First Day, Task 1
Mastermind (2 sec, 64MB)

Mastermind: Върховният противник (mastermind)

Играта, наречена "Mastermind" се играе от двама играчи: *кодосъставител* и *кодоразбивач*. Кодосъставителят съставя код, състоящ се от n числа, всяко в интервала от 0 до $m-1$ (включително). Допускат се повторения. Кодоразбивачът се опитва да пробие този код. Той прави предположение, състоящо се от n числа, всяко в интервала от 0 до $m-1$. След това той иска подсказка от кодосъставителя. Кодосъставителят му отговаря с две числа: c и p. Те показват, че кодоразбивачът е познал c цифри, които са на правилните позиции, и други p цифри, които се срещат в кода, но са на грешни позиции. Целта на играта е да се познае тайният код за колкото се може по-малко хода. Ето и един пример ($n = 4, m = 6$):

1. Кодосъставителят е избрал кода $(3,3,5,4)$.
2. Кодоразбивачът предлага $(3,3,3,3)$.
3. Отговорът на кодосъставителя е $c = 2, p = 0$. Двете тройки в началото на кода са налични и на правилни позиции. Забележете, че останалите две тройки не се определят като "налични, но на грешни позиции" защото има само две тройки в тайния код.
4. Кодоразбивачът предлага $(4,4,3,3)$.
5. Отговорът е $c = 0, p = 3$. Този път едната от двете четворки и двете тройки са налични, но на грешни позиции.
6. Следващото предположение е $(3,3,4,5)$.
7. Отговорът за него е $c = 2, p = 2$.
8. Следва предположението $(3,3,5,4)$.
9. Отговорът за него е $c = 4, p = 0$.
10. И тук играта свършва след 4 предположения на кодоразбивача.

Това е един кръг от играта. За следващия кръг кодосъставителят създава нов код, който кодоразбивачът отново трябва да разбие. Една игра (с фиксирани n и m) се играе в няколко кръга.

Важно е да отбележим, че кодосъставителят не е длъжен да е изцяло честен в своята игра, т.е. тайният код може да не е предварително съставен. Кодосъставителят може да променя кода по време на кръга, за да направи играта по-трудна. Неговите отговори не трябва да противоречат на кода. С други думи, връщайки се към началото на играта, всички отговори трябва да съответстват на един и същ таен код.

Задача

Вашата задача е да е реализирате този *върховен противник*. Оценяващата система ще бъде кодоразбивачът, докато Вашата програма ще изпълнява ролята на кодосъставител. Системата ще извиква Вашата процедура Init с два параметъра n и m. Гарантирано е, че тази функция се извиква само веднъж в началото на изпълнението на Вашата програма.

След това Вашата процедура NewRound ще бъде извиквана без параметри. Това дава началото на нов кръг и може да се изпълни много пъти по време на една игра, но само след извикването на Init или след като оценяващата система е познала Вашия код.

След началото на нов кръг, Вашата програма (кодосъставителят) трябва да дава отговори на оценяващата система (кодоразбивач). Това се осъществява от оценяващата система, която извиква Вашата процедура Hint, като и предава три параметъра. Първият от тях е масив от n цели числа, всяко в интервала от 0 до $m-1$. Този масив е предположението на кодоразбивача. Вторият и третият параметри са c и p, които се предават по място (Pascal) или като указатели (C/C++). Програмата Ви трябва да върне броя на наличните цифри на правилни позиции като c и броя на наличните цифри на грешни позиции като p, както е описано по-горе. Отговорите трябва да бъдат непротиворечиви!

Всеки кръг завършва, когато Hint върне стойност n на параметъра c и стойност 0 на параметъра p, което означава, че кодоразбивачът е разгадал кода. Играта свършва, когато кодоразбивачът престане да извиква NewRound и прекрати програмата.

Оценяване

По време на едно изпълнение на Вашата програма (т.е. по време на една игра) броят на предположенията, необходими на оценяващата система да разбие вашия код ще бъде записван. Най-малкото от тези числа ще бъде използвано за определяне на броя точки, които ще получите за съответния тест. Нека отбележим това число с t. Ще има само един тест за всяка подзадача, така че това ще бъдат точките за цялата подзадача. За всяка подзадача е определено число k. Ако $t = 1$, то ще получите 0 точки. Ако $t = 2$, то ще получите 7% от точките за подзадачата. Ако $t \geq k$, ще получите всички точки за подзадачата. Всяко число между $t = 2$ и $t = k$ се оценява линейно между 7% и 100% от точките.

Подзадачи

Подзадача	Точки	
1	12	$n = 2, n = 4, k = 4$
2	21	$n = 3, m = 10, k = 7$
3	23	$n = 4, m = 6, k = 5$
4	21	$n = 5, m = 5, k = 5$
5	23	$n = 6, m = 3, k = 5$

Детайли по реализацията

Можете да свалите подготвените среди от сървъра на BOI (`mastermind_c.zip`, `mastermind_cpp.zip` or `mastermind_pas.zip`) с основните файлове за C, C++ или Pascal.

Ако използвате **C** или **C++**, ще трябва да напишете фунцкии със следните прототипи:

```
void Init(int n, int m);
void NewRound();
void Hint(int guess[], int *c, int *p);
```

във файла `mastermind.[c/cpp]`.

Ако използвате **Pascal**, ще трябва да напишете функции със следните интерфейси:

```
interface
procedure Init(n : LongInt; m : LongInt);
procedure NewRound();
procedure Hint(guess : Array of LongInt; var c:LongInt; var p:LongInt);
```

във файла `mastemind.pas`. Масивите са индексирани от 1.

Примерен грейдър

Примерният грейдър ще изисква да въведете предположенията от стандартния вход и ще дава отговорите от Вашата реализация на екрана. Използвайте feedback-а от оценяващата система, за да видите как се представя Вашата програма срещу истинския кодоразбивач.

First Day, Task 2
Old (2 sec, 64MB)

Стар компютър (old)

Годината е 2341. През последните няколко века човечеството експоненциално развива всички аспекти на науката.

Например, по време на Великата Математическа Конференция беше обявено, че транзитивното свойство на равенството е вече отживелица. И така, от a=b и b=c вече не следва, че a=c. Шокиращо, нали?

Тъй като сте намерили стария компютър на Вашия пра^3 дядо, се чудите дали е възможно да решите някои от днешните тривиални задачи на него. След като вече сте решили задачата за Най-дълга обща подредица[1] (LCS) за големи низове на Ядрения Квантов Компютър 21 за O(1), вече се чудите дали е възможно да решите същата задача на стария компютър, отчитайки че може би няма да е възможно за O(1). Понеже информацията в бъдещето се представя много компактно, Вие разполагате само с индексите, за които низовете съвпадат, но не и с целите низове.

Задача

Вашата задача е да създадете функция LcsLength, която има 5 параметъра. Първите два са цели числа n_1 и n_2 – брой на символите в първия и втория низ съответно. Третият е цяло число m – брой на двойките индекси, където двата низа съвпадат. Последните два параметъра са масиви a и b, всеки с по m елемента. За всяко i, $(0 \leq i < m)$ символът с индекс a_i, $(1 \leq a_i \leq n_1)$ от първия низ и символът с индекс b_i $(1 \leq b_i \leq n_2)$ от втория низ съвпадат.

Функцията трябва да връща дължината на най-дългата обща подредица, която удовлетворява описаните изисквания. **Имайте предвид, че транзитивното свойство на равенството вече е в миналото!**

Пример

$$\text{LcsLength}(5,5,5,(1,2,3,4,5),(2,1,3,5,5)) = 3$$

Например, низовете "ABCDD" и "BACID" отговарят на изискванията и "ACD" е една от възможните най-дълги общи подредици на двата дадени низа.

Забележете, че първият низ също би могъл да бъде и "ABCDE" като D=D и E=D (символично), понеже транзитивността липсва. Е, не съдете мъдростта на бъдещето. Ако пък вторият низ е "BACDD" (а първият отново е "ABCDD"), LCS няма да бъде "ACDD" защото без транзитивност не е сигурно, че всички D-та са равни.

[1] Подредица е редица, която може да бъде получена от друга редица, чрез изтриване на някои елементи, без промяна на реда на останалите елементи. Например, редицата (A, B, D) е подредица на (A, B, C, D, E, F).

Подзадачи

Подзадача	Точки	
1	15	$n_1, n_2 \leq 20$, $m \leq 20$
2	15	$n_1, n_2 \leq 2{,}000$, $m \leq 100{,}000$
3	10	$n_1, n_2 \leq 2{,}000$, $m \leq 20{,}000{,}000$
4	30	$n_1, n_2 \leq 100{,}000$, $m \leq 50{,}000$
5	30	$n_1, n_2 \leq 50{,}000{,}000$, $m \leq 100{,}000$

Детайли по реализацията

Можете да свалите подготвените среди от сървъра на BOI (`old_c.zip`, `old_cpp.zip` или `old_pas.zip`) с основните файлове за C, C++ или Pascal.

Ако използвате **C** или **C++**, ще трябва да напишете фунцкия със следния прототип:

```
int LcsLength(int n1, int n2, int m, int a[], int b[]);
```

във файла `old.[c/cpp]`.

Ако използвате **Pascal**, ще трябва да напишете функция със следния интерфейс:

```
interface
function LcsLength(n1 : LongInt; n2 : LongInt; m : LongInt;
                   var a : Array of LongInt;
                   var b : Array of LongInt) : LongInt;
```

във файла `old.pas`. Масивите са индексирани от 0.

Примерен грейдър

Примерният грейдър чете данните от стандартния вход. Първият ред съдържа целите числа n_1 и n_2. На втория ред стои цялото число m. Всеки от следващите m реда съдържа двойка цели числа a_i и b_i.

За горния пример, данните от стандартния вход ще бъдат:

```
5 5
5
1 2
2 1
3 3
4 5
5 5
```

First Day, Task 3
Pentagon (2 sec, 64MB)

Почивна резиденция (pentagon)

Годината е 3013. Тъй като всичкият лед на земята се стопи, Босна и Херцеговина е под вода. Животът е изместен на платформи, които имат формата на правилни петоъгълници[1]. Всеки петоъгълник има до 5 съседни петоъгълника, всеки от които е близо до една от страните му. Не се мъчете с геометрията тук – всичко е в бъдещето и всякакво разположение на петоъгълниците е възможно. Всеки петоъгълник има един или повече полумоста, които са насочени към някой от съседните петоъгълници. Полумостовете се използват за преминаване на съседен петоъгълник. Но, за да стане това, съседният петоъгълник също трябва да има полумост, насочен към първия петоъгълник.

Макар че е възможно да няма полумост към всеки съсед, все пак преминаване може да се осъществи, тъй като петоъгълниците могат да се въртят по посока на часовниковата стрелка. Когато петоъгълник се върти по посока на часовниковата стрелка, всеки полумост се върти заедно с него. В резултат на въртенето полумостът се ориентира към следващия съсед по посока на часовниковата стрелка, или към водата, ако такъв съсед няма. Петоъгълниците имат начална позиция и се въртят по необходимост. Пет последователни завъртания по часовниковата стрелка на един и същ петоъгълник го връщат в начална позиция. Има общо N петоъгълника, номерирани от 1 до N.

Въртенето на петоъгълниците е енергоемка операция, а също и преминаването по мостовете. Затова, те се въртят само когато Президентът иска да достигне до своята почивна резиденция, разположена на петоъгълника с номер N. Президентът винаги тръгва от своя офис, намиращ се на петоъгълника с номер 1.

След многогодишни изследвания, босненските учени са успели да намалят използваната енергия, така че да е необходимо едно и също количество енергия (наричано единица енергия) за следните три действия:

1. единично завъртане по посока на часовниковата стрелка на някой от петоъгълниците;
2. преминаването на президентското возило през мост[2] между два петоъгълника;
3. единично завъртане по посока на часовниковата стрелка на някой от петоъгълниците, последвано от преминаване;

Не е възможно да се направят две последователни ротации или две последователни преминавания, както и преминаване, последвано от ротация, използвайки само една единица енергия.

Петоъгълниците остават завъртени, докато президентът преминава през тях и веднага след напускането на президентското возило се връщат в началната си позиция, без да използват енергия.

[1] Многоъгълник с пет равни страни.

[2] Мост между два петоъгълника се състои от два полумоста, които са правилно ориентирани, т.е. полумостът от първия петоъгълник сочи втория петоъгълник и полумостът от втория петоъгълник сочи първия петоъгълник.

Задача

Вашата задача е да създадете функция MinimalSteps, която има два параметъра. Първият е цяло число N – броят на петоъгълниците. Вторият е матрица P:

$$\begin{pmatrix} p_{0,0} & p_{0,1} & p_{0,2} & p_{0,3} & p_{0,4} \\ p_{1,0} & p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{i,0} & p_{i,1} & p_{i,2} & p_{i,3} & p_{i,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{n-1,0} & p_{n-1,1} & p_{n-1,2} & p_{n-1,3} & p_{n-1,4} \end{pmatrix}$$

където всеки ред $i, (0 \leq i < N)$ описва реда на петте страни по посока на часовниковата стрелка[3] $j, (0 \leq j < 5)$ на петоъгълника $i + 1$, със следната интерпретация:

- ако петоъгълникът $i + 1$ няма нито съсед, нито полумост от страна j, тогава $p_{ij} = 0$;
- ако петоъгълникът $i + 1$ няма съсед от страна j, но има полумост от тази страна, тогава $p_{ij} = 10000$;
- ако петоъгълникът $i + 1$ има съсед k от страна j, без полумост, насочен към него, то $p_{ij} = k$;
- ако петоъгълникът $i + 1$ има съсед k от страна j, и полумост към него, то $p_{ij} = 10000 + k$.

Функцията връща най-малкия брой използвани единици енергия, така че Президентът да достигне до заслужената си почивка, преминавайки от петоъгълник с номер 1 до петоъгълник с номер N. Ако не е възможно Президентът да достигне почивната си резиденция, то функцията трябва да върне -1 (негативна енергия).

Пример

$$MinimalSteps\left(4, \begin{pmatrix} 10000 & 0 & 2 & 0 & 0 \\ 10003 & 10001 & 0 & 0 & 0 \\ 4 & 0 & 10000 & 2 & 0 \\ 10000 & 10000 & 10003 & 10000 & 10000 \end{pmatrix}\right)$$

Ето как може да се достигне от петоъгълник 1 до петоъгълник N с използване на 5 единици енергия:

1. Завъртане на петоъгълник 1;
2. Завъртане на петоъгълник 1 и преминаване към петоъгълник 2;
3. Завъртане на петоъгълник 3 и преминаване към петоъгълник 3;
4. Завъртане на петоъгълник 3;
5. Завъртане на петоъгълник 3 и преминаване към петоъгълник 4.

Не е възможно да се достигне резиденцията, използвайки по-малко от 5 единици енергия.

Подзадачи

Подзадача	Точки	
1	14	$N \leq 100$, има полумост на всяка страна на всеки петоъгълник
2	17	$N \leq 100$, всеки петоъгълник има точно два съседа, с изключение на петоъгълниците с номера 1 и N, които имат само един съсед

[3] Забележете, че номерацията на страните е напълно произволна и само редът им е важен, т.е. това, че те са описани по посока на часовниковата стрелка.

3	12	$N \leq 10$
4	23	$N \leq 400$
5	34	$N \leq 5000$

Детайли по реализацията

Можете да свалите подготвените среди от сървъра на BOI (`pentagon_c.zip`, `pentagon_cpp.zip` or `pentagon_pas.zip`) с основните файлове за C, C++ или Pascal.

Ако използвате **C** или **C++**, ще трябва да напишете фунцкия със следния прототип:

```
int MinimalSteps(int n, int pentagons[][5]);
```

във файла `pentagon.[c/cpp]`.

Ако използвате **Pascal**, ще трябва да напишете функция със следния интерфейс:

```
interface
type PentagonsT = Array of Array[0..4] of LongInt;
function MinimalSteps(n : LongInt; var pentagons : PentagonsT) : LongInt;
```

във файла `pentagon.pas`. Забележете, че `pentagons` е `Array of Array`, вместо двумерен `Array`, така че за достъп до неговите елементи използвайте `pentagons[i][j]` вместо `pentagons[i,j]`. Масивите са индексирани от 0.

Примерен грейдър

Примерният грейдър чете данните от стандартния вход. Първият ред съдържа цяло число N – броят на петоъгълниците. Всеки от следващите N реда $(0 \leq i < N)$ съдържа пет цели числа p_{ij}, $(0 \leq j < 5)$ – елементи на матрицата P.

За горния пример, данните от стандартния вход ще бъдат:

```
4
10000 0 2 0 0
10003 10001 0 0 0
4 0 10000 2 0
10000 10000 10003 10000 10000
```