Amulet sir Bajcelota

PREOI 2025
Dzień 5 - 29 stycznia 2025

Kod zadania: **szy**
Limit pamięci: **512 MiB**

Dwoje kochanków, rycerz Bajcelot i księżniczka Bajtyna, zostało rozdzielenych przez siedem gór, siedem lasów i siedem mórz.

Sir Bajcelot ma do przekazania Bajtynie list miłosny. Ponieważ Bajcelot jest człowiekiem konkretnym, list ten składa się z co najwyżej 43 liter, z których każda to 'A' lub 'B' (bez żadnych spacji ani znaków interpunkcyjnych).

Swego płomienistego listu miłosnego Bajcelot nie może wysłać na taką odległość w żaden konwencjonalny sposób. Na szczęście tuż przed rozstaniem, kochankowie zaopatrzyli się w parę magicznych amuletów:

Amulety składają się z 64 kryształów ułożonych w 8 wierszy po 8 kolumn. Każdy z kryształów ma 2 możliwe stany: uśpiony i aktywowany. Sir Bajcelot może aktywować wybrane przez siebie kryształy i dokładnie raz wypowiedzieć specjalne zaklęcie. Po jego inkantacji, amulet Bajtyny ustawi się w dokładnie tę samą konfigurację (czyli dokładnie te same kryształy, co u Bajcelot się u niej aktywują, a dokładnie te same będą zgaszone).

Niestety, amulety zostały zakupione od dosyć szemranego obwoźnego czarownika i cały czas się psują. W taki mianowicie sposób, że sir Bajcelot traci kontrolę nad kryształami w wierszu nr X oraz kolumnie nr Y: nie może ich aktywować ani uśpić, a po wypowiedzeniu zaklęcia kryształy w wierszu X i kolumnie Y przybierają stany niemożliwe do przewidzenia (czyli u Bajtyny mogą się one dowolnie aktywować lub uśpić).

Bajtyna nie wie, który wiersz i która kolumna uległy awarii ani nie jest w stanie w żaden inny sposób niż przez amulet porozumieć się z Bajcelotem. Mimo tych komplikacji, Bajtyna chciałaby umieć odczytać niezakłóconą wiadomość od swego lubego.

Jak widać, żeby rozwiązać problem zakochanej pary, potrzeba więcej niż czarodzieja. Potrzeba informatyka! Pomóż Bajcelotowi i Bajtynie się porozumieć. Napisz dwie funkcje:

- Pierwsza z nich dla danej wiadomości S, zepsutego wiersza X i zepsutej kolumny Y wyznaczy, które kryształy w amulecie należy zgasić, a które aktywować. Tę funkcję przekażesz sir Bajcelotowi.
- Druga z nich dla danej konfiguracji zapalonych i zgaszonych kryształów, poprawnie odczyta zaszyfrowaną wiadomość. Tę funkcję przekażesz księżniczce Bajtynie.

Funkcje te zapiszesz w jednym programie. Jedną jego kopie otrzyma Bajcelot, drugą – Bajtyna. Najpierw Bajcelot wykona funkcję pierwszą, a następnie Bajtyna funkcję drugą. Pomiędzy kopiami programów nie ma możliwości komunikacji w inny sposób niż poprzez działanie amuletu.

Implementacja

Twoim zadaniem jest zaimplementowanie poniższych funkcji `Zaszyfruj` i `Odszyfruj` w jednym pliku.
Dla każdego pliku testowego Twój program będzie kopiowany i uruchamiany w dwóch niewspółdzielących pamięci instancjach. Dla każdego przypadku testowego (których w jednym pliku może być więcej niż jeden), najpierw będzie wykonywana funkcja `Zaszyfruj` z pierwszej instancji, a później `Odszyfruj` z instancji drugiej.

Enkoder:

- `void Zaszyfruj(int X, int Y, int N, std::string S)`
  - Liczba X ($0 \leq X \leq 7$) oznacza numer zepsutego wiersza.
  - Liczba Y ($0 \leq Y \leq 7$) oznacza numer zepsutej kolumny.

  Jak widać wiersze i kolumny numerowane są kolejnymi liczbami całkowitymi od 0 do 7.
  - Liczba N($1 \leq N \leq 43$) oznacza długość listu B.
  - W zmiennej S znajduje się treść listu Bajcelota – słowo składające się z N liter, z których każda to 'A' lub 'B'.

  Do ustawiania stanów amuletu powinieneś korzystać z udostępnionej ci zewnętrznej funkcji

  - `void Oznacz(int a, int b, int c)`

    Funkcja ta ustawia stan kryształu w a-tym wierszu i b-tej kolumnie na:
    - zgaszony, jeśli $c = 0$
    - aktywny, jeśli $c = 1$.

    Nie powinieneś ustawiać żadnego stanu w kryształach należących do zepsutego wiersza i zepsutej kolumny. Jeśli wywołasz tę funkcję z $a = X$ lub $b = Y$, Twój program otrzyma werdykt `Zła odpowiedź` z kodem błędu 1.
    Jeśli wykonasz tę funkcję z $c \notin \{0, 1\}$, Twój program otrzyma werdykt `Zła odpowiedź` z kodem błędu o2. Stan każdego kryształu, nienależącego do zepsutego wiersza ani kolumny, powinieneś ustawić dokładnie raz.
    Jeśli stan jakiegoś kryształu ustawisz więcej niż raz, Twój program otrzyma werdykt `Zła odpowiedź` z kodem błędu 3.

  Jeśli nie ustawisz stanu jakiegoś niezepsutego kryształu, Twój program otrzyma werdykt `Zła odpowiedź` z kodem błędu 4.

Dekoder:

- `std::string Odszyfruj(int N, std::vector<std::vector<int> > T)`
  - Liczba N($1 \leq N \leq 43$) oznacza długość listu Bajcelota.
  - T stanowi wektor wektorów o wymiarach 8 na 8, w którym w komórce `T[i][j]` (dla $0 \leq i, j \leq 7$) znajduje się informacja, jaki stan kryształu w wierszu i-tym i kolumnie j-tej zaobserwowała Bajtyna.

  Funkcja ta powinna zwrócić jedno słowo, składające się z N liter, z których każda to 'A' lub 'B'. Słowo to powinno oznaczać odszyfrowaną przez Bajtynę wiadomość sir Bajcelota.

  Jeśli zwrócone słowo będzie składało się z więcej niż 43 znaków, Twój program otrzyma werdykt `Zła odpowiedź` z kodem błędu 5.
  Jeśli zwrócone słowo będzie zawierało znak, niebędący literą 'A' ani 'B', Twój program otrzyma werdykt `Zła odpowiedź` z kodem błędu 6.

Program **musi** zaczynać się od linijki `#include "szylib.h"`

Twój program może używać zmiennych globalnych, ale **nie będą** one współdzielone pomiędzy programami przekazanymi Bajcelotowi i Bajtynie.

Ocenianie

We wszystkich testach limit czasu wynosi 4 sekundy.

Pliki testowe w tym zadaniu mogą składać się z wielu przypadków testowych. W każdym z plików testowych może być od jednego do co najwyżej 20 000 przypadków testowych.

Przypominamy: funkcje `Zaszyfruj` i `Odszyfruj` będą wykonywane dokładnie raz w każdym z przypadków testowych, natomiast Twój cały program **nie** będzie uruchamiany od początku dla każdego z przypadków testowych. Oznacza to m.in. że ustawione przez Ciebie zmienne globalne nie będą się automatycznie zerowały pomiędzy kolejnymi przypadkami testowymi.

Jeżeli Twój program dla jakiegokolwiek pliku testowego zakończy się z dowolnym rodzajem werdyktu `Zła odpowiedź` lub zostanie przerwany ze względu na np. przekroczenie limitu czasu lub naruszenie ochrony pamięci, otrzyma 0 punktów.

Niech dla danego pliku testowego L oznacza największą liczbę całkowitą taką, że funkcja `Odszyfruj` poprawnie odkodowała wszystkie wiadomości o $N \leq L$. Jeżeli Twój program poprawnie odkodował wszystkie wiadomości, liczba L ustawiana jest na 43.

Jako L' oznaczmy najmniejszą liczbę L pomiędzy wszystkimi plikami testowymi. Liczba punktów otrzymywana przez Twój program jest wyznaczana zgodnie z poniższą tabelką:

L'	**Liczba punktów**
0	0
1	5
2	8
3	10
4	11
5	13
6	14
7	16
8	18
9	19
10	21
11	22
12	24
13	26
14	27
15	29
16	30
17	32
18	34
19	35
20	37
21	38

L'	**Liczba punktów**
22	40
23	42
24	43
25	45
26	46
27	48
28	50
29	51
30	53
31	54
32	56
33	57
34	59
35	60
36	62
37	65
38	68
39	71
40	74
41	77
42	84
43	100

Kompilacja

Dla uczestników korzystających z systemu operacyjnego Linux (lub WSL itp.) w zakładce 'Pliki' została umieszczona przykładowa biblioteczka do tego zadania (może ona się różnić od tej używanej do oceny programów w systemie). Archiwum `dlazaw.zip` należy rozpakować i swój kod wkleić do pliku `szy.cpp`. Pozostałych plików nie należy modyfikować.

Dla uczestników nie mających dostępu do Linux'a udostępniamy środowisko w przeglądarce. Aby z niego skorzystać, należy wejść w link, założyć konto (jeśli takowego się jeszcze nie ma) i w prawym górnym rogu kliknąć `Use this template -> Open in codespaces`.

W archiwum (lub w środowisku przeglądarkowym) znajduje się skrypt `compile.sh`. Aby z niego skorzystać należy wykonać w terminalu komendę:
◦ `./compile.sh 0`
albo
◦ `./compile.sh 1`
odpowiednio bez/z pomocniczymi informacjami.

Skrypt ten stworzy plik `szy.e`, który następnie można uruchomić komendą `./szy.e` i ręcznie wpisać wejście lub podać ścieżkę do testu, np.: `./szy.e < in/szy0a.in`.

Testy mają następującą strukturę:
Q – liczba przypadków testowych
X_1 Y_1 N_1 S_1 – parametry X, Y, N, S w pierwszym przypadku testowym
X_2 Y_2 N_2 S_2 – parametry X, Y, N, S w drugim przypadku testowym
⋮
X_Q Y_Q N_Q S_Q – parametry X, Y, N, S w Q-tym przypadku testowym

Wyjściem programu będzie albo:
`ANSWER L`, gdzie L jest zdefiniowane w sekcji 'Ocenianie'
`ERROR ANSWER_ERROR_CODE W`, gdzie W jest jednym z kodów błędów zdefiniowanych w poprzednich sekcjach.
`ERROR ***`, gdzie `***` to inny rodzaj błędu biblioteczki, w tym przypadku pojawi się również opis tego rodzaju błędu.

Uwaga. Warto zwrócić uwagę na to, że werdykt `ANSWER 0`, choć oznacza formalną poprawność przeprowadzonej komunikacji, prawdopodobnie oznacza, że Twój program otrzyma 0 punktów.

Przykłady

Wejście dla testu `szy0a`:

```
2
0 0 1 B
2 5 8 AAAABBBB
```

Wyjaśnienie do przykładu: Test ten składa się z dwóch przypadków testowych.
W pierwszym przypadku testowym zepsute są zerowy wiersz i zerowa kolumna, a sir Bajcelot chce przekazać Bajtynie jednoliterową wiadomość: `A`.
W drugim przypadku testowym zepsute są drugi wiersz i piątą kolumna, a sir Bajcelot chce przekazać Bajtynie ośmioliterową wiadomość: `AAAABBBB`.

Wejście dla testu szy0b:

```
30
3 1 1 A
1 4 1 A
6 6 2 AA
1 1 2 BB
3 1 3 BAB
7 4 3 AAB
6 4 4 BAAB
6 7 4 BABA
3 3 5 BABBA
1 5 5 ABBBA
4 3 6 ABBBBB
2 1 6 ABAAAA
6 0 7 AAABABA
6 6 7 BBABBAA
0 4 8 AABAABAB
2 1 8 AABBBBBA
2 0 9 BABABBAAA
1 5 9 BBAAABABB
6 7 10 BAAABAAABB
1 7 10 BBBBBBBABA
2 6 12 AABAABABABAB
3 4 15 BBAABAAAABABAAB
5 6 18 BAAAABBABABBBABBAB
7 0 22 BABBAABAAABBABBBBBBABA
2 0 26 AAAABBABBAAAAABABABBAABAAA
0 7 30 AAABBBAAABAABBBBAABBAAABBBABBB
2 7 34 BABAABBAABABBABAABBABBABAABBBBABBB
2 5 38 BBBBAABAABAABABABBBBBAAABBABAAABAAABBB
5 2 41 AABABBAAABBABAAAABBABABBAAAAAABBABBABBABA
1 0 43 AABBABBBBABABBBABBBBAAAAAABABAAABBBAABBAAAB
```