Malen nach Zahlen

Malen nach Zahlen ist ein bekanntes Puzzlespiel. Wir betrachten eine einfache eindimensionale Version dieses Puzzles. Der Spieler hat eine Reihe von n Zellen. Die Zellen sind von links nach rechts von 0 bis $n-1$ durchnummeriert. Der Spieler muss jede Zelle entweder schwarz oder weiss färben. Wir verwenden 'X' für schwarze und '_' für weisse Zellen.

Dem Spieler wird eine Folge $c = [c_0, \ldots, c_{k-1}]$ von k positiven Integers vorgegeben: die *Hinweise*. Er muss die Zellen auf eine Art und Weise einfärben, dass die schwarzen Zellen in der Zeile exakt k Blöcke von aufeinanderfolgenden Zellen bilden. Zusätzlich sollte von links beginnend die Anzahl schwarzer Zellen im i-ten Block (beginnend mit 0) gleich c_i sein.

Zum Beispiel: Wenn die Hinweise $c = [3, 4]$ sind, muss das gelöste Puzzle exakt zwei Blöcke aufeinander folgender schwarzer Zellen enthalten: einen der Länge 3, dann einen anderen der Länge 4. Wenn etwa $n = 10$ und $c = [3, 4]$ ist, so ist "_XXX__XXXX" eine richtige Lösung. Beachte, dass "XXXX_XXX__" keine richtige Lösung ist: Die Blöcke mit schwarzen Zellen sind nicht in der richtigen Reihenfolge. Auch "__XXXXXXX_" ist keine richtige Lösung, da es nur einen Block schwarzer Zellen gibt und nicht zwei getrennte Blöcke.

Du bekommst ein teilweise gelöstes "Malen nach Zahlen" Puzzle. Das heisst, du kennst n und c und weisst zusätzlich, dass einige Zellen schwarz und einige Zellen weiss sein müssen. Deine Aufgabe ist es, zusätzliche Informationen über die Zellen herzuleiten.

Eine gültige Lösung muss alle Hinweise erfüllen. Ebenso muss eine Übereinstimmung mit den vorgegebenen Zellfarben vorhanden sein. Du sollst dann die Zellen finden, die in allen gültigen Lösungen schwarz bzw. weiss gefärbt sind. Du kannst annehmen, dass die Eingabe derart gestaltet ist, dass mindestens eine richtige Lösung existiert.

Details zur Implementierung

Du sollst die folgende Funktion (Methode) implementieren:

- `string solve_puzzle(string s, int[] c)`.
  - `s`: String der Länge n. Für jedes i ($0 \leq i \leq n-1$) ist das i-te Zeichen:
    - 'X', wenn die Zelle i schwarz sein muss,
    - '_', wenn die Zelle i weiss sein muss,
    - '.', wennn es keine Information über die Zelle i gibt.
  - `c`: Array der Länge k welches Hinweise wie oben definiert enthält,
  - die Funktion soll einen String der Länge n zurückgeben. Für jedes i (

($0 \leq i \leq n-1$) soll das i-te Zeichen des Outputstrings wie folgt sein:
    - 'X', wenn die Zelle i in jeder gültigen Lösung schwarz ist,
    - '_', wenn die Zelle i in jeder gültigen Lösung weiss ist,
    - '?' sonst (d.h., wenn zwei gültige Lösungen existieren, sodass in der einen Lösung die Zelle schwarz und in der anderen Lösung die Zelle weiss gefärbt ist).

In der Sprache C ist die Funktionssignatur etwas anders:

- `void solve_puzzle(int n, char* s, int k, int* c, char* result)`
    - `n`: Länge des Strings `s` (Anzahl der Zellen),
    - `k`: Länge des Arrays `c` (Anzahl der Hinweise),
    - die anderen Parameter sind gleich wie oben,
    - anstatt der Rückgabe eines Strings mit n Zeichen, soll die Funktion das Ergebnis in den String `result` schreiben.

Die ASCII-Codes der in diesem Problem verwendeten Zeichen sind:

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

Bitte verwende die zur Verfügung gestellten Templatefiles für die Details der Implementierung in deiner Programmiersprache.

Beispiele

Beispiel 1

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

Alle möglichen Lösungen des Puzzles sind:

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

Man kann beobachten, dass die Zellen mit (0-basierten) Indizes 2, 6 und 7 in jeder gültigen Lösung schwarz sind. Jede der anderen Zellen kann, muss aber nicht schwarz sein. Daher ist die richtige Antwort "??X???XX??".

Beispiel 2

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

In diesem Beispiel ist die vollständige Lösung eindeutig bestimmt und die richtige Antwort ist "XXX_XXXX".

Beispiel 3

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

In diesem Beispiel können wir herleiten, dass die Zelle 4 auch weiss sein muss -- es gibt nämlich keine Möglichkeit, drei aufeinanderfolgende schwarze Zellen zwischen die weissen Zellen mit den Indizes 3 und 5 hineinzupassen. Daher ist die richtige Antwort "`???___????`".

Beispiel 4

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

Hier gibt es nur zwei gültige Lösungen, die zur obigen Beschreibung passen.

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

Die richtige Antwort ist daher "`?XX?______`".

Subtasks

In allen Subtasks ist $1 \leq k \leq n$, und $1 \leq c_i \leq n$ für jedes $0 \leq i \leq k-1$.

1. (7 Punkte) $n \leq 20$, $k = 1$, s enthält nur '.' (leeres Puzzle),
2. (3 Punkte) $n \leq 20$, s enthält nur '.',
3. (22 Punkte) $n \leq 100$, s enthält nur '.',
4. (27 Punkte) $n \leq 100$, s enthält nur '.' und '_' (Information nur über weisse Zellen),
5. (21 Punkte) $n \leq 100$,
6. (10 Punkte) $n \leq 5\,000$, $k \leq 100$,
7. (10 Punkte) $n \leq 200\,000$, $k \leq 100$.

Beispielgrader

Der Beispielgrader liest die Eingabe im folgenden Format:

- Zeile 1: String s,
- Zeile 2: Integer k gefolgt von k Integers $c_0, \ldots, c_{k-1}$.