



## 古代の機械 2 (Ancient Machine 2)

考古学者のビ太郎とビバ子は JOI 国の遺跡で発掘調査を行った。遺跡でビ太郎は古代の石板を、ビバ子は古代の機械を発見した。

ビ太郎は研究の結果、石板には文字列  $S$  が書かれており、その長さは  $N$ 、その各文字は '0' または '1' であるということを突き止めた。しかしながら、文字列  $S$  の各文字がそれぞれ何であるかは未だにわかっていない。

ビバ子は研究の結果、機械の使い方を特定した。機械に石板を設置し、1つの整数  $m$  と2つの数列  $a, b$  を入力して質問することができる。ただし、整数  $m$ 、数列  $a, b$  は次の条件を満たしていなければならない。

- 整数  $m$  は 1 以上 1002 以下である。
- 数列  $a$  と数列  $b$  の長さはともに  $m$  である。
- 数列  $a$  と数列  $b$  の各要素は 0 以上  $m-1$  以下の整数である。

機械に石板を設置し、1つの整数  $m$  と2つの数列  $a, b$  を入力して質問すると以下のように機械は動作し、1つの数を表示する。

1. 機械の記憶領域に 0 が設定される。
2. 機械は処理を  $N$  回行う。  $i+1$  回目 ( $0 \leq i \leq N-1$ ) の処理は以下のように行われる。

現在、機械の記憶領域に設定されている値を  $x$  とする。機械は  $S_i$  を参照する。

ここで、 $S_i$  とは、先頭の文字を 0 番目と数えたときの文字列  $S$  の  $i$  番目の文字のことを指す。

- $S_i$  が '0' であるとき、機械の記憶領域に  $a_x$  が設定される。  
ここで、 $a_x$  とは、先頭の要素を 0 番目と数えたときの数列  $a$  の  $x$  番目の要素のことを指す。
- $S_i$  が '1' であるとき、機械の記憶領域に  $b_x$  が設定される。

ここで、 $b_x$  とは、先頭の要素を 0 番目と数えたときの数列  $b$  の  $x$  番目の要素のことを指す。

3. 機械は最終的に記憶領域に設定されている値を表示する。

ビ太郎は機械を用いて石板に書かれている文字列を特定したい。しかしながら、機械は非常に脆いため、質問を行う回数は 1000 回以下でなければならない。さらに、各質問で入力する整数  $m$  の最大値をなるべく小さくしたい。

機械を用いて石板に書かれている文字列を特定するためのプログラムを作成せよ。



## 実装の詳細

あなたは 1 つのファイルを提出しなければならない。

あなたの提出するファイルは `ancient2.cpp` という名前である。このファイルは以下の関数を実装していなければならない。そのプログラムは `#include` プリプロセッサ指令によって `ancient2.h` を読み込むこと。

- `std::string Solve(int N)`

この関数は各テストケースにおいて 1 回だけ呼び出される。この関数は石板に書かれた文字列  $S$  と等しい文字列を返さなければならない。

- 引数  $N$  は石板に書かれた文字列  $S$  の長さ  $N$  である。
- 戻り値は、長さ  $N$  の文字列でなければならない。長さが  $N$  でない場合、不正解 [1] と判定される。
- 戻り値の各文字は '0' または '1' でなければならない。これが満たされていない場合、不正解 [2] と判定される。
- 戻り値は石板に書かれた文字列  $S$  と一致していなければならない。これが満たされていない場合、不正解 [3] と判定される。

あなたのプログラムは以下の関数を呼び出すことができる。

- ★ `int Query(int m, std::vector<int> a, std::vector<int> b)`

この関数を用いて、機械に質問を行う。

- ◇ 引数  $m$  は、機械に入力する 1 つの整数  $m$  を表す。
- ◇ 引数  $a, b$  は、機械に入力する 2 つの数列  $a, b$  を表す。
- ◇ 戻り値は、機械に石板を設置し、上記の入力を行い質問した際に表示される 1 つの数である。
- ◇ 引数  $m$  は、1 以上 1002 以下でなければならない。これが満たされていない場合、不正解 [4] と判定される。
- ◇ 引数  $a, b$  の長さはともに  $m$  でなければならない。これが満たされていない場合、不正解 [5] と判定される。
- ◇ 引数  $a, b$  の各要素は 0 以上  $m-1$  以下でなければならない。これが満たされていない場合、不正解 [6] と判定される。
- ◇ 関数 `Query` を 1000 回を超えて呼び出してはならない。1000 回を超えて呼び出した場合、不正解 [7] と判定される。



## 重要な注意

- 内部での使用のために他の関数を実装したり、グローバル変数を宣言するのは自由である。
- あなたの提出したプログラムは、標準入力・標準出力、あるいは他のファイルといかなる方法でもやりとりしてはならない。ただし、標準エラー出力にデバッグ情報等を出力することは許される。

## コンパイル・実行の方法

作成したプログラムをテストするための、採点プログラムのサンプルが、コンテストサイトからダウンロードできるアーカイブの中に含まれている。このアーカイブには、提出しなければならないファイルのサンプルも含まれている。

採点プログラムのサンプルは1つのファイルからなる。そのファイルは `grader.cpp` である。作成したプログラムをテストするには、これらのファイル `grader.cpp`, `ancient2.cpp`, `ancient2.h` を同じディレクトリに置き、次のようにコマンドを実行する。なお、アーカイブの中に含まれている `compile.sh` というファイルを代わりに実行してもよい。

```
g++ -std=gnu++17 -O2 -o grader grader.cpp ancient2.cpp
```

コンパイルが成功すれば、`grader` という実行ファイルが生成される。

実際の採点プログラムは、採点プログラムのサンプルとは異なることに注意すること。採点プログラムのサンプルは単一のプロセスとして起動する。このプログラムは、標準入力から入力を読み込み、標準出力に結果を出力する。

## 採点プログラムのサンプルの入力

採点プログラムのサンプルは標準入力から以下の形式で入力を読み込む。

```
N
S
```

## 採点プログラムのサンプルの出力

採点プログラムのサンプルは標準出力へ以下の情報を出力する（引用符は実際には出力されない）。

- 正解の場合、関数 `Query` の引数 `m` の最大値が “Accepted: 22” のように出力される。ただし、関数 `Query` が1回も呼び出されことなく正解した場合、“Accepted: 0” と出力される。



- 不正解の場合、不正解の種類が“Wrong Answer [4]”のように出力される。

実行するプログラムが複数の不正解の条件を満たした場合、表示される不正解の種類はそれらのうち 1 つのみである。

## 制約

すべての入力データは以下の条件を満たす。

- $N = 1000$ .
- $S$  は長さ  $N$  の文字列である。
- $S$  の各文字は ‘0’ または ‘1’ である。

## 採点に関する注意

すべてのテストケースについて、実際の採点プログラムは適応的 (adaptive) ではない。これは、採点プログラムは初めから固定された答えを持つということを意味する。

## 採点基準

この課題におけるテストケースに対して、1 つでも不正解があった場合、この課題の得点は 0 点となる。

また、この課題におけるすべてのテストケースに正解した場合、この課題のすべてのテストケースで呼び出された関数 `Query` の引数  $m$  の最大値を  $M$  として、以下のように採点される。ただし、すべてのテストケースにおいて関数 `Query` が 1 回も呼び出されることなく正解した場合、 $M = 0$  として採点される。

- $103 \leq M \leq 1002$  のとき、 $10 + \left\lfloor \frac{(1002-M)^2}{9000} \right\rfloor$  点。
- $0 \leq M \leq 102$  のとき、100 点。

ここで、 $\lfloor x \rfloor$  は  $x$  を超えない最大の整数を表す。



## やりとりの例

採点プログラムのサンプルが読み込む入力の例と、それに対応する関数の呼び出しの例を以下に示す。

| 入力例 1    | 関数の呼び出しの例   |       |                                      |     |
|----------|-------------|-------|--------------------------------------|-----|
|          | Solve の呼び出し | 戻り値   | Query の呼び出し                          | 戻り値 |
| 3<br>110 | Solve(3)    |       |                                      |     |
|          |             |       | Query(4, [3, 3, 2, 2], [2, 2, 1, 0]) | 3   |
|          |             |       | Query(2, [0, 1], [1, 0])             | 0   |
|          |             |       | Query(1, [0], [0])                   | 0   |
|          |             |       | Query(3, [1, 1, 1], [1, 1, 1])       | 1   |
|          |             | "110" |                                      |     |

石板に書かれている文字列  $S$  が“110”であるとき、機械に石板を設置し、 $(m, a, b) = (4, [3, 3, 2, 2], [2, 2, 1, 0])$  を入力して質問すると、以下のように機械は動作する。

1. 機械の記憶領域に 0 が設定される。
2. 1 回目の処理では、 $S_0$  は ‘1’ であるため、機械の記憶領域に  $b_0$  すなわち 2 が設定される。
3. 2 回目の処理では、 $S_1$  は ‘1’ であるため、機械の記憶領域に  $b_2$  すなわち 1 が設定される。
4. 3 回目の処理では、 $S_2$  は ‘0’ であるため、機械の記憶領域に  $a_1$  すなわち 3 が設定される。
5. 最終的に記憶領域に設定されている値は 3 であるため、3 が表示される。

この入力例は  $N = 1000$  という制約を満たさないことに注意すること。

コンテストサイトからダウンロードできるファイルのうち、sample-02.txt は制約を満たす。