Marathon Match

Problem Statement

Contest: 2011 TCO Marathon Round 1

Normal view

Problem: ImageScanner

Problem Statement

TCO competitors (except those who received a bye to Round 2) must not submit solutions to MM 69. Doing so may result in disqualification.

A black-and-white bitmap image of height **H** and width **W** is created by drawing **nLetter** black letters of various sizes on white foreground. You can't see the whole image, but you can figure out its contents by scanning its rows and analyzing the results. Your task is to restore the image as fully as possible.

Your code should implement a method **restore**. Its parameters give you **H**, **W**, **nLetter** and the number of black pixels in the image **nb**. You can call a static library function **scan(R)** in class **ImageToScan** which performs a scan of row **R** (0-based) of the image and returns its contents as a string of length **W**. Character i of return string represents the contents of pixel in row **R**, column i (0-based) of the original image: '0' for white, '1' for black. You must return your guess for the image as vector <string>. Your return must contain **H** elements, each having **W** characters, and only '0' and '1' characters in it. The j-th character in i-th element must describe the color of pixel in row i, column j of the image.

Your score for an individual test case will be calculated as follows. All unscanned rows of your return will be compared with the correct image pixel by pixel. The number of correctly guessed pixels CP and the total number of white pixels in the unscanned rows of the original image WP will be calculated. The score is then defined as max{0, (CP - WP) / **nb**}. Your overall score will be the sum of individual scores over all test cases.

Invalid return of any kind results in zero score for this test case. Attempts to scan non-existing row and attempts to scan the same row more than once are considered to be invalid. The return value of **scan** function for such scans is an empty string. Making an invalid scan results in zero score for this test case, even if your code subsequently returns a valid image description.

The test cases are generated as follows: **H**, **W** and **nLetter** are chosen randomly and uniformly. After this, the letters are drawn on the image one by one. For each letter its character (A..Z), font size (8..28) and font style (plain or bold) are chosen randomly and uniformly. The corresponding image is fetched from the pre-generated list of letter images. Next, the location for this letter is chosen at random, so that the letter is completely located within the image and it doesn't intersect any of the previously placed letters too much. To be more specific, for each letter its smallest bounding rectangle is stored, and for each pair of placed letters the sum of dimensions of intersection of their bounding rectangles can be at most half of the sum of the dimensions of any of these two bounding rectangles. If 100 attempts to place the new letter fail, it is placed without checking for intersections. Finally, **nb** is calculated from the resulting image.

The letter images used for generation are available for download. Each image is stored in a separate file with name consisting of three parts: letter, font style ('P' for plain, 'B' for bold) and font size. Each file starts from two lines containing height H_0 and width W_0 of the letter's bounding rectangle. It is continued with letter image description: H_0 rows, each containing W_0 characters '0'/'1'.

A visualizer is available for offline testing. You can check its source code for a precise implementation of test case generation and scoring.

Definition

Class:	ImageScanner
Method:	restore
Parameters:	int, int, int, int
Returns:	vector <string>
Method signature:	vector <string> restore(int H, int W, int nb, int nLetter)
(be sure your method is public)	

Available Libraries

Class:	ImageToScan
Method:	scan
Parameters:	int
Returns:	string
Sample Call:	val = ImageToScan::scan(row);

Notes

- The memory limit is 1024 MB and the time limit is 10 seconds (which includes only time spent in your code).
- There is no explicit code size limit. The implicit source code size limit is around 1 MB (it is not advisable to submit codes of size close to that or larger).
- There are 10 example test cases and 100 full submission test cases.

Constraints

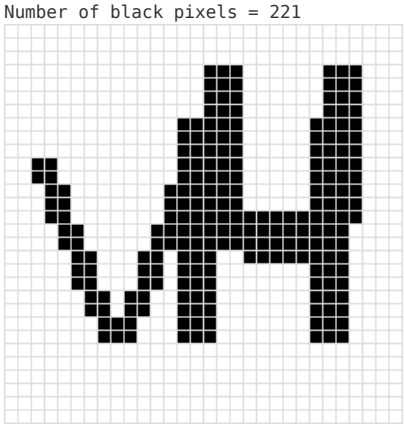
- **H** and **W** will each be between 50 and 300, inclusive (except for example 0).
- **nLetter** will be between **W*****H**/400 and **W*****H**/200, inclusive.

Examples

0)

```
seed = 1

H = 30
W = 30
Number of letters = 3
Number of black pixels = 221
```



1)

```
seed = 2

H = 225
W = 129
Number of letters = 119
Number of black pixels = 7818
```

Picture

2)

```
seed = 3

H = 98
W = 196
Number of letters = 58
Number of black pixels = 3205
```

Picture

3)

```
seed = 4

H = 97
W = 237
Number of letters = 88
Number of black pixels = 6361
```

Picture

4)

```
seed = 5

H = 212
W = 150
Number of letters = 127
Number of black pixels = 8798
```

Picture

5)

```
seed = 6

H = 132
W = 155
```

```
Number of letters = 93
Number of black pixels = 5512
```
Picture

6)

```
seed = 7

H = 278
W = 276
Number of letters = 283
Number of black pixels = 17821
```
Picture

7)

```
seed = 8

H = 299
W = 209
Number of letters = 167
Number of black pixels = 9768
```
Picture

8)

```
seed = 9

H = 100
W = 167
Number of letters = 63
Number of black pixels = 4102
```
Picture

9)

```
seed = 10

H = 270
W = 73
Number of letters = 89
Number of black pixels = 5919
```
Picture
