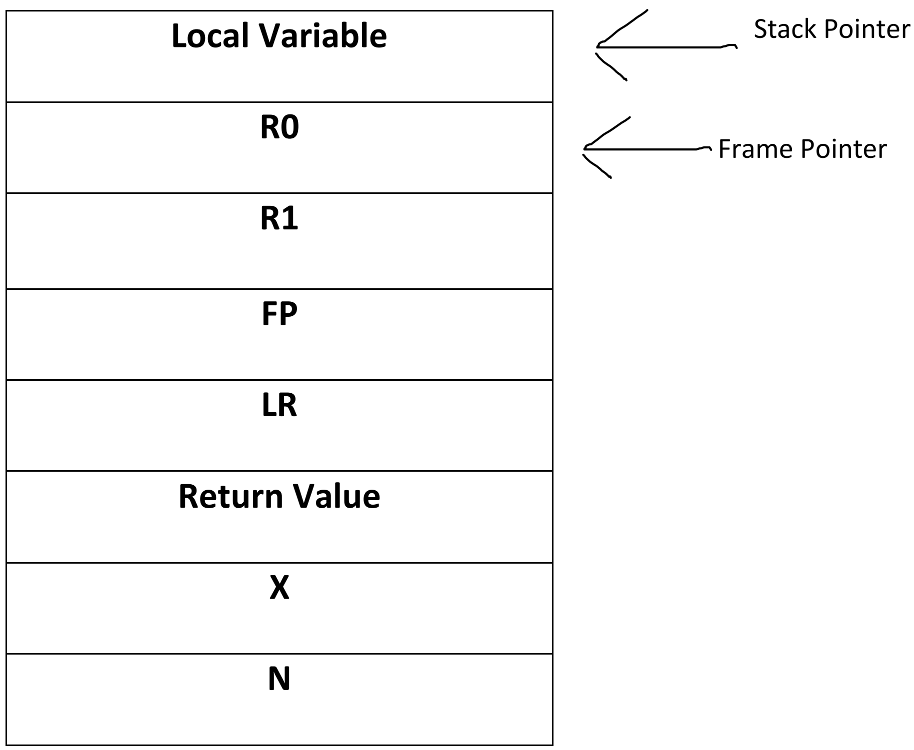

How many stack frames are needed to calculate X^N , when n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12?

Stack frames required to calculate X^N:

When N = 0, 1 frame is required.

When N = 1, 2 frame is required.

When N = 2, 3 frame is required.

When N = 3, 4 frame is required.

When N = 4, 4 frame is required.

When N = 5, 5 frame is required.

When N = 6, 5 frame is required.

When N = 7, 6 frame is required.

When N = 8, 5 frame is required.

When N = 9, 6 frame is required.

When N = 10, 6 frame is required.

When N = 11, 7 frame is required.

When N = 12, 6 frame is required.

```
        AREA power, CODE, READWRITE
        ENTRY

        ADR     r13, STAK                       ;initialize stack
        MOV     r1, #X                          ;move input value X for recursive function X^N
        MOV     r2, #N                          ;move input value N for recursive function X^N
        STMFD   r13!, {r1-r2}                   ;push X and N values of r0-r1 onto the stack
        BL      CALC                            ;branch to CALC function

        LDR     r3, [r13], #12          ;after CALC function is finished executing, load return value from top of stack into empty
        ADR     r4, RESULT              ;obtain memory address of variable RESULT
        STR     r3, [r4]                ;store the final value into RESULT

DONE    B       DONE                    ;endless loop to end program

CALC    SUB     r13, r13, #4            ;allocate 4 bytes of space for final return value RESULT
        STMFD   r13!,{r1-r2, FP, LR}    ;push FP and LR into the stack
        MOV     FP, r13                 ;set frame pointer (FP) to the position of stack pointer which is the top of the stack
        SUB     r13, r13, #4            ;allocate 4 bytes of space for final return value RESULT
        LDR     r2, [FP, #24]           ;load input n value into the stack
        CMP     r2, #0                  ;compare loaded n value with 0
        MOVEQ   r2, #1                  ;if the loaded n value is 0, then move 1 into r2 to be returned later
        BEQ     RET                     ;if equal to zero, branch to RET and end calculation

        LDR     r1, [fp, #20]           ;obtain next value of X that was stored
        AND     r2, #1                  ;since n is NOT 0, AND the value of n with #1 with TST
        CMP     r2, #1                  ;compare the value to 1 to see if it is odd or even
        BEQ     NUMEVEN                 ;if not equal, the value is even, branch to NUMEVEN to handle
        B       NUMODD                  ;else, the value is odd, branch to NUMODD to handle

NUMEVENMOV      r2, r2, LSR #1          ;divide r2/2 by LSRing one position
        STMFD   r13!, {r1-r2}           ;push X and N values of r0-r1 onto the stack
        SUB     r13, r13, #4            ;allocate 4 bytes of space in stack frame for the return value
        BL      CALC
        LDR     r1, [FP, #-16]          ;obtain return value from previous call of CALC
        STR     r1, [FP, #-4]           ;store the returned value within the previously allocated stack frame space
        MUL     r2, r1, r1              ;multiple the returned value accordingly (r1*r1) and store into r2

        B               RET

NUMODDSUB       r2, r2, #1              ;if the new value is not equal, reduce its value by 1
        STMFD   r13!, {r1-r2}           ;push X and N values of r0-r1 onto the stack
        SUB     r13, r13, #4            ;allocate next 4 bytes of space in stack frame for the return value
        BL      CALC
        LDR     r2, [FP, #-16]          ;obtain return value from position reserved above in CALC
        MUL     r2, r1, r2              ;multiply x with return value stored in r1

RET     STR     r2, [FP, #16]           ;store final value into designated stack frame for return value
        MOV     r13, FP                 ;adjust frame pointer
        LDMFD   r13!, {r1-r2, FP, PC}   ;restore registers modified by the stack (r0, r1, FP, PC) & return to branch calling

                AREA    power, DATA, READWRITE
STAK    DCD     0x00
RESULT  DCD             0x00
X               EQU             4
N               EQU             2

                END
```