

PATCHES

Write a program that overlays a series of patches onto a frame. A patch is a rectangular grid of characters containing m (maximum 24) rows of n (maximum 79) characters each. The program reads in several patches and places copies of these patches onto a frame. A frame is just another rectangular grid of characters, initially filled with spaces. The input specifies the positions on the frame where copies of the patches are placed.

First, we establish some notation. Given an m by n grid, we address the upper leftmost position on the grid as position $(0,0)$ while the lower rightmost position as $(m-1,n-1)$. Positions on a grid are therefore addressed by their row and column coordinates that range from 0 to $m-1$ for rows and 0 to $n-1$ for columns.

Two input files will be required per run: a patches file and a commands file. The patches file contains all information regarding the patches (we shall have a maximum of ten patches). Patches are described one after another and each patch is specified by a name consisting of no more than ten lowercase letters (on one line), two integers (on another line) that indicate the patch's dimensions, and the contents of the patch (m lines of n characters each). Refer to the following example.

```
rect
7 9
+-----+
|         |
|         |
|         |
|         |
|         |
+-----+
ex
5 5
*   *
* *
 *
* *
*   *
dollar
1 1
$
```

The first line of the commands file contains the dimensions of the frame. The succeeding lines are a sequence of commands indicating where particular patches are overlaid onto the frame. A command follows the format: `[Row] [Col] [Patch-name]`. These fields are separated by at least one space. The following is an example of the contents of a commands file.

```
15 20
2 2 rect
4 5 rect
3 3 dollar
5 6 ex
0 0 dollar
1 2 dollar
0 19 dollar
14 0 dollar
14 19 dollar
```

The input files above indicate that we have three patches named `rect`, `ex`, and `dollar`, with sizes 7 by 9, 5 by 5, and 1 by 1, respectively, and a frame of size 15 by 20. Upon reading these two files, the program shall, in sequence, place the named patches on the specified positions on the frame. In the above example, a `rect` is placed on position (2,2) on the frame, then another `rect` on position (4,5), then a `dollar` on position (3,3), and so on. Placing a patch on position (x,y) on the frame entails aligning position (0,0) of the patch with position (x,y) on the frame. Upon placing a patch, previous contents on the corresponding portion of the frame are overwritten. After all patches have been overlaid onto the frame, the program should output the resulting frame. In the above example, we have the following resulting output:

```

$                                     $
$
+-----+
| $           |
| +-----+
| | *       * |
| | * *     |
| | *       |
+--| * *     |
   | *       |
   +-----+
$                                     $

```

This program is an exercise in arrays and structures. You could use dynamically allocated arrays or assume the maximum sizes specified (e.g., 24 by 79 for the patches). You should make good use of structures to organize your data. Use functions as necessary.

Important: There will be three sets of files: `Patches1.txt` and `Commands1.txt`, `Patches2.txt` and `Commands2.txt`, and, `Patches3.txt` and `Commands3.txt`. (These files should will be available in the course website). Your program should process all three sets **in one execution**, and write all output to a single file called **“Frames.txt”**. It is recommended that you have a function in your program whose parameters include the names of the patches file and commands file, so that you call this function three times in `main()`. The expected format of your output file is:

Programmer: <your name here>
CS 1044 Summer I 2004 Project 5

FRAME OUTPUT FOR Patches1.txt,Commands1.txt

```
$                $  
 $  
 +-----+  
 |$       |  
 | +-----+  
 | |*   * |  
 | | * *  |  
 | |  *   |  
 +--| * *  |  
   |*   * |  
   +-----+
```

```
$                $
```

FRAME OUTPUT FOR Patches2.txt,Commands2.txt

<frame output here>

FRAME OUTPUT FOR Patches3.txt,Commands3.txt

<frame output here>

Submitting your program:

Your program should follow the usual documentation and submission standards. Include the pledge statement in your program header. Refer to “Elements of Programming Style” available in the course website for some guidelines. As usual, submissions will be handled and auto-graded through the curator system, but will also be verified and hand-graded for compliance and documentation by the GTA.