Task Overview Sheet

TASK		WRITING	PYRAMID	FORBIDDEN
Working directory		/home/ioi06/day1/writing	/home/ioi06/day1/pyramid	/home/ioi06/day1/forbidden/
Input file		writing.in	pyramid.in	forbiddenK.in
Output file		writing.out	pyramid.out	forbiddenK.out
Time limit per task		3 seconds	1 second	N/A
Memory limit		32 MB	32 MB	N/A
Compiler options	**C**	`-pipe –O2 -static -lm`		N/A
	C++	`-pipe –include /usr/include/stdlib.h –O2 –static –lm`		N/A
	Pascal	`-O1 -XS`		N/A
Detailed feedback enabled		YES	NO	NO
Maximum total points		100	100	100
Program header comment when using C		/* PROB: writing LANG: C */	/* PROB: pyramid LANG: C */	Not applicable, the output files you submit shall always begin with `#FILE forbidden K` where *K* is the number of the input file being solved
Program header comment when using C++		/* PROB: writing LANG: C++ */	/* PROB: pyramid LANG: C++ */	

Program header comment when using Pascal		{
		PROB: writing
		LANG: PASCAL
		}
			{
			PROB: pyramid
			LANG: PASCAL
			}	

Solution acceptance rule		Example input is solved correctly		Output file is formatted correctly

DECIPHERING THE MAYAN WRITING

Deciphering the Mayan writing has proven to be a harder task than anticipated by the early investigations. After almost two hundred years, very little of it was actually understood. It has been only in the last three decades that real advances have been made.

Mayan writing is based on small drawings known as glyphs which represent sounds. Mayan words are normally written as glyphs put together at various positions.

One of several problems in deciphering Mayan writing arises in the order of reading. When placing several glyphs in order to form a word, Mayan writers sometimes decided the position based more on their own esthetic views than on any particular rule. This leads to the fact that, even though the sound for many glyphs is known, sometimes archaeologists are not sure how to pronounce a written word.

The archaeologists are looking for a special word *W*. They know the glyphs for it, but they don't know all the possible ways of arranging them. Since they knew you were coming to IOI'06, they have asked for your help. They will provide you with the *g* glyphs from *W* and a sequence *S* of all the glyphs (in the order they appear) in the carvings they are studying. Help them by counting the number of possible appearances of the word *W*.

TASK
Write a program that, given the glyphs for *W* and the sequence *S* of glyphs in the carvings, counts the number of possible appearances of *W* in *S;* that is, every sequence of consecutive *g* glyphs in *S* that is a permutation of the glyphs in *W*.

CONSTRAINTS

$1 \le g \le 3\,000$	the number of glyphs in *W*
$g \le \lvert S \rvert \le 3\,000\,000$	where $\lvert S \rvert$ is the number of glyphs in the sequence *S*

INPUT
Your program must read the following data from the file `writing.in`

`writing.in`	DESCRIPTION

`4 11`
`cAda`
`AbrAcadAbRa`
	LINE 1: Contains 2 space-separated integers that represent *g* and $\lvert S \rvert$.
	LINE 2: Contains *g* consecutive characters that represent the glyphs in *W*. Valid characters are 'a'-'z' and 'A'-'Z'; uppercase and lowercase characters are considered different.
	LINE 3: Contains $\lvert S \rvert$ consecutive characters that represent the glyphs in the carvings. Valid characters are 'a'-'z' and 'A'-'Z'; uppercase and lowercase characters are considered different.

OUTPUT
Your program must write the following data to the file `writing.out`

`writing.out`	DESCRIPTION
2	**LINE 1:** Must contain the count of possible appearances of *W* in *S*.

GRADING
For a set of test cases worth a total of 50 points, each test run will meet the requirement that $g \le 10$.

IMPORTANT NOTE FOR PASCAL PROGRAMMERS
By default in FreePascal, a variable of type **`string`** has a size limit of 255 characters. If you want to use strings longer than that, you should add the directive {$H+} to your code just below the **`program ...;`** line.

PYRAMID

After winning a great battle, King Jaguar wants to build a pyramid that will serve both as a monument to remember his victory and as a tomb for the brave soldiers that died in battle. The pyramid will be built in the battlefield and will have a rectangular base of *a* columns by *b* rows. Inside it, at ground level, is a smaller, rectangular chamber of *c* columns by *d* rows that will contain the corpses and weapons of the fallen soldiers.

The King's architects have surveyed the battlefield as an *m* columns by *n* rows grid and have measured the elevation of each square as an integer.

Both the pyramid and the chamber are to be built covering complete squares of the grid and with their sides parallel to those of the battlefield. The elevation of the squares of the internal chamber must remain unchanged but the remaining terrain of the base of pyramid will be leveled by moving sand from higher squares to lower ones. The final elevation of the base will be the average elevation of all the squares of the base (excluding those of the chamber). The architects are free to locate the internal chamber anywhere within the pyramid as long as they leave a wall at least one square thick surrounding the chamber.

Help the architects pick the best place to locate the pyramid and the internal chamber so that the final elevation of the base is the maximum possible for the sizes given.

The figure shows an example of the battlefield; the number in each square represents the elevation of the terrain in that particular position of the field. The gray squares represent the base of the pyramid while the surrounded white squares represent the chamber. This figure illustrates an optimal placement.

	1	2	3	4	5	6	7	8
1	1	5	10	3	7	1	2	5
2	6	12	4	4	3	3	1	5
3	2	4	3	1	6	6	19	8
4	1	1	1	3	4	2	4	5
5	6	6	3	3	3	2	2	2

TASK
Write a program that, given the dimensions of the field, the pyramid, and the chamber along with the elevation of every square in the field, locates both the pyramid in the field and the chamber inside the pyramid so that the elevation of the base is the maximum possible.

CONSTRAINTS
$3 \le m \le 1000$
$3 \le n \le 1000$
$3 \le a \le m$
$3 \le b \le n$
$1 \le c \le a - 2$
$1 \le d \le b - 2$
All elevations are integers in the range from 1 to 100.

INPUT

Your program must read the following data from the file the `pyramid.in`

`pyramid.in`	**DESCRIPTION**

`8 5 5 3 2 1`
`1 5 10 3 7 1 2 5`
`6 12 4 4 3 3 1 5`
`2 4 3 1 6 6 19 8`
`1 1 1 3 4 2 4 5`
`6 6 3 3 3 2 2 2`
	LINE 1: Contains six space-separated integers, respectively: *m*, *n*, *a*, *b*, *c*, and *d*.
	NEXT *n* LINES: Each line contains *m* space-separated integers that represent the elevations of one row of the grid. The first of these lines represents the top row (row 1) of the grid, and the last line represents the bottom row (row *n*). The *m* integers in each line represent the elevations of squares of that row starting from column 1.

OUTPUT

Your program must write the following data to the file the `pyramid.out`

`pyramid.out`	**DESCRIPTION**
`4 1` `6 2`	**LINE 1:** Must contain 2 space-separated integers that represent the upper-left corner of the base of the pyramid, the first number being the column and the second the row. **LINE 2:** Must contain 2 space-separated integers that represent the upper-left corner of the chamber inside the pyramid, the first number being the column and the second the row.

NOTE: If there are multiple optimal placements, then any one of them you output will be considered correct.

GRADING

For a number of test cases worth a total of 30 points, every test run will meet the following requirements:

$3 \le m \le 10$

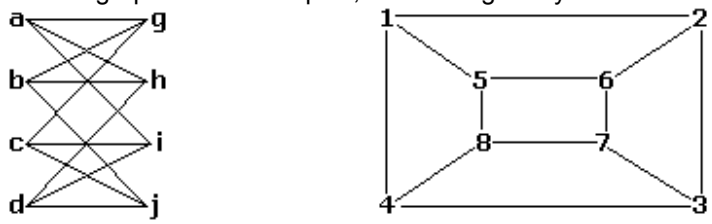
$3 \le n \le 10$

FORBIDDEN SUBGRAPH

Two undirected graphs *G* and *H* are said to be *isomorphic* if:

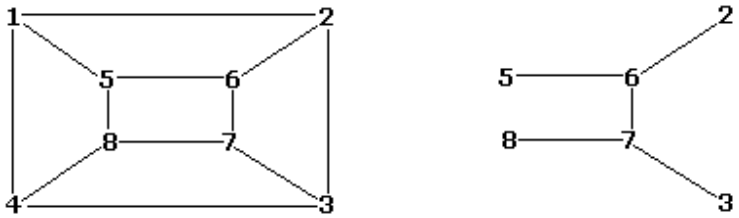
- they have the same number of vertices and
- a one-to-one correspondence exists between their vertices so that, for any two distinct vertices of *G*, there exists an edge between them if and only if there exists an edge between their corresponding vertices in *H*.

For example, the next two graphs are isomorphic, even though they look different here:

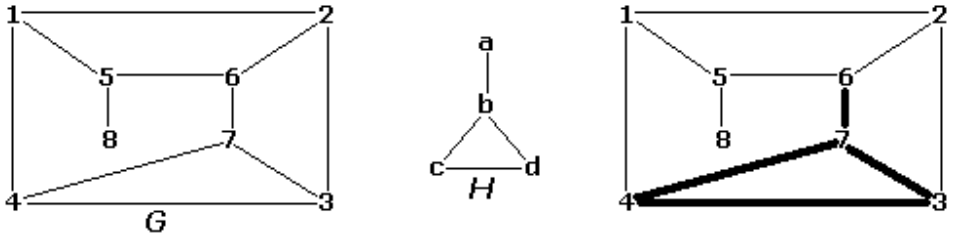


A possible one-to-one correspondence showing that these two graphs are isomorphic is given by {a-1, b-6, c-8, d-3, g-5, h-2, i-4, j-7}, but others exist too.

A *subgraph* of a graph *G* is a graph whose sets of vertices and edges are subsets of those in *G*. Note that *G* is a subgraph of itself. The following example shows a graph and one of its subgraphs:



We say that a graph *G contains* another graph *H* if there is at least one subgraph *H'* of *G* which is isomorphic to *H*. The following figure shows a graph *G* that contains the graph *H.*



TASK

Given two undirected graphs *G* and *H,* produce a subgraph *G'* of *G* such that:

- the number of vertices in *G* and *G'* is the same and
- *H is* ***not*** *contained in G'*.

Naturally, there may be many subgraphs G' with the above properties. Produce one of those subgraphs with as many edges as possible.

BASE ALGORITHM
Perhaps the most basic strategy to approach this problem is to consider the edges of *G* in the order that they are represented in the input file, then attempting to add them one by one to *G'*, verifying at each step whether *H* is contained in *G'* or not. The correct implementation of this greedy algorithm will earn some points, but much better strategies exist.

CONSTRAINTS
$3 \le m \le 4$ The number of vertices of *H.*
$3 \le n \le 1000$ The number of vertices of *G.*

INPUT
You will be given 10 files `forbidden1.in` to `forbidden10.in` each with the following data:

`forbiddenK.in`	**DESCRIPTION**

`3 5`
`0 1 0`
`1 0 1`
`0 1 0`
`0 1 0 0 0`
`1 0 1 0 0`
`0 1 0 1 0`
`0 0 1 0 1`
`0 0 0 1 0`
	LINE 1: Contains two space-separated integers, respectively: *m* and *n*.
	NEXT m LINES: Each line contains *m* space-separated integers and represents one vertex of *H* in the order 1, ..., *m*. The *i*-th element of the *j*-th line in this section is equal to 1 if vertices *i* and *j* are joined by an edge in *H* and is equal to 0 otherwise.
	NEXT n LINES: Each line contains *n* space-separated integers and represents one vertex of *G* in the order 1, ..., *n*. The *i*-th element of the *j*-th line in this section is equal to 1 if vertices *i* and *j* are joined by an edge in *G* and is equal to 0 otherwise.

Observe that, except for line 1, the above input represents the adjacency matrices of *H* and *G*.

OUTPUT
You must provide 10 files, one for each of the inputs. Each file must contain the following data:

`forbiddenK.out`	**DESCRIPTION**

`#FILE forbidden K`
`5`
`0 1 0 0 0`
`1 0 0 0 0`
`0 0 0 0 0`
`0 0 0 0 0`
`0 0 0 0 0`
	LINE 1: The file header. The file header must contain **`#FILE forbidden K`** where K is a number between 1 and 10 that corresponds to the input file solved.
	LINE 2: Contains one integer: *n*.
	NEXT n LINES: Each line contains *n* space-separated integers and represents one vertex of *G'* in the order 1, ..., *n*. The *i*-th element of the *j*-th line in this section is equal to 1 if vertices *i* and *j* are joined by an edge in *G'*, and is 0 otherwise.

Observe that, except for lines 1 and 2, the above output represents the adjacency matrix of *G'.* Note that there are many possible outputs, and that the above output is correct but not optimal.

GRADING
Your score will depend on the number of edges in the *G'* you output. Your score will be determined in the following way: you will receive a non-zero score for each output file only if it meets the task specification. If it does, your score will be calculated as follows. Let E_y be the number of edges in your output, let E_b be the number of edges in *G'* as computed by the BASE ALGORITHM, and let E_m be the maximum number of edges in the output of any of the contestants submissions. Your score for the case will be:

- $30\ \boldsymbol{E_y / E_b}$ if $\boldsymbol{E_y \le E_b}$, or
- $30 + 70(\boldsymbol{E_y - E_b})/(\boldsymbol{E_m - E_b})$ if $\boldsymbol{E_y > E_b}$.