5.20 ***(Displaying a Rectangle of any Character)*** Write a function **rectangle(length, breadth, fillCharacter)** that forms a rectangle out of whatever character is contained in character parameter fillCharacter. Thus, if length is 2, breadth is 4 and fillCharacter is M, i.e. **rectangle(2, 4, M)**, then this function should print:

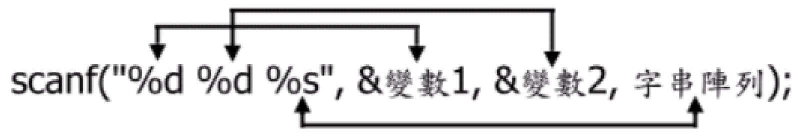
```
MMMM
MMMM
```

ANS:

Useful hints:

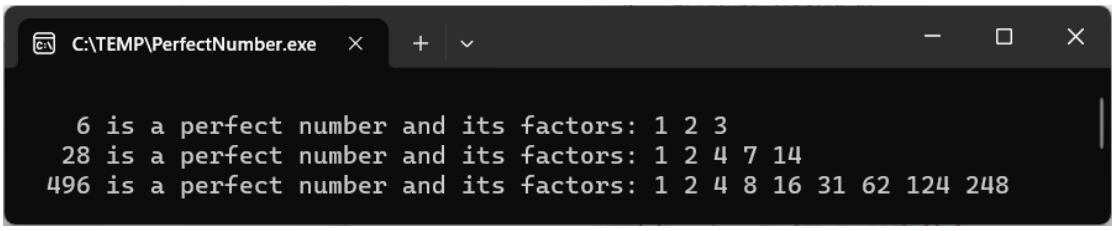
- scanf 輸入函式
    - 格式: scanf("△變數對應 輸入參數列", &變數1, &變數2, ...);
    - 說明：
        1. 變數對應參數列順序要對應變數串列，且對應參數型態需與變數型態一致。
        2. 參數串列前，可空一格，使 scanf 讀取時忽略空白的字元。
        3. 變數前需加上 & 符號，scanf 定義使用 & 符號來取變數的位址。
        4. 雙引號內不能加入字串。

      例如：



```
scanf("%d %d %s", &變數1, &變數2, 字串陣列);
```

5.26 ***(Perfect Numbers)*** An integer number is said to be a *perfect number* if its factors, including 1 (but not the number itself), sum to the number. For example, 6 is a perfect number because 6=1+2+3. Write a function **isPerfect** that determines whether parameter **number** is a perfect number or not. Use this function in a program that determines and prints all the perfect numbers between 1 and 1000. Print the factors of each perfect number to confirm that the number is indeed perfect number. Challenge the power of your computer by testing numbers much larger than 1000.



```
  6 is a perfect number and its factors: 1 2 3
 28 is a perfect number and its factors: 1 2 4 7 14
496 is a perfect number and its factors: 1 2 4 8 16 31 62 124 248
```

ANS:

Perfect numbers aren't common at all. There are only 6, 28, 496 and 8,128, below a million. Only 50 total perfect numbers are known, even with a dedicated worldwide effort to computationally discover more.

5.36 ***(Towers of Hanoi)*** Every budding computer scientist must grapple with certain classic problems, and the Towers of Hanoi (see Fig. 5.23) is one of the most famous of these. Legend has it that in a temple in the Far East, priests are attempting to move a stack of disks from one peg to another. The initial stack had 64 disks threaded onto one peg and arranged from bottom to top by decreasing size. The priests are attempting to move the stack from this peg to a second peg under the constraints that exactly one disk is moved at a time, and at no time may a larger disk be placed above a smaller disk. A third peg is available for temporarily holding the disks. Supposedly the world will end when the priests complete their task, so there is little incentive for us to facilitate their efforts. Let us assume that the priests are attempting to move the disks from peg 1 to peg 3. We wish to develop an algorithm that will print the precise sequence of disk-to-disk peg transfers.

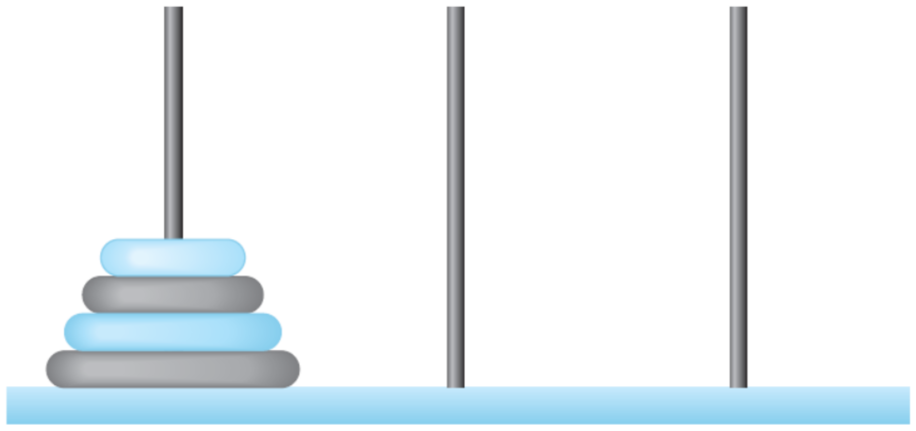
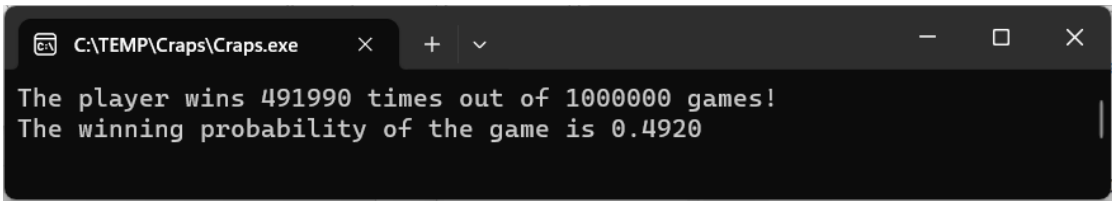


Fig. 5.23 | Towers of Hanoi for the case with four disks.

Extra Problem ***(Craps)*** Write a function that simulates a popular dice game called "craps" (see the lecture note of Chap.5). Use this function in a program that plays the game 10,000 times or more to estimate your winning probability? Is it 49.29%?



```
The player wins 491990 times out of 1000000 games!
The winning probability of the game is 0.4920
```

ANS: