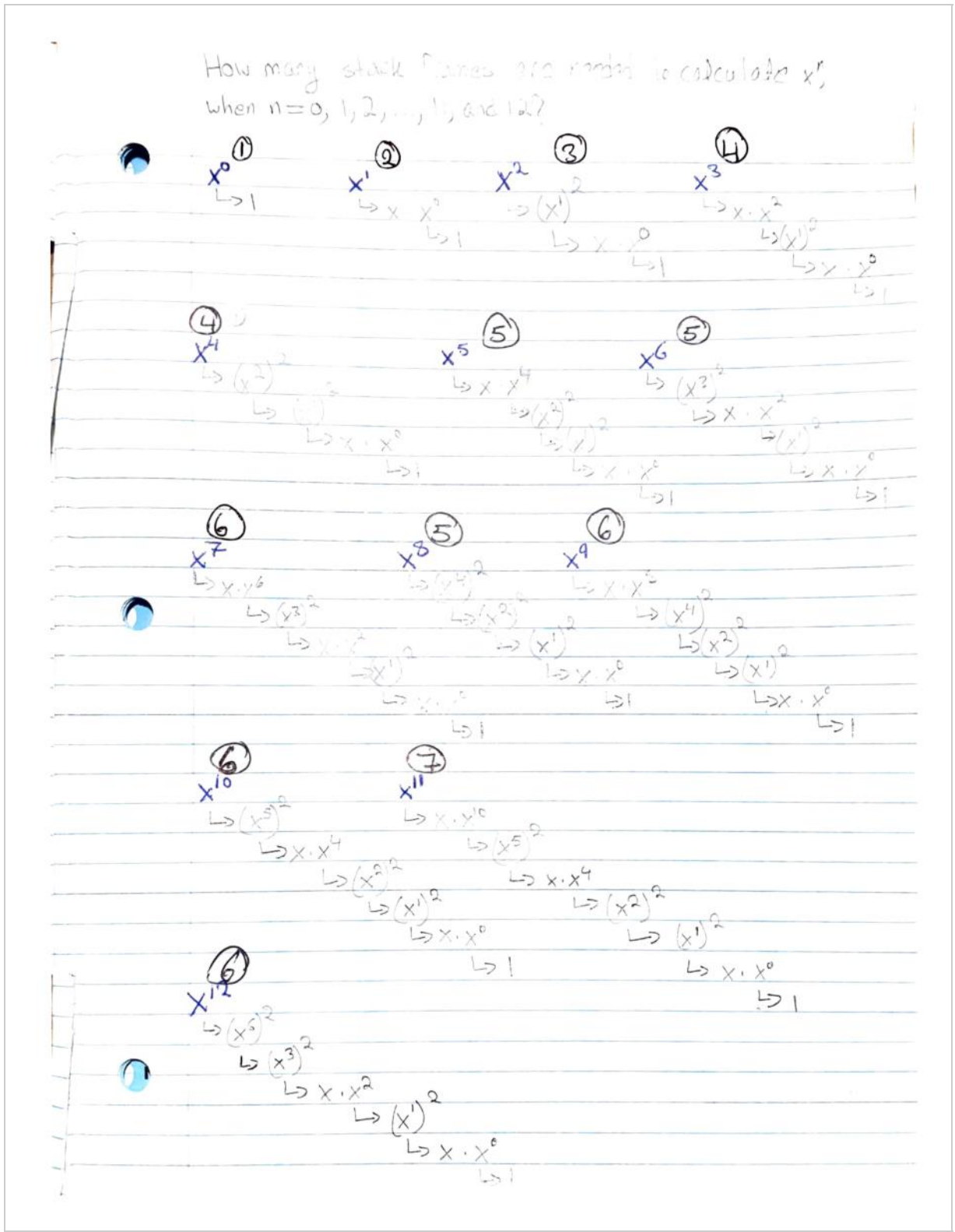


```
        AREA power, CODE, READONLY
x       EQU 2                       ;declare the initial x value
n       EQU 3                       ;declare the initial n value
        ENTRY
main    ADR sp,stack                ;define the stack
        MOV R0, #x                  ; prepare parameter x
        MOV r1, #n                  ; prepare parameter n
```

```
        STMFD sp!, {r0-r1}              ; push the parameters x and n on the stack
        SUB sp,sp,#4                    ; save a position in stack for returned value

        BL P                                ; call power subroutine
        LDR r1,[sp],#4                  ; pop result from stack and save it in R1
        ADD sp,sp,#8                    ; make sp point to base of stack

        ADR r2,result                   ; get address of result
        STR r1,[r2]                         ; store result in its register
loop    B loop                          ; infinite loop


P       STMFD sp!,{r0-r3,fp,lr}         ; push needed registers, fp, and lr for new workspace
        MOV fp,sp                           ; set the fp for this call
        LDR r1,[fp,#0x1C]               ; get n from the stack
        LDR r0,[fp,#0x20]               ; get x from the stack

                                        ; "First Condition"
        CMP r1,#0                           ; comparing if (n == 0)
        MOVEQ r1,#1                         ; prepare returning value
        STREQ r1,[fp,#0x18]             ; store the returned value in the stack, beyond workspace
        BEQ return                          ; then return to calling function

                                        ; "Else Condition"
        TST r1,#1                           ; else if (!n & 1) [n is even]
        BEQ EVE                                 ; skip going through the second condition to the block that handles even exponents

                                        ; "Second Condition"
                                        ;else [n is odd]
        SUB r2,r1,#1                        ;{ prepare the new parameter value `n-1`
        STR r2,[sp,#-4]!                ; push the parameter `n-1` on the stack in place of n
        STR r0,[sp,#-4]!                ; push the parameter `x` on the stack as is
        SUB sp,sp,#4                    ; reserve a place in the stack for the return value
        BL P                                ; call the power subroutine
        LDR r2,[sp],#4                  ; load the result in r2 and pop it from the stack
        ADD sp,sp,#8                    ; remove also the parameters from the stack
        MUL r3,r0,r2                    ; prepare the value to be returned
        STR r3,[fp,#0x18]               ; store the returned value in the stack
        B return

                                        ;}
EVE         LSR r2,r1,#1                    ;{ prepare the new parameter value `n >> 1`
        STR r2,[sp,#-4]!                ; push the parameter `n >> 1` on the stack in place of n
        STR r0,[sp,#-4]!                ; push the parameter `x` on the stack
        SUB sp,sp,#4                    ; reserve a place in the stack for the return value
        BL P                                ; call the power subroutine
        LDR r2,[sp],#4                  ; load the result in r2 and pop it from the stack
        ADD sp,sp,#8                    ; remove also the parameters from the stack
        MUL r3,r2,r2                    ; prepare the value to be returned
        STR r3,[fp,#0x18]               ; store the returned value in the stack
                                        ;}
return  MOV sp,fp                               ; collapse all working spaces for this function call
        LDMFD sp!,{r0,r1,r2,r3,fp,pc}   ; load all registers and return to the caller

        AREA power, DATA, READWRITE
result  DCD 0x00                                ; result
        SPACE 0xE0                          ; space for stack (max will be 7), I'll make it 8 just in case
stack   DCD 0x00                                ; base of Stack (configuration is Full Descend)
        END
```

Sketch the structure of the stack frame that you utilized in your program.		
Stack position	content	
28		
24		
20	lr	<-- fp

1C	fp	
18	r3	
14	r2	
10	r1	
C	r0	
8	reserved for return value	
4	n	
0	x	