Paint By Numbers

Paint By Numbers és un puzzle ben conegut. En aquest problema considerarem una versió unidimensional d'aquest puzzle. En aquest puzzle se li dóna al jugador una fila de n caselles. Les caselles estan numerades del 0 al $n-1$ d'esquerra a dreta. El jugador ha de pintar cada casella amb color negre o blanc. Denotarem amb 'X' les caselles negres i amb '_' les caselles blanques.

Al jugador se li dóna una seqüència $c = [c_0, \ldots, c_{k-1}]$ de k enters positius: les *pistes*. A continuació, ha de pintar les caselles de manera que les caselles negres a la fila formin exactament k blocs de caselles consecutives. A més a més, el nombre de caselles negres al bloc i-èssim (començant per 0) des de l'esquerra ha de ser igual a c_i. Per exemple, si les pistes són $c = [3, 4]$, llavors el puzzle, un cop resolt, ha de tenir exactament dos blocs de caselles negres consecutives: un de mida 3, i a continuació un altre de mida 4. Així, si $n = 10$ i $c = [3, 4]$, una solució que satisfà les pistes seria "_XXX__XXXX". Fixeu-vos que "XXXX_XXX__" no satisfà les pistes perquè els blocs de caselles negres no estan en l'ordre correcte. D'altra banda, "__XXXXXXX_" tampoc no satisfà les pistes perquè hi ha un únic bloc de caselles negres, i no pas dos blocs separats.

Se us dóna un puzzle de Paint By Numbers parcialment resolt. És a dir, coneixeu n i c, i a més a més sabeu que algunes de les caselles han de ser negres i d'altres han de ser blanques. La vostra tasca consisteix a deduir informació adicional sobre les caselles.

Més concretament, una *solució vàlida* és la que satisfà les pistes, i també fa que els colors de les caselles conegudes coincideixen. El vostre programa ha de trobar quines caselles s'han de pintar de negre en qualsevol solució vàlida, i quines caselles s'han de pintar de blanc en qualsevol solució vàlida.

Podeu suposar que sempre existeix com a mínim una solució vàlida amb la informació de l'entrada.

Detalls d'implementació

Se us demana que implementeu la següent funció (mètode):

- `string solve_puzzle(string s, int[] c)`.
  - `s`: string de mida n. Per a cada i ($0 \leq i \leq n-1$) el caràcter i és:
    - 'X', si la casella i s'ha de pintar de color negre,
    - '_', si la casella i s'ha de pintar de color blanc,
    - '.', si no hi ha informació sobre la casella i.
  - `c`: array de mida k que conté les pistes, tal i com s'han definit abans,

- la funció ha de retornar un string de mida n. Per a cada i ($0 \leq i \leq n-1$) el caràcter i de l'string de sortida ha de ser:
  - 'X', si la casella i és negra en totes les solucions vàlides,
  - '_', si la casella i és blanca en totes les solucions vàlides,
  - '?', en qualsevol altre cas (és a dir, si existeixen dues solucions vàlides tals que la casella i és negra en una de elles i blanca en l'altra).

En el llenguatge C la capçalera de la funció és una mica diferent:

- `void solve_puzzle(int n, char* s, int k, int* c, char* result)`
  - `n`: mida de l'string `s` (nombre de caselles),
  - `k`: mida de l'array `c` (nombre de pistes),
  - la resta de paràmetres són iguals que abans,
  - en comptes de retornar un string de n caràcters, la funció ha de desar la solució a l'string `result`.

Els codis ASCII dels caràcters utilitzats en aquest problema són:

- 'X': 88,
- '_': 95,
- '.': 46,
- '?': 63.

Si us plau, feu servir les plantilles donades per veure els detalls d'implementació en el vostre llenguatge de programació.

Exemples

Exemple 1

```
solve_puzzle("..........", [3, 4])
```

Aquestes són totes les solucions vàlides possibles del puzzle.

- "XXX_XXXX__",
- "XXX__XXXX_",
- "XXX___XXXX",
- "_XXX_XXXX_",
- "_XXX__XXXX",
- "__XXX_XXXX".

Es pot veure que les caselles amb índexs 2, 6 i 7 (començant per 0) són negres en qualsevol solució vàlida. Tota la resta de caselles poden ser-ho, però no és necessari. Per tant, la resposta correcta és "??X???XX??".

Exemple 2

```
solve_puzzle(".......", [3, 4])
```

En aquest exemple hi ha una única solució, i per tant la resposta correcta és "XXX_XXXX".

Exemple 3

```
solve_puzzle("..._._....", [3])
```

En aquest exemple podem deduir que la casella 4 també ha de ser blanca — no hi ha manera de fer cabre tres caselles negres consecutives entre les caselles blanques als índexs 3 i 5. Per tant, la resposta correcta és "`???___????`".

Exemple 4

```
solve_puzzle(".X........", [3])
```

Hi ha només dues solucions vàlides que satisfacin la descripció donada:

- "`XXX_______`",
- "`_XXX______`".

Per tant, la resposta correcta és "`?XX?______`".

Subtasques

Per a totes les subtasques es compleix que $1 \leq k \leq n$, i $1 \leq c_i \leq n$ per a cada $0 \leq i \leq k-1$.

1. (7 punts) $n \leq 20$, $k = 1$, s només conté '`.`' (puzzle buit),
2. (3 punts) $n \leq 20$, s només conté '`.`',
3. (22 punts) $n \leq 100$, s només conté '`.`',
4. (27 punts) $n \leq 100$, s només conté '`.`' i '`_`' (només hi ha informació sobre caselles blanques),
5. (21 punts) $n \leq 100$,
6. (10 punts) $n \leq 5\,000$, $k \leq 100$,
7. (10 punts) $n \leq 200\,000$, $k \leq 100$.

Grader de mostra

El grader de mostra llegeix l'entrada en el format següent:

- línia 1: string s,
- línia 2: enter k seguit per k enters $c_0, \ldots, c_{k-1}$.