Problem C

Problem C
Shapes

Let's be given a rectangular matrix. There are M rows and N columns in that matrix. Let us also assume that M*N <= 60000. This matrix can be filled with the following characters (each character indicates a different colour):

. (dot) - the background
A-Z (only capital letters) - different colours

Each position in the matrix contains one character indicating either the background or the assigned colour. The same adjacent characters create a coloured area (shape). Consider the following matrix:

```
. . . . . . .
. . 1 2 3 . .
. . 8 A 4 . .
. . 7 6 5 . .
. . . . . . .
```

Letter A in the middle creates a consistent area with another letter (or letters) A placed in any position (positions) numbered from 1 to 8.

A Single letter A, surrounded only by background characters (i.e. dots) or letters other than A forms the consistent area, too.

Letters do not belong to the same consistent area if there is any gap between them i.e. they are separated by a different letter (representing a different colour) or by dot (background).
Let's consider the following matrices:

```
. . . . .       . . .
. A . A .  or   A B A
. . . . .       . . .
```

In each matrix presented above letters A do not form a single consistent area, but two separate areas.

The input contains a matrix with some shapes. Some of them have got neighbours (in different colours, of course). Different shapes are treated as neighbours if there exists any location where one shape adheres to the other. In that case we can say that the colours of these shapes are the neighbour colours.
For each colour occurring in the input matrix your program has to count how many consistent areas (shapes) have that colour and it should also write its neighbours' colours (if there are any).
Input
First line contains two numbers:

*   N - number of columns,
*   M - number of rows
and next lines contain the matrix with shapes.

Output
As a result, for every colour which occurs in the input matrix
you should write:
*   the name of that colour (the proper capital letter),
*   the number of consistent areas in that colour,
*   and colours of all the neighbours (if there are any).

EXAMPLE

Input

```
20 9
F....Q......Q......F
.....A.A........G...
.....AAA........B....
....A.........B.....
.AAA..AA.....B......
.AA...AA.BBBB.......
.A....AAAAAAB...FFFF
............B......F
F...........B......F
```

Output

```
Results :
---------
Color A - 2   Neighbours: B Q
Color B - 1   Neighbours: A G
Color F - 4   Neighbours:
Color G - 1   Neighbours: B
Color Q - 2   Neighbours: A
```

Solution

Rozwiazanie polega na zastosowaniu "algorytmu malarza" do wypelnienia obszarow w tablicy. Obszary do kolejnego wypelniania wybierane sa poprzez przegladanie kolejnych pozycji w tablicy. Jesli kolor w tablicy ro&#65533;ny jest od koloru tla to wywolujemy procedur&#65533; wypelniania od tej pozycji. Wypelnianie polega na zamianie koloru wypelnianego obszaru na kolor tla Zastosowana metoda wypelniania zapewnia "ustawienie" wszystkich punktow obszaru spojnego na kolor tla Jednoczesnie zliczana jest liczba obszarow w danym kolorze. Po ustawieniu wszystkich punktow macierzy na kolor tla wypisywana jest informacja koncowa

Tests

TEST 1

input 40 16

```
.......................................
........G..G...GG..G...G.............
.....G.G..G...G..G..G..G...G.G.......
........G..G..G..G..G..G..G...........
......AAAAAAAAAAAAAAAAAAAA...........
.....A....................A..........
....A..QQQQQQQ.....QQQQQQQ...A.......
....A...BBBBB.......BBBBB....A.......
```

```
         .QQQA...B...B..GGG..B...B....AQQQ.......
         .Q..A...BBBBB..GGG..BBBBB....A..Q.......
         .Q..A..........GGG...........A..Q.......
         ..QQA..........GGG...........AQQ........
         ....A...CCC...........C......A..........
         ....A......CCCCCCCCCCC.......A..........
         .....AAAA.....CCCCC......AAAA...........
         .........AAAAAAAAAAAAAAAA...............

output   Results :
        ---------
        Color A - 1   Neighbours: C G Q
        Color B - 2   Neighbours: Q
        Color C - 1   Neighbours: A
        Color G - 9   Neighbours: A
        Color Q - 4   Neighbours: A B

TEST 2

input        the matrix of the size:  600 X 100
         ADGJMPS.ADGJMPS ...
         DGJMPS.ADGJMPS ...
         ...

output   Results :
        ---------
        Color A - 88  Neighbours: D G S
        Color D - 88  Neighbours: A G J
        Color G - 88  Neighbours: A D J M
        Color J - 87  Neighbours: D G M P
        Color M - 87  Neighbours: G J P S
        Color P - 87  Neighbours: J M S
        Color S - 87  Neighbours: A M P

TEST 3

input        the matrix of the size:  60000 X 1
         AFKPU. AFKPU. AFKPU. AFKPU. ...

output   Results :
        ---------
        Color A - 10000    Neighbours: F
        Color F - 10000    Neighbours: A K
        Color K - 10000    Neighbours: F P
        Color P - 10000    Neighbours: K U
        Color U - 10000    Neighbours: P

TEST 4

input        matrix of the size:  1 X 60000
         A
         F
         K
         P
         U
         .
        .
        .
        .
output   Results :
         ---------
         Color A - 10000    Neighbours: F
         Color F - 10000    Neighbours: A K
         Color K - 10000    Neighbours: F P
         Color P - 10000    Neighbours: K U
         Color U - 10000    Neighbours: P
```

Listing

```
#include

long rows,cols;
char *tab;
unsigned long neighbours[30]; /*neighbours for given color*/
long shapes[30];      /*counters for shapes */

void markNeighbour(int n,char color){
  neighbours[n]|=((unsigned long)1L<<(color-'A'));
}

void printNeighbours(int n){
  int i;
  for (i=0;i<30;i++)
   if (neighbours[n] & ((unsigned long)1L<=0)  fill(x-1,y,ch);
  if (y+1=0)  fill(x,y-1,ch);
  if (y-1>=0 && x+1=0 && x-1>=0) fill(x-1,y-1,ch);
  if (y+1=0) fill(x-1,y+1,ch);
}

void scan(){
  long i,j;
  char ch;

  for (j=0;j0)
      {
     printf("Color %c - %ld\tNeighbours: ",i+'A',shapes[i]);
     printNeighbours(i);
     printf("\n");
      }
}

int main()
{
   reset();
   read();
   scan();
   results();
   if (tab) free(tab);
   return 0;
}
```


 [zwir](), [wierzej](), Mon Oct 28 23:01:26 MET DST 1996