Paint By Numbers

Paint By Numbers este un binecunoscut joc. Vom considera varianta simplă unidimensională a acestui joc. În acest joc, jucatorul are o linie formată din n celule. Celulele sunt numerotate de la 0 la $n-1$ de la stânga la dreapta. Jucătorul trebuie să coloreze fiecare celulă cu negru sau cu alb. Vom folosi 'X' pentru a simboliza celulele negre și '_' pentru a simboliza celulele albe.

Jucătorul primește un șir $c = [c_0, \ldots, c_{k-1}]$ de k numere întregi pozitive: *indiciile*. El trebuie să coloreze celulele în așa fel încât celulele negre să formeze exact k blocuri de celule consecutive. Mai mult, numărul celulelor negre din al i-lea bloc (indexat din 0) din stânga trebuie sa fie egal cu c_i. De exemplu , dacă indiciile sunt $c = [3, 4]$, jocul rezolvat trebuie să aibă exact două blocuri de celule negre consecutive: unul de lungime 3 și celălalt de lungime 4. Astfel, daca $n = 10$ și $c = [3, 4]$, o soluție care satisface indiciile este "_XXX__XXXX". Observați că "XXXX_XXX__" nu este o soluție care satisface indiciile: blocurile de celule negre nu sunt în ordinea corectă. Deasemenea, "__XXXXXXX_" nu este o soluție care satisface indiciile: există un singur bloc de celule negre, nu două separate.

Se dă un joc Paint By Numbers rezolvat parțial. Adică, se știe n și c, și în plus se cunoaște că anumite celule trebuie să fie negre și anumite celule trebuie să fie albe. Sarcina voastră este să deduceți informații suplimentare despre celule.

Mai precis, o *soluție validă* este una care satisface indiciile și respectă culorile din celulele cunoscute. Pogramul vostru trebuie să găsească celulele care sunt colorate în negru în orice soluție validă și celulele care sunt colorate în alb în orice soluție validă.

Se garantează că pentru datele de intrare există întotdeauna cel puțin o soluție validă.

Detalii de implementare

Trebuie să implementați următoarea funcție(metodă):

- `string solve_puzzle(string s, int[] c)`.
  - `s`: string de lungime n. Pentru fiecare i ($0 \leq i \leq n-1$) caracterul i este:
    - 'X', dacă celula i trebuie să fie neagră,
    - '_', dacă celula i trebuie să fie albă,
    - '.', dacă nu există informații despre celula i.
  - `c`: șir de lungime k conținând indiciile, așa cum au fost definite anterior,
  - funcția trebuie să returneze un string de lungime n. Pentru fiecare i ($0 \leq i \leq n-1$) caracterul i din stringul returnat trebuie să fie:
    - 'X', dacă celula i este neagră pentru orice soluție validă,

    - '_', dacă celula i este albă pentru orice soluție validă,
    - '?', altfel (adică, dacă există două soluții valide astfel încât celula i să fie neagră pentru una și albă pentru cealaltă).

În limbajul C signatura funcției este un pic diferită:

- `void solve_puzzle(int n, char* s, int k, int* c, char* result)`
    - `n`: lungimea stringului `s` (numărul de celule),
    - `k`: lungimea șirului `c` (numărul de indicii),
    - ceilalți parametrii sunt ca mai sus,
    - în loc să returneze un string de n caracteres, funcția trebuie să scrie răspunsul în stringul `result`.

Codurile ASCII ale caracterelor utilizate în această problemă sunt:

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

Vă rugăm utilizați fișierele exemplu pentru detalii de implementare în limbajul vostru de programare.

Exemple

Exemplul 1

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

Acestea sunt toate soluțiile valide ale jocului:

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

Se observă că celulele (indexate din 0) cu indicii 2, 6, și 7 sunt negre în toate soluțiile. Toate celelate celule pot fi negre, dar nu neapărat. Deci, răspunsul corect este "??X???XX??".

Exemplul 2

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

În acest exemplu soluția este unic determinată deci răspunsul corect este "XXX_XXXX".

Exemplul 3

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

În acest exemplu se poate deduce că și celula 4 trebuie să fie albă - nu este posibil să avem trei celule negre consecutive între pozițiile 3 și 5. Deci răspunsul corect este

“???___????”.

Exemplul 4

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

Există doar două soluții valide care se potrivesc descrierii de mai sus:

- “XXX_______”,
- “_XXX______”.

De aceea, răspunsul corect este “?XX?______”.

Subtaskuri

Pentru subtaskurile $1 \leq k \leq n$, și $1 \leq c_i \leq n$ pentru orice $0 \leq i \leq k-1$.

1. (7 puncte) $n \leq 20$, $k = 1$, s conține numai ‘.’ (empty puzzle),
2. (3 puncte) $n \leq 20$, s conține numai ‘.’,
3. (22 puncte) $n \leq 100$, s conține numai ‘.’,
4. (27 puncte) $n \leq 100$, s conține numai ‘.’ și ‘_’ (informații numai despre celule albe),
5. (21 puncte) $n \leq 100$,
6. (10 puncte) $n \leq 5\,000$, $k \leq 100$,
7. (10 puncte) $n \leq 200\,000$, $k \leq 100$.

Sample grader

Sample grader-ul citește datele de intrare în următorul format:

- linia 1: stringul s,
- linia 2: întregul k urmat de k întregi $c_0, \ldots, c_{k-1}$.