Han Wang (wang2786, 0028451697)

1.

```
1.main process before appl1():
Base of the stack : Address - 0x0EFD8FBC, Value - 0x00103B75
Top of the stack : Address - 0x0EFD8FC8, Value - 0x0EFD8FE8

2. after appl1() is created before fun1() is called:
Base of the stack : Address - 0x0EFC8FC8, Value - 0x0EFC8FE8
Top of the stack : Address - 0x0EFC8FD4, Value – 0x0EFC8FF4

3. after appl1() calls fun1() and before fun1() returns:
Base of the stack : Address - 0x0EFC8F98, Value - 0x00101D76
Top of the stack : Address - 0x0EFC8FA4, Value - 0x0EFC8FD4

4. after appl1() calls fun1() and after fun1() has returned:
Base of the stack : Address - 0x0EFC8FC8, Value - 0x00000000
Top of the stack : Address - 0x0EFC8FD4, Value - 0x0EFC8FF4
```

After color code all the same value, we can clearly see the context switch between appl1()created to call fun1() to fun1() return. The stack got pushed with return address so that once the function is done, it can find its way back to the original context. The important note is that after calling fun1(), the top of the stack address is pushed into the top of new context(fun()'s new stack) as a VALUE, so that it can remember it and not lost by changing pointer address. Then safely return back to appl1()'s top of the stack.

5.

Attack strategy: Overflow the stack by recursively calling a custom function with local variable, which will eventually get access to victim's stack and causing OS to panic.

Impact: The OS will trap the process from keep functioning and dump the register memory to stdout.