses.