**Homework 10**

Note: The first three questions of this homework must be done

作業前三題一定要做

1.Please use recursive function to compute the following:

(a). S =  +  +  + … + 

(Think about when n becomes to 2 what the value of this term)

(b). π= 4 \*  \*  \*  \*  \*  \*  \* …

(You may consider ( and  ), ( and ), ( and ), …as one pair and let n=0 as 4 )

(c). From (b) and using the constant M\_PI in the math.h to find the approximated value of π till the error between M\_PI and your approximated valueπreduces to 0.005.

1. Write a program that inputs two numbers: x and y. (data are all integers) in the main program and passes these two numbers

( pass by value) to the **recursive function: power** that returns

the .

If y ≧0,

1 if y=0

power(x, y)= x if y=1

x\*power(x,y-1) if y>1

If y<0

power(x, y)= 

Print the result in the main program.

3. A robot can take steps of 1meter, 2 meters and 3 meters.

Write a recursive function to evaluate the number of ways the

robot can walk n meters.

4.

A robot can take steps of 1meter, 2 meters. Write a

function that lists all of the ways that the robot can walk n meters.

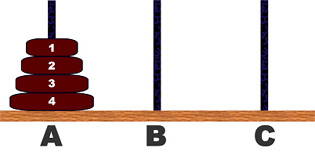
5.

Suppose that we have a 2 × n rectangular board divided into 2n squares. Write a function that computes the number of ways to cover this board exactly by 1 × 2 dominoes.

6.

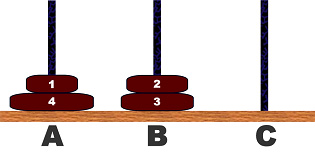
A Different Task

|  |
| --- |
| **A Different Task** |



The (Three peg) Tower of Hanoi problem is a popular one in computer science. Briefly the problem is to transfer all the disks from peg-**A** to peg-**C** using peg-**B** as intermediate one in such a way that at no stage a larger disk is above a smaller disk. Normally, we want the minimum number of moves required for this task. The problem is used as an ideal example for learning recursion. It is so well studied that one can find the sequence of moves for smaller number of disks such as **3** or **4**. A trivial computer program can find the case of large number of disks also.

Here we have made your task little bit difficult by making the problem more flexible. Here the disks can be in any peg initially.



If more than one disk is in a certain peg, then they will be in a valid arrangement (larger disk will not be on smaller ones). We will give you two such arrangements of disks. You will have to find out the minimum number of moves, which will transform the first arrangement into the second one. Of course you always have to maintain the constraint that smaller disks must be upon the larger ones.

**Input**

The input file contains at most **100** test cases. Each test case starts with a positive integer ***N*** ( 1***N***60), which means the number of disks. You will be given the arrangements in next two lines. Each arrangement will be represented by ***N*** integers, which are **1**, **2** or **3**. If the ***i***-th ( 1***i******N***) integer is **1**, you should consider that ***i***-th disk is on Peg-**A**. Input is terminated by ***N*** = 0. This case should not be processed.

**Output**

Output of each test case should consist of a line starting with `Case #: ' where # is the test case number. It should be followed by the minimum number of moves as specified in the problem statement.

**Sample Input**

3

1 1 1

2 2 2

3

1 2 3

3 2 1

4

1 1 1 1

1 1 1 1

0

**Sample Output**

Case 1: 7

Case 2: 3

Case 3: 0