Assignment 1

Vedala Sai Ashok - EE18BTECH11044

Download all latex-tikz codes from

https://github.com/hellblazer1/EE4013_A1/tree/main/codes

1 Problem

Consider the C functions foo and bar given below

```
int foo(int val){
    int x = 0;
    while(val > 0){
        x = x + foo(val−−);
    }
    return val;
}

int bar(int val){
    int x = 0;
    while(val > 0){
        x = x + bar(val−1);
    }
    return val;
}
```

Invocations of foo(3) and bar(3) will result in:

1) Return of 6 and 6 respectively.
2) Infinite loop and abnormal termination respectively.
3) Abnormal termination and infinite loop respectively.
4) Both terminating abnormally.

2 Solution

Answer : 3) Abnormal termination and infinite loop respectively.

3 function foo

3.1 Implementation

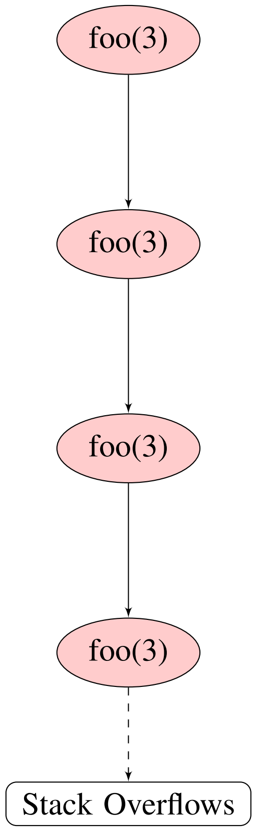
Below is the C implementation of the algorithm.

https://github.com/hellblazer1/EE4013_A1/tree/main/codes/foo_func.c

3.2 Explanation

In the function foo, in each of its recursive calls inside the while the same functions is being called with the value 3, because val is passed with post decrement operator so the value 3 is passed and val is decremented after the recursive call finishes execution. Every time the function is called a new local variable, with scope limited to the function being called, with name val is being assigned the value 3. So the program will be terminated abruptly with segmentation fault as the stack is overflowed.

3.3 Logic Flow



4 function bar

4.1 Implementation

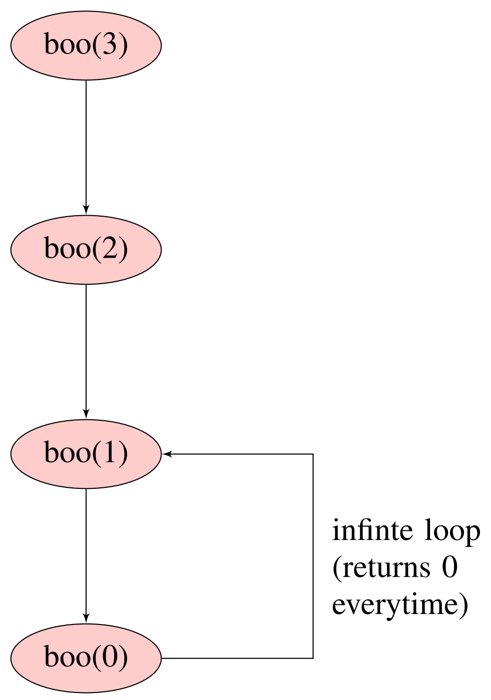
Below is the C implementation of the algorithm.

```
https://github.com/hellblazer1/EE4013_A1/tree/
    main/codes/bar_func.c
```

4.2 *Explanation*

In the function bar, the program will be stuck in an infinite while loop after the third recursive call to the function bar (val = 1 in this recursive call). The fourth recursive call made to the function bar with val = 0 will always return 0 and as the val is not getting updated in the third recursive call (val is always 1 in third recursive call),the while loop will run until the program is terminated. Here stack is not getting overflowed with recursive call as at any instant in time there are a maximum of four recursive calls to the function bar in the stack.

4.3 *Logic Flow*



5 Function for sum till n natural numbers

5.1 *Implementation*

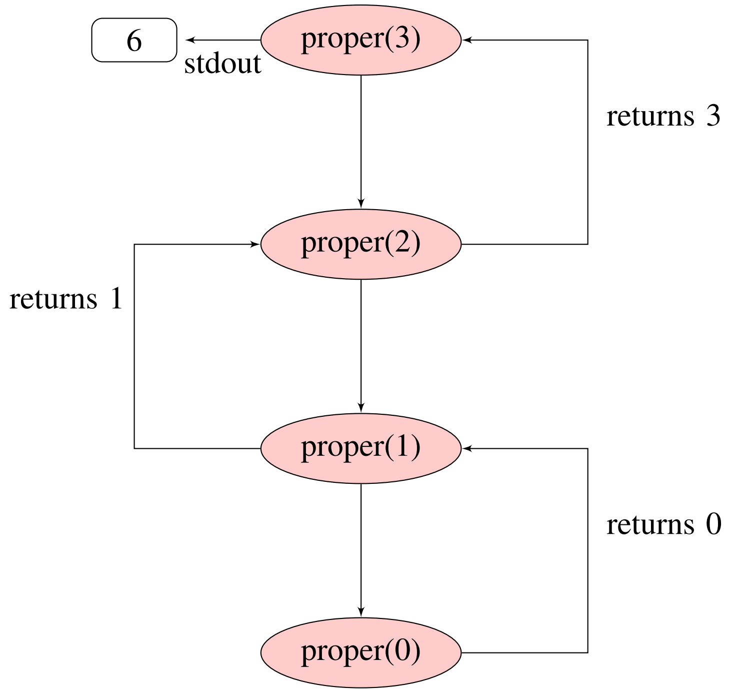
Below is the C implementation of the algorithm.

```
https://github.com/hellblazer1/EE4013_A1/tree/
    main/codes/proper_func.c
```

5.2 *Explanation*

- Both the functions foo and bar have attempted to make recursive calls after decrementing the value of variable val by 1. The output of these recursive calls and value of variable val is getting accumulated in local variable x after the execution of each recursive call.So It would be a fair assumption to make that both these functions attempt to calculate the sum of first n natural numbers (3 in this case).
- In the function proper, the program will output the sum of first n natural numbers by making n recursive calls and accumulating the value in local variable x after and returning it after each recursive call execution.

5.3 *Logic Flow*


6
stdout
proper(3)
returns 3
proper(2)
returns 1
proper(1)
returns 0
proper(0)