CS280 – Data Structures – Quiz

`DigiPen login:` _____

1. Given an array of size n write a recursive function that reverses it in place. For example if original array is $1, 2, 3, 4, 5$, after the function executes it should be $5, 4, 3, 2, 1$.

2. Write a recursive function that takes 2 arrays of the same size (say `a1` and `a2` and index range is $0, \ldots, n-1$) and returns the **first** index i so that `a1[i] is equal a2[i]`. If such index does not exists the function returns n (one past the last element).

3. Given two natural numbers x and y find product using recursion. You are ONLY allowed to use addition.

4. Rewrite your solution to the previous problem as a tail-recursion.

5. (also a lab) (Definition copied from Wikipedia) The Tower of Hanoi (also called the Tower of Brahma or Lucas' Tower and sometimes pluralized) is a mathematical game or puzzle. It consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape.

   The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

   (a) Only one disk can be moved at a time.

   (b) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack.

   (c) No disk may be placed on top of a smaller disk.

   Algorithm: denote pegs A, B, C. Initially all disks (enumerated $1...n$, where 1 is biggest – at the bottom) are on peg A (source), destination is peg C (destination), peg B is currently empty (auxiliary). The meaning of source and auxiliary will alternate throughout the algorithm:

   (a) Move top $n-1$ disks from source to auxiliary (recursive call)

   (b) Move disk n from source to destination.

   (c) Move top $n-1$ disks from auxiliary to destination (recursive call)

   What is the run-time of your algorithm?

6. Solve previous problem but with 4 pegs – provide algorithm and run-time complexity?

   What is the run-time of your algorithm?

7. Write recursive function `power` that uses the following idea:

$$b^p = \begin{cases} b^{\frac{p}{2}} * b^{\frac{p}{2}} & \text{if p is even,} \\ b * b^{\frac{p-1}{2}} * b^{\frac{p-1}{2}} & \text{if p is odd.} \end{cases}$$

   Extra: Rewrite your solution to the previous problem as a tail-recursion.

8. (also a lab) Write recursive function that prints all possible subsets of a given set. Assume set is $\{1, \ldots, n\}$. Output for $n = 3$ shoud be (in any order):

$$\{\}\{1\}\{2\}\{3\}\{12\}\{13\}\{23\}\{123\}$$

.

Idea: given a set $\{1, \ldots, n\}$, each subsets is either

- subset of $\{1, \ldots, n-1\}$ or
- subset $\{1, \ldots, n-1\}$ with n added.

Hint: 2 recursive calls – one to generate subsets without n, one with n.

9. (harder) Recursively reverses the given linked list in one pass of the list. You are not allowed to use dynamically allocated memory. I.e. no `new/malloc` in your code. Also do not use STL. You may first work on iterative version to get familiar with the problem.

```
void RecursiveReverse(struct node** headRef) {
    // your code...
```