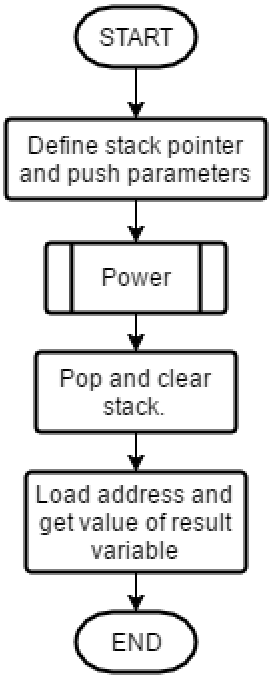
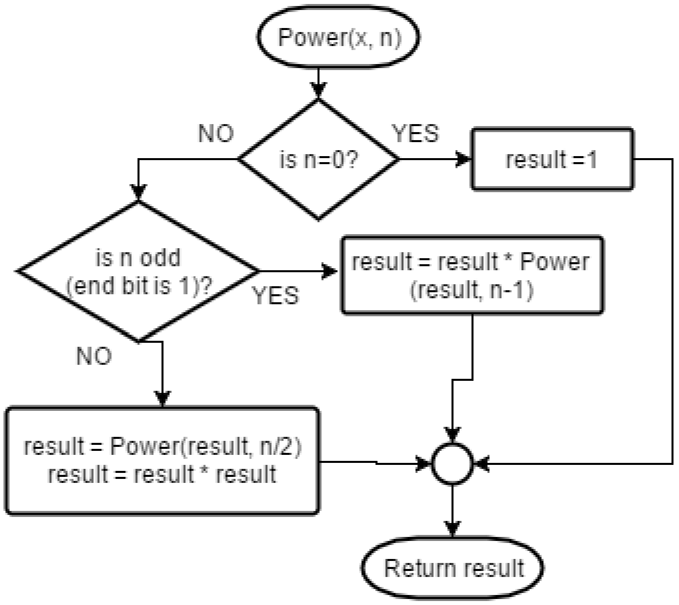
Question 1


START
Define stack pointer and push parameters
Power
Pop and clear stack.
Load address and get value of result variable
END


Power(x, n)
is n=0?
YES
result =1
NO
is n odd (end bit is 1)?
YES
result = result * Power (result, n-1)
NO
result = Power(result, n/2)
result = result * result
Return result

```
;Vivian Lam

;program to recursively computre factorial. uses the stack


;----------------------------------------------------------------------
-----------

     AREA power, CODE, READONLY

n    EQU 12

x    EQU 2

     ENTRY


Main LDR   sp, =stackk      ;define the stack by setting a pointer to
it

     MOV   r0, #n        ;prepare the parameter

      MOV   r2, #x


     STR   r0,[sp,#-4]!  ;push the parameter (n)on the stack

      STR   r2,[sp,#-4]!    ;pushing x


      SUB   sp,sp,#4      ;reserve a place in the stack for the return
value


      BL    Power          ;call the power subroutine


     LDR   r0,[sp],#4    ;load the result in r0 and pop it from the
stack

     ADD   sp,sp,#8      ;also remove the parameter from the stack


     ADR   r1,result     ;get the address of the result variable
     STR   r0,[r1]       ;store the final result in the result variable


Loop B     Loop          ;infinite loop so no error


;-----------------------------------------------------------------------------------

     AREA power, CODE, READONLY

Power STMFD sp!,{r0,r1,r2,fp,lr} ;push general registers, as well as fp and lr

     MOV   fp,sp         ;set the fp for this call


      SUB   sp,sp,#4      ;create space for the y local variable


      LDR r0,[fp,#28];load n: 5 registers*4 + 4(from result return address) + 4 (from x) + (we do not consider the 4 from n cus we already pointing to top) = 28 forward

      LDR r2,[fp,#24];load x


     CMP   r0,#0         ;if (n = 0)

     MOVEQ r0,#1         ;{ prepare the value to be returned

     STREQ r0,[fp,#20] ;  store the returned value in the stack

     BEQ   ret           ;  branch to the return section

                         ;}


      ;is n odd?

      TST r0, #1 ;AND to check last bit: if it's a 1 then the register contains an odd number
      ;BNE Odd   ;n is odd (zero flag not set) return x * power(x, n - 1);

      BEQ Even;n is even (ztero flag set), y= power(x, n >> 1); return y * y


;          LDR   r0,[fp,#0x18] ;get the parameter from the stack (get n)

;     LDR   r2,[] ;load x


Odd ;n is odd (zero flag not set) return x * power(x, n - 1);

      ;prepare the value to be returned (x, n-1)

      SUB r0,r0,#1 ;preparing n-1 and storing result into r0

      ;push parameters to the stack (push x and n-1)

      STR r0,[sp,#-4]! ;pushin n-1

      STR r2,[sp,#-4]!;push x to the stack

      SUB sp,sp,#4;reserve space in stack for return value


      BL Power ;call power subroutine

      LDR r0,[sp],#4;load result (the empty space we created above) and pop


      ADD   sp,sp,#8      ;also remove the parameters (there's two:n and x, so use 8) from the stack


      MUL r2,r0,r2;calculating: return x * fact(x,n-1);

      STR r2,[fp,#20];store the returned value in the stack

      B ret;branch to return (so that we don't calculate the Even case)
```

```
Even ;n is even (ztero flag set), y= power(x, n >> 1); return y * y

      ;preparing new parameter: divide n by 2 by shifting right by 1

      LSR r0,r0, #1

      ;push parameters to the stack

      STR r0,[sp,#-4]! ;pushin n/2

      STR r2,[sp,#-4]!;push x to the stack

      SUB sp,sp,#4;reserve space in stack for return value


      BL Power ;call power subroutine

      LDR r0,[sp],#4;load result (the empty space we created above) and pop


      ADD   sp,sp,#8      ;also remove the parameters (there's two:n and x, so use 8) from the stack


      STR r0,[fp,#-4] ;set y equal to the result (store r0 into the location of y, which is fp-4<offset cus stack type is FD)

      MUL r1,r0,r0 ;calculate y*y

      STR r1,[fp,#20];store the returned value in the stack




ret  MOV   sp,fp         ;collapse all working spaces for this function call

     LDMFD sp!,{r0,r1,r2,fp,pc} ;load all registers and return to the caller

;---------------------------------------------------------------------------------

     AREA power, DATA, READWRITE

result DCD   0x00        ;the final result
```

```
       SPACE 0x200       ;declare the space for stack
        ALIGN
stackk  DCD   0x00        ;initial stack position (FD model)
;---------------------------------------------------------------------
-----------
       END
```

CS2208 Ass5 vlam54

%%

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?

n=0: 1 frame needed
n=1: 2 frames needed
n=2: 3 frames needed
n=3: 4 frames needed
n=4: 4 frames needed
n=5: 5 frames needed
n=6: 5 frames needed
n=7: 6 frames needed
n=8: 5 frames needed
n=9: 6 frames needed
n=10: 6 frames needed
n=11: 7 frames needed
n=12: 6 frames needed