1. Loop Invariant (5 points)

Use the loop invariant (I) to show that the code below correctly computes the product of all elements in an array A of n integers for any n ≥ 1. First use induction to show that (I) is indeed a loop invariant, and then draw conclusions for the termination of the while loop.

Algorithm 1 computeProduct(int[ ] A, int n)

```
p = a[0]
i = 0
while i < n − 1 do
    //(I) p = a[0] · a[1] · · · a[i] (Loop Invariant)
    i + +
    p = p · a[i]
end while
return p
```

Base case is n=1

$n=1 \rightarrow p=a[0]$, as loop does not run...

$at\ n=1,\ p\ is\ invariant\ .$

$1 \leq i < n$

$i=n-1 \rightarrow p=a[0]*a[1]*...\ a[n-1]$

when i = n-1, loop exits...$\rightarrow p=a[0]*a[1]*....\ a[i]$

p is invariant at loop exit

2. Recursive Definitions (12 points)

(1) (4 points) Write down the first 6 elements of the following sequence (where $n \in Z+$), then give a recursive definition for a_n. Do not forget the base case. (You do not need to prove it is correct).

(a) $a_n = 3n - 10$

$$n \in \mathbb{Z}^+ \rightarrow n \geq 1$$

$$a_1 = 3(1) - 10 = -7$$
$$a_2 = 3(2) - 10 = -4$$
$$a_3 = 3(3) - 10 = -1$$
$$a_4 = 3(4) - 10 = 2$$
$$a_5 = 3(5) - 10 = 5$$
$$a_6 = 3(6) - 10 = 8$$

Base Case for n: $n=1$

$$f(n) = 3(n) - 10$$
$$f(n+1) = 3(n+1) - 10$$
$$f(1) = -7$$
$$f(2) = -4 = f(1) + 3$$
$$f(n) = f(n-1) + 3 = ((3(n-2) - 10) + 3) + 3$$
$$a_2 = f(2) = f(1) + 3 = ((3(0) - 10) + 3) + 3 = ((-10) + 3) + 3 = (-7) + 3 = -4$$
$$a_4 = f(4) = f(3) + 3 = ((3(2) - 10) + 3) + 3 = (-1) + 3 = 2$$
$$a_5 = f(5) = f(4) + 3 = ((3(3) - 10) + 3) + 3 = (2) + 3 = 5$$
$$a_5 = a_4 + 3$$
$$\ldots$$
$$a_1 = -7$$
$$a_n = a_{n-1} + 3 \quad , \quad n > 1$$

(b) $a_n = (1 + (-1)^n)^n$

$$a_1 = (1 + (-1)^1)^1 = 0^1 = 0$$
$$a_2 = (1 + (-1)^2)^2 = 2^2 = 4$$
$$a_3 = (1 + (-1)^3)^3 = 0^3 = 0$$
$$a_4 = (1 + (-1)^4)^4 = 2^4 = 16$$
$$a_5 = (1 + (-1)^5)^5 = 0^5 = 0$$
$$a_6 = (1 + (-1)^6)^6 = 2^6 = 64$$

n must be even, or result is 0

$$n \in \mathbb{Z}^+$$
$$a_1 = 0 \text{ and } a_2 = 4$$
$$a_n = 4\, a_{n-2}$$

(c) $a_n = 2^{n!}$

$$n = 1$$
$$a_1 = 2^{1!} = 2^1$$
$$a_2 = 2^{2!} = 2^2$$
$$a_3 = 2^{3!} = 2^6$$
$$a_4 = 2^{4!} = 2^{24}$$
$$a_5 = 2^{5!} = 2^{120}$$
$$a_6 = 2^{6!} = 2^{720}$$
$$a_n = a_{n-1}{}^{n}$$

(2) (4 points) Give a recursive algorithm to compute the length of an array of positive integers A. You can assume that the final element of the array is -1. Also give the initial call to your recursive algorithm

```
int [ ] A
findLength(int[ ] a)
        if (a[0] == -1)
                return 1
        else
                return 1 + findLength(a+1)

findLength(A)
```

(3) (4 points) Define A as follows:

$$A = \begin{pmatrix} 1 1 \\ 1\ 0 \end{pmatrix}$$

Prove the following using weak induction:

$$A^n = \begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix}$$

$$n=1$$
$$f_0=0$$
$$f_1=1$$
$$f_2=f_1+f_0$$
$$A^1=\begin{pmatrix} f_2 & f_1 \\ f_1 & f_0 \end{pmatrix}=\begin{pmatrix} 11 \\ 10 \end{pmatrix}$$

$$\text{Assume A}^k=\begin{pmatrix} f_{k+1} & f_k \\ f_k & f_{k-1} \end{pmatrix} \forall\, 1\le k<n$$

$$A^{n-1}=\begin{pmatrix} f_n & f_{n-1} \\ f_{n-1} & f_{n-2} \end{pmatrix}$$

$$A^n=A*A^{n-1}=\begin{pmatrix} f_n & f_{n-1} \\ f_{n-1} & f_{n-2} \end{pmatrix}*\begin{pmatrix} 11 \\ 10 \end{pmatrix}=\begin{pmatrix} f_n+f_{n-1} & f_n+0 \\ f_{n-1}+f_{n-2} & f_{n-1}+0 \end{pmatrix}=\begin{pmatrix} f_{n+1} & f_n \\ f_n & f_{n-1} \end{pmatrix}$$

3. Structural Induction (5 points)

Let S be the subset of the set of ordered pairs of integers defined recursively

by:

Base case: (0, 0) $\in$ S

Recursive step: If (a, b) $\in$ S, then (a + 1, b + 3) $\in$ S and (a + 3, b + 1) $\in$ S.

(1) (1 point) List the elements of S produced by the first four applications

of the recursive definition (this should produce 14 new elements).

$$S_0=\{(0,0)\}$$
$$S_1=S_0\cup(1,3),(3,1)$$
$$S_2=S_1\cup(2,6),(4,4),(6,2)$$
$$S_3=S_2\cup(3,9),(5,7),(7,5),(9,3)$$
$$S_4=S_3\cup(4,12),(6,10),(8,8),(10,6),(12,4)$$

(2) (4 points) Use structural induction to show for all (a, b) $\in$ S that (a+b) =

4k for some k $\in$ Z.

Reminder: In other words (a + b) is divisible by 4.

S sub 0 = (0,0), therefore base case is a=0 and b=0

Each recursive step a+=1 and b+=3, or a+=3 and b+=1, either way,

(a+b)+=4 in total. Therefore, since base is (a+b)=0, and each recursive

call (a+b)+=4, the total sum of any point at a certain recursive depth of 'n'

is (a+b)=4n. Also, since (a+b)=4n for some integer 'n', 4|(a+b)

4. Applications of Recurrence Relations (4 points)

(1) (0.5 points) Count the number of length 2 bit strings that start and end in 1.

A : 1

(2) (0.5 points) Count the number of length 3 bit strings that start and end in 1.

A : 2

(3) (1.5 points) Find a recurrence relation for the number length n bit strings that start and end in 1 (where n $\geq$ 2).

Length n bit string : 2^n

2 predetermined bits : 2^{n-2}

$$a_2 = 1$$

$$a_n = 2^{n-2} \rightarrow a_{n-1} = 2^{n-3} \rightarrow a_n = 2 * 2^{n-3} = 2 * a_{n-1}$$

(4) (1.5 points) Use this recurrence relation to find the number of permutations of a set with n elements using the expansion method.

Reminder: We used expansion method in class to generate a guess for the number of moves in the Towers of Hanoi problem