Dibujar imágenes en una pizarra (en modo carácter) X45762_es

En este ejercicio os damos un programa a medio hacer que hay que completar. El programa trabaja con imágenes que queremos dibujar sobre una pizarra.
Más concretamente, tendremos un tipo de datos `Image`, que consiste en un nombre, una profundidad, una ubicación representada por dos naturales i, j y un vector de strings v. Usualmente llamaremos `image` a las variables de tipo `Image`.
Una pizarra será un vector de strings que típicamente llamaremos `board`.
Decimos que una `image` es válida si su `v` es una matriz rectangular de ciertas dimensiones $n \times m$ no nulas.
Decimos que un `board` es válido si es una matriz rectangular de ciertas dimensiones $N \times M$ no nulas.
Además, decimos que esta `image` encaja dentro del `board` si $i + n \leq N$ y $j + m \leq M$, donde i, j, es la ubicación de esa `image`.
El resultado de dibujar `image` sobre el `board` consiste en modificar `board` de manera que, para cada posición i', j' de `image` con un caracter diferente de `'.'`, se cumpla que `board`$[i + i'][j + j'] ==$`image`$[i'][j']$. Ningún otro caracter de `board` deberá ser alterado.
La función `main`, que ya se da hecha, lee una lista de imágenes, las ordena de mayor a menor produndidad, y las dibuja sobre una pizarra por ese orden.
Tendréis que implementar una función que lea una imagen de la entrada, una para calcular las dimensiones mínimas de una pizarra que hacen que todas las imágenes encajen en ella, y una función para dibujar una imagen sobre la pizarra. Además, tendréis que implementar una función que compara imagenes y se usa para ordenarlas. Una imagen es "menor que" otra si tiene mayor profundidad que la otra, o tiene la misma profundidad pero su nombre es menor que el nombre de la otra en orden lexicográfico.
Completad el siguiente código a medias para solucionar el ejercicio:

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

// Here you can add more includes if you wish.
// ...

using namespace std;

struct Image {
string name;
int depth;
int i, j;
vector<string> v;
};

typedef vector<Image> ListImages;

// Auxiliary functions (you can add more functions if you wish)
```

```
// Pre: The input has a description of an image with this format:
//      - First line: name depth i j n m
//      - n lines with m characters each (the contents of v)
//        These characters are different from whitespace, as we use '.' to repr
Image readImage()
{
// Implement this function.
//...
}

// Pre: listimages contains a non-empty list of valid images.
// Post: N,M are the dimensions of the minimum board such that
//       all of those images fit in it.
//       In other words, N,M must be the minimum naturals satisfying that,
//       for each image in listimages,
//       if i,j are its location and n,m are the dimensions of its v,
//       then i+n<=N and j+m<=M must be satisfied,
void computeMinimumBoardDimensions(const ListImages &listimages, int &N, int &M
// Implement this function.
//...
}

// Pre:  image is valid and board is valid and image fits in board.
// Post: image has been drawn on board. Nothing else has changed.
//       Recall that occurrences of character '.' in image are not printed on b
void drawImage(const Image &image, vector<string> &board)
{
// Implement this function.
//...
}

// Pre:  image1, image2 represent valid images.
// Post: Returns true iff one of the following conditions holds:
//         - depth of image1 is strictly bigger than depth of image2.
//         - image1 and image2 have same depth, but image1 has smaller name tha
bool compareImages(Image image1, Image image2)
{
// Implement this function.
//...
}

// Pre: listimages has a list of valid images.
// Post: prints on the output the result of drawing all of those
//       images on the minimum board such that all of them fit in,
//       and sorted by depth and name.
void drawListImages(const ListImages &listimages)
{
sort(listimages.begin(), listimages.end(), compareImages);
```

```
int N, M;
computeMinimumBoardDimensions(listimages, N, M);
vector<string> board(N, string(M, '.'));
for (int i = 0; i < int(listimages.size()); i++)
drawImage(listimages[i], board);
for (int i = 0; i < N; i++)
cout << board[i] << endl;
cout << endl;
}

int main()
{
int n;
cin >> n;
ListImages listimages(n);
for (int i = 0; i < n; i++)
listimages[i] = readImage();
drawListImages(listimages);
}
```

Entrada

La entrada consiste en la descripción de una lista de imágenes. No hay que preocuparse mucho de esto porque el `main` que os damos ya se encarga de llamar a las funciones correspondientes para tratar esta entrada.

Salida

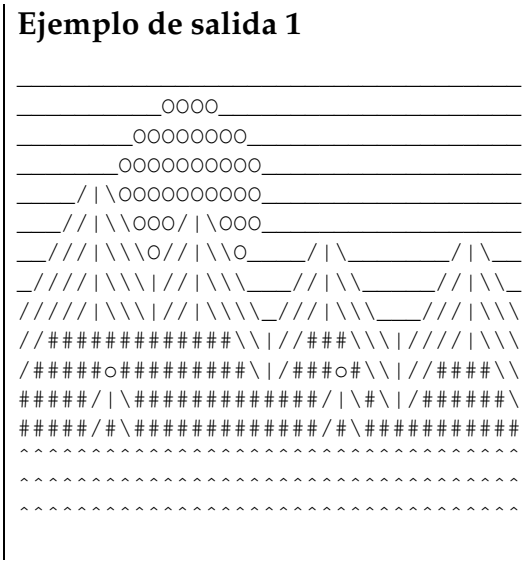
La salida muestra el resultado de escribir las imágenes sobre la pizarra por orden de profundidad y nombre. No hay que preocuparse mucho sobre esto porque el `main` que os damos ya se encarga de producir esta salida llamando a las funciones correspondientes.

Ejemplo de entrada 1

```
7
grass 8 9 0 7 35
####################################
####################################
####################################
####################################
####################################
####################################
####################################
person_a 1 10 5 3 3
.o.
/|\
/.\
person_b 1 10 21 3 3
.o.
/|\
/.\
mountains 5 4 0 8 35
..../|\.............................
...///|\\.../|\.....................
..///|\\\.//|\\......./|\......./|\..
.////|\\\|//|\\\.../|\\.....//|\\.
/////|\\\|//|\\\\.///|\\\...////|\\\
//.............\\|//...\\\|////|\\\
/...............\|/.....\\|//....\\
.........................\|/......\
sea 7 13 0 3 35
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
sky 10 0 0 10 35
____________________________________
____________________________________
____________________________________
____________________________________
____________________________________
____________________________________
____________________________________
____________________________________
____________________________________
____________________________________
sun 9 1 7 7 10
```

```
...OOOO...
.OOOOOOOO.
OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOOO
.OOOOOOOO.
...OOOO...
```

Ejemplo de salida 1



Ejemplo de entrada 2

```
100
name0 13 3 5 2 3
bbb
bbb
name1 0 6 11 5 4
kkkk
kkk.
kkk.
kkkk
kkkk
name2 9 9 5 3 5
bbbb.
.bbb.
bbb.b
name3 23 4 7 1 1
.
name4 4 1 11 5 4
p.pp
ppp.
pppp
pppp
ppp.
name5 4 0 2 4 5
mm..m
mmmmm
mmmmm
.mm.m
name6 8 9 11 5 1
r
r
r
.
r
name7 19 5 13 3 2
.w
ww
ww
name8 5 3 3 3 5
ww..w
www.w
ww.ww
name9 0 12 14 4 2
qq
qq
qq
qq
name10 21 16 8 1 5
ll.ll
name11 15 5 17 4 1
j
j
j
j
name12 25 15 4 4 5
bbbb.
bbbbb
bbbbb
.b.bb
name13 5 1 19 5 1
i
i
i
i
i
name14 19 4 8 3 1
d
d
d
name15 19 7 11 2 4
..hh
.h..
name16 2 10 4 5 2
qq
qq
qq
q.
qq
name17 13 12 11 5 1
n
n
```

```
.
n
n
name18 22 16 3 4 5
wwww.
.www.
.wwww
w.www
name19 19 10 1 3 4
h.hh
hhhh
h.hh
name20 17 3 4 5 5
.nn.n
n.nnn
nn.nn
nn...
nn.nn
name21 13 14 10 4 1
t
t
t
t
name22 3 11 1 5 5
l...l
lll.l
lllll
lllll
lllll
name23 20 4 6 5 3
mmm
mmm
mm.
mmm
mm.
name24 6 1 10 4 3
ppp
ppp
.pp
.pp
name25 5 10 1 2 1
u
u
name26 25 16 2 1 4
oooo
name27 4 14 12 4 4
hhh.
hhh.
hhhh
h.hh
name28 9 16 11 3 1
p
p
p
name29 20 4 7 1 4
u.uu
name30 8 0 9 3 2
bb
bb
..
name31 20 6 6 3 3
yyy
y.y
yy.
name32 16 0 8 2 5
yy.y.
y.yyy
name33 9 6 17 3 2
.e
e.
ee
name34 18 0 11 1 2
tt
name35 18 16 1 1 5
.kk.k
name36 11 12 13 4 1
q
q
q
q
name37 19 2 9 4 3
.bb
b.b
bbb
bbb
name38 16 15 5 5 3
bbb
bbb
b.b
bb.
bbb
name39 9 9 16 3 2
.t
.t
.t
name40 6 5 9 2 3
ww.
ww.
name41 4 15 13 3 5
yyyyy
yyyyy
.y.y.
name42 3 7 13 5 5
u..uu
uuu.u
uuu.u
uuuuu
uuuuu
name43 1 2 19 3 1
m
m
m
name44 6 11 1 1 3
tt.
name45 13 17 6 1 4
..p.
name46 20 0 16 2 2
zz
z.
name47 10 2 8 4 4
nn.n
nnnn
```

```
n..n
..nn
name48 4 7 1 3 2
ww
ww
.w
name49 4 0 7 4 1
.
o
o
o
name50 22 12 11 3 5
mmmmm
.mmmm
mmmmm
name51 24 3 13 3 5
.hhhh
hhhhh
hhhhh
name52 4 6 8 5 4
ff.f
ffff
.fff
f.ff
ffff
name53 3 14 7 2 3
vvv
v..
name54 13 10 11 1 5
bbbbb
name55 6 11 9 4 3
f.f
fff
fff
ff.
name56 17 9 2 4 1
n
n
n
n
name57 19 8 15 3 3
g.g
g.g
...
name58 5 4 8 5 2
.u
u.
.u
.u
uu
name59 2 6 12 3 2
..
.s
ss
name60 12 13 17 4 1
h
h
h
h
name61 9 19 5 1 4
.yy.
name62 0 3 11 5 3
ppp
ppp
.pp
p.p
.pp
name63 10 14 5 4 4
.jj.
..jj
jjjj
jjjj
name64 13 13 14 5 1
j
j
j
j
j
name65 14 17 15 3 3
y.y
yy.
yyy
name66 23 16 7 3 5
cc.c.
c.ccc
cccc.
name67 2 15 3 4 3
ee.
eee
eee
.ee
name68 22 1 10 5 5
uuuu.
uuu.u
uuuu.
u.uuu
uu..u
name69 2 2 3 2 1
a
a
name70 15 1 1 1 1
g
name71 8 15 12 1 5
mmmmm
name72 15 15 16 4 2
vv
vv
v.
vv
name73 7 17 0 3 3
qqq
qqq
qqq
name74 12 7 3 4 3
bbb
bbb
b.b
.b.
name75 14 2 0 5 5
qq.qq
..qqq
.qqqq
```

```
qq...
.qqq.
name76 23 5 4 3 5
.pppp
pppp.
p.ppp
name77 7 11 15 5 5
ll.l.
l.lll
llllll
llllll
llllll
name78 8 14 11 4 3
iii
i..
iii
.ii
name79 9 12 5 3 5
qqqqq
qqqq.
qqqqq
name80 15 2 7 1 1
v
name81 7 5 2 2 4
eeee
eee.
name82 6 14 11 2 5
rrrrr
rr.rr
name83 15 2 11 1 1
p
name84 25 3 0 2 2
.s
ss
name85 9 6 4 3 2
mm
mm
mm
name86 22 11 1 5 4
oooo
.ooo
oooo
oooo
...o
name87 0 8 13 2 5
c.ccc
cccc.
name88 6 9 0 2 5
.cccc
c..cc
name89 2 1 15 5 1
v
v
v
v
.
name90 0 1 3 1 5
ggg.g
name91 25 13 14 1 5
mmmm.
name92 13 2 2 5 1
l
l
l
l
l
name93 7 18 13 2 5
uuuuu
uuuuu
name94 10 11 8 4 5
yyyyy
...yy
yy.yy
yyyyy
name95 16 8 10 2 2
.x
xx
name96 21 7 14 1 2
ll
name97 10 13 15 5 4
mmm.
.mm.
m.mm
mmmm
mmm.
name98 17 10 12 1 2
kk
name99 11 10 11 3 5
fffff
f.fff
f.fff
```

Ejemplo de salida 2

```
..mm..m.ybbyt...zz..
.gmgggmgybppppvz..i
qqmammmonnppppuv...m
.slambmonnnppppvhh.m
sqlwwwbwnubppppvhh.m
qqewwewwuwwpppwhhj.i
.qeeemyyffwpkpk..je.
.wwbmmynfffkppwluu..
.wwbmmyyuffkkcuccce.
.cwccbbbf.fkkccccu..
```

```
cuncqqbbfffkkkkuuu..
.lthqqbbyfyfyuuuuul.
.lllqqqqqfffyfqq.lll
.lllqlqqqfffyqqqllll
.lllqqqvvvfrhhqqllll
.lleelbvj.trhyqqyyll
..keeejjjltihyyyyym.
qqqeeejjjctphiyhym..
qqq.eebwcccp.uuuuu..
qqqw.byy.....uuuuu..
```

Información del problema

Autor : PRO1
Generación : 2024-01-02 20:19:08