Worksheet 10 - Run-time Stack

Exercise 1.
Consider the following C code.

```
int foo(int data, int count){
    int i, ret=0;

    for(i=count;i>0;i--){
        printf("%d", data++);
        ret += data;
    }
    return ret;
}

int main(){
    int ret;
    ret = foo(1,3);
    return 0;
}
```

1) Determine what's on the Run-time stack (NA if the value cannot be determined).
   a) Just before passing the control to the callee function (i.e. after caller setup)

Address	Value (if can be determined)	Contents	R5/R6
x4000			
x4001			
x4002			
x4003			
x4004			
x4005			
x4006			
x4007			
x4008	NA	ret (main's local variable)	<-R5
...		main's bookkeeping info	

   b) After callee setup

Address	Value (if can be determined)	Contents	R5/R6
x4000			
x4001			
x4002			
x4003			
x4004			
x4005			
x4006			
x4007			
x4008	NA	ret (main's local variable)	
...		main's bookkeeping info	

c) Just before callee tear-down

Address	Value (if can be determined)	Contents	R5/R6
x4000			
x4001			
x4002			
x4003			
x4004			
x4005			
x4006			
x4007			
x4008	NA	ret (main's local variable)	
...		main's bookkeeping info	

d) After callee tear-down (i.e. just after returning to the caller function)

Address	Value (if can be determined)	Contents	R5/R6
x4000			
x4001			
x4002			
x4003			
x4004			
x4005			
x4006			
x4007			
x4008	NA	ret (main's local variable)	
...		main's bookkeeping info	

e) After caller tear-down

Address	Value (if can be determined)	Contents	R5/R6
x4000			
x4001			
x4002			
x4003			
x4004			
x4005			
x4006			
x4007			
x4008	9	ret (main’s local variable)	
...		main’s bookkeeping info	

2) Convert the foo function to LC3 code. The LC3 code for the main function is given. You can assume a single digit output for the printf function.

```
.ORIG   x3000
    LEA R6, STACK_END
    ADD R5, R6, #0      ;set main frame pointer
    ;;Caller setup
    AND R0, R0, #0
    ADD R0, R0, #3
    ADD R6, R6, #-1
    STR R0, R6, #0      ;push 'count' for foo
    AND R0, R0, #0
    ADD R0, R0, #1
    ADD R6, R6, #-1
    STR R0, R6, #0      ;push 'data' for foo

    ;;Call the function
    JSR FOO
    ;;Caller tear down
    LDR R0, R6, #0
    STR R0, R5, #0
    HALT
FOO
    ;;Callee setup (no more than 8 lines)
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________
    ____________________

    ;;Function logic
    _________________   ;R1=count
    _________________   ;R2=data
    AND R3, R3, #0      ;R3=ret(zero)
LOOP
    LD  R0, ASCII_ZERO
    ADD R0, R2, R0
    OUT                 ;printf
    ADD R2, R2, #1      ;data++
```

```
    ADD R3, R3, R2      ;ret+=data
    ADD R1, R1, #-1
    BR__    LOOP
    ;;Callee tear down
    ;Push return value
    _____________________
    ;pop local var
    _____________________
    ;recover cfp
    _____________________
    _____________________
    ;recover return addr
    _____________________
    _____________________
    RET
.BLKW   #20
STACK_END   .FILL   #0
ASCII_ZERO  .FILL   x30
.END
```