火星 (Mars)

法老 (Pharaoh) 們最先到達外太空是廣為人知的。他們發射了第一艘太空船前往圖特摩斯一號星 (Thutmus I，現在一般被稱為火星)。這個星球的表面可被視為一 $(2n+1) \times (2n+1)$ 的網格，其中每個小方格 (cell) 不是陸地就是水域。位於第 i 列 (row) 及第 j 行 (column) ($0 \le i, j \le 2 \cdot n$) 的小方格以 $s[i][j]$ 表示，若該小方格為陸地，則 $s[i][j] =$ `'1'`；若為水域，則 $s[i][j] =$ `'0'`。

若有一條由陸地小方格構成的路徑 (其中連續兩個小方格共用一邊界) 連接兩個陸地小方格，則稱該二小方格相連。星球上的一個島嶼被定義為一個極大的陸地小方格集合 (maximal set of land cells) 滿足其中任二小方格皆相連。

這艘太空船的任務是計算此星球上的島嶼個數。然而，因為太空船使用的是先祖時期的計算機，所以這個任務並不容易。這個計算機有一個記憶體 h 用來儲存一大小為 $(2n+1) \times (2n+1)$ 的二維陣列型式的資料，陣列的每個元素可儲存長度 100 的二元字串，其中每個字元是 `'0'` (ASCII 48) 或 `'1'` (ASCII 49)。初始時，此記憶體中每個陣列元素的第一個字元儲存了網格中一個小方格的資訊，即 $h[i][j][0] = s[i][j]$ (對所有 $0 \le i, j \le 2 \cdot n$)。記憶體中的其他字元初始皆為 `'0'` (ASCII 48)。

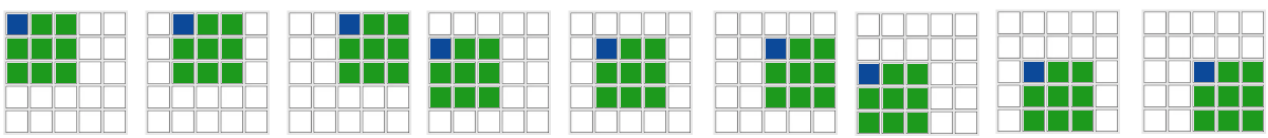
為了處理記憶體中的資料，此計算機僅能存取其中一 3×3 的區域，並且只能覆寫該區域左上角的小方格。更正式地，此計算機能夠存取 $h[i..i+2][j..j+2]$ ($0 \le i, j \le 2 \cdot (n-1)$) 的每個元素，並且覆寫 $h[i][j]$。這個過程被稱為「處理小方格 (i, j)」。

法老們想出了以下的方法來因應計算機有限的處理能力：

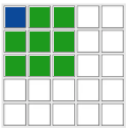
- 此計算機會經由 n 個階段來處理記憶體中的資料。
- 在第 k 階段 ($0 \le k \le n-1$)，令 $m = 2 \cdot (n-k-1)$，此計算機會以 i 的遞增順序來處理小方格 (i, j) (對所有 $0 \le i, j \le m$)，對於每個 i 再以 j 的遞增順序來進行處理。換句話說，此計算機將以下面的順序來處理小方格：$(0,0), (0,1), \cdots, (0,m)$, $(1,0), (1,1), \cdots, (1,m), \cdots, (m,0), (m,1), \cdots, (m,m)$。
- 在最後一階段 ($k = n-1$)，此計算機只會處理小方格 $(0, 0)$。處理完後，寫於 $h[0][0]$ 的字串應等於此星球上的島嶼個數的二進制表示，其中最低有效位元 (least significant bit) 為該字串的第一個字元。

下圖展示了此計算機處理大小 5×5 ($n = 2$) 的記憶體之過程。藍色小方格表示要被覆寫之小方格，有顏色的所有小方格表示被處理的子陣列。

在第 0 階段，此計算機用以下的順序處理子陣列：



在第 1 階段，此計算機僅處理一個子陣列：



你的任務是實作一個方法使得此計算機能以前述的運作方式來計算圖特摩斯一號星上的島嶼數量。

實作細節 (Implementation details)

你應實作下列函式:

```
string process(string[][] a, int i, int j, int k, int n)
```

- a: 一 3×3 陣列，表示要處理的子陣列；特定地說 $a = h[i..i+2][j..j+2]$，其中 a 的每個元素為一長度恰好 100 的字串，字串的每個字元為 `'0'` (ASCII 48) 或 `'1'` (ASCII 49)。
- i, j: 此計算機目前正在處理的小方格之列與行註標。
- k: 目前的階段編號。
- n: 階段總數；星球表面的網格由 $(2n+1) \times (2n+1)$ 個小方格構成。
- 此函式應回傳一長度 100 的二元字串。此回傳值被儲存於記憶體中 $h[i][j]$ 處。
- 此函式最後一次的呼叫應發生於 $k = n-1$。在此呼叫中，此函式應回傳此星球上的島嶼數量的二進制表示，其中最低有效位元位於註標 0 (即該字串的第一個字元)，次低有效位元位於註標 1，以此類推。
- 此函式之回傳值應依函式原型中的參數計算，而不應使用任何靜態 (static) 或全域 (global) 變數。

每組測試包含了 T 個獨立的情境 (即不同的星球表面)。你的實作之運行結果不該因情境的處理順序不同而有所改變。同一個情境其所有 `process` 函式的呼叫不一定連續發生，但保證每個情境的所有 `process` 函式會依題目敘述指定的順序進行呼叫。

此外，對於每組測試，你的程式會同時有數份實例 (instance) 被執行。記憶體用量以及 CPU 時間為所有實例合算。在這些實例間不應以任何非常規的管道傳遞資料，任何蓄意的嘗試將被視為舞弊，並將取消參賽資格。

特別地，在一個 `process` 函式呼叫的過程中，任何以靜態或全域變數儲存的資訊不保證能在下次的函式呼叫中被取得。

條件 (Constraints)

- $1 \le T \le 10$
- $1 \le n \le 20$
- $s[i][j]$ 為 `'0'` (ASCII 48) 或 `'1'` (ASCII 49) (對所有 $0 \le i, j \le 2 \cdot n$)
- $h[i][j]$ 的長度恰為 100 (對所有 $0 \le i, j \le 2 \cdot n$)
- $h[i][j]$ 的每個字元為 `'0'` (ASCII 48) 或 `'1'` (ASCII 49) (對所有 $0 \le i, j \le 2 \cdot n$)

對每個 `process` 函式的呼叫:

- $0 \le k \le n-1$
- $0 \le i, j \le 2 \cdot (n-k-1)$

子任務 (Subtasks)

1. (6 points) $n \leq 2$
2. (8 points) $n \leq 4$
3. (7 points) $n \leq 6$
4. (8 points) $n \leq 8$
5. (7 points) $n \leq 10$
6. (8 points) $n \leq 12$
7. (10 points) $n \leq 14$
8. (24 points) $n \leq 16$
9. (11 points) $n \leq 18$
10. (11 points) $n \leq 20$

範例 (Examples)

Example 1

考慮 $n = 1$ 且 s 如下:

```
'1' '0' '0'
'1' '1' '0'
'0' '0' '1'
```

在此範例中，星球的表面為 3×3 的網格，其中有 2 個島嶼。 僅有 1 個階段會呼叫 `process` 函式。

在第 0 階段，評分程式會呼叫 `process` 函式恰好一次:

```
process([["100","000","000"],["100","100","000"],["000","000","100"]],0,0,0,1)
```

注意 h 中每個元素僅顯示了前 3 個字元。

此呼叫應回傳 `"0100..."` (省略的字元皆為零)，其中 ...0010 為十進制數值 2 的二進制表示。注意，這當中有 96 個零被省略，並以 `...` 代替。

Example 2

考慮 $n = 2$ 且 s 如下:

```
'1' '1' '0' '1' '1'
'1' '1' '0' '0' '0'
'1' '0' '1' '1' '1'
'0' '1' '0' '0' '0'
'0' '1' '1' '1' '1'
```

在此範例中，星球的表面為 5×5 的網格，其中有 4 個島嶼。將有 2 個階段會呼叫 `process` 函式。

在第 0 階段，評分程式會呼叫 `process` 函式 9 次：

```
process([["100","100","000"],["100","100","000"],["100","000","100"]],0,0,0,2)
process([["100","000","100"],["100","000","000"],["000","100","100"]],0,1,0,2)
process([["000","100","100"],["000","000","000"],["100","100","100"]],0,2,0,2)
process([["100","100","000"],["100","000","100"],["000","100","000"]],1,0,0,2)
process([["100","000","000"],["000","100","100"],["100","000","000"]],1,1,0,2)
process([["000","000","000"],["100","100","100"],["000","000","000"]],1,2,0,2)
process([["100","000","100"],["000","100","000"],["000","100","100"]],2,0,0,2)
process([["000","100","100"],["100","000","000"],["100","100","100"]],2,1,0,2)
process([["100","100","100"],["000","000","000"],["100","100","100"]],2,2,0,2)
```

假設上列呼叫之回傳值分別為 `"011"`, `"000"`, `"000"`, `"111"`, `"111"`, `"011"`, `"110"`, `"010"`, `"111"`，其中被省略的字元皆為零。當第 0 階段結束，h 會儲存下面的值：

```
"011", "000", "000", "100", "100"
"111", "111", "011", "000", "000"
"110", "010", "111", "100", "100"
"000", "100", "000", "000", "000"
"000", "100", "100", "100", "100"
```

在第 1 階段，評分程式會呼叫 `process` 函式 1 次：

```
process([["011","000","000"],["111","111","011"],["110","010","111"]],0,0,1,2)
```

最後，此呼叫應回傳 `"0010000..."` (省略的字元皆為零)，其中 ...0000100 為十進制數值 4 的二進制表示。注意，這當中有 93 個零被省略，並以 `...` 代替。

範例評分程式 (Sample grader)

此範例評分程式用以下格式讀取輸入：

- line 1: T
- block i ($0 \le i \le T-1$): 一個代表情境 i 的區塊。
  - line 1: n
  - line $2+j$ ($0 \le j \le 2 \cdot n$): $s[j][0]$ $s[j][1]$ $\ldots$ $s[j][2 \cdot n]$

此範例評分程式用以下格式輸出：

- line $1+i$ ($0 \le i \le T-1$): 函式 `process` 對於第 i 個情境最後的回傳值的十進制表示。