Laborator 4

Structuri de date:

struct + array + queue + stack + BT + BST

Categoria A (8 max)

1A	N points are given in R^2. Find the nearest pair of points. Each point is described by its coordinates (X, Y). Use Struct, dynamic memory allocation, ways to make the algorithm more efficient.
2A	**Farmer and tractor** Everyday, Farmer John sells vegetables on bazaar. Each day, in the evening, he puts the daily incomings into an envelope and places it in a secure place. He has a stack of envelopes now. Yesterday, he decided to buy a new tractor, that costs S MDL. So, John will use the money from envelopes in inverse order of their place in the stack. First, he will open the uppermost envelope (containing the today's income money), then the next one (containing the yesterday incomings) and so on, till the sum S will be collected. Write a program to simulate the process of collecting and spending money in John’s secure stack.
3A	Write a program which displays the truth table for a given logical function, in a table with borders. Columns will be aligned right. Operate with functions containing no more than 5 logical variables.
4A	Write a program to manage binary search trees. The program will have a simple user interface to select a required operation : add node, search node, tree traversal, remove node.

5 A-	Write a program to manage a queue / stack. The program will have a simple user interface to select a required operation: add node, search node, view queue / stack, remove node from queue / stack.
6A	Write a program to manage an arbitrary linked list. The program will have a simple user interface to select a required operation: add node, search node, view list, remove node from the list by value.

Category B (9 max)

1B	Write a program which displays the truth table for an arbitrary logical function in a table with borders. Columns will be aligned right. Operate with functions containing no more than 10 logical variables. The function will be described in a string, read from standard input.

2B	Compania MoldPetrol se ocupă de căutarea zăcămintelor de petrol. MoldPetrol lucrează de obicei pe o regiune dreptunghiulară, creând o rețea de sonde, care divide regiunea în numeroase sectoare pătrate. Pentru fiecare sondă se obțin rezultatele – se conține în esa petrol sau nu. Sonda cu petrol se numeşte pungă. Dacă două pungi sunt adiacente (orizontal, vertical sau diagonal), ele fac parte din acelaşi zăcământ de petrol. Zăcămintele pot fi foarte mari şi conține un număr mare de pungi. Sarcina este de a determina câte zăcăminte distincte(separate) sunt în regiunea cercetată.
	
	Fişierul input va conține una sau mai multe regiuni. fiecare regiune începe cu o linie, care conține 2 întregi - m şi n, - numărul de rânduri şi coloane in regiune, separate prin un spațiu. Dacă m = 0 – e sfârşitul inputului. În celelalte cazuri 1 < m < 100 şi 1 < n < 100. Urmează m linii a câte n caractere fiecare. Fiecare caracter corespunde unei sonde, dacă e '*',- sonda nu dispune de petrol, '@', - reprezintă o pungă de petrol.
	
	Pentru fiecare regiune dați numărul de zăcăminte de petrol separate.
	
	Exemplu input:
	
	`1 1`
	`*`
	`3 5`

```
*@*@*
**@**
*@*@*
1 8
@@****@*
5 5
****@
*@@*@
*@**@
@@@*@
@@**@
0 0

Exemplu output:

0
1
2
2
```

The company MoldPetrol is concerned with digging out petroleum deposits. Usually, MoldPetrol covers a rectangular area, establishing a network of petroleum rigs, which split the area into numerous square areas. For each rig, feedback is returned according to whether the rig contains or doesn't contain petroleum. Rigs that contain oil are called "pockets". If two pockets are adjacent (horizontally, vertically or diagonally), they are considered to belong to the same petroleum deposit. Deposits can be huge, containing lots of pockets. Your task is to determine how many distinct (separate) deposits the researched area contains.

The input file will consist of one or several areas. Each of them starts with a line, which contains 2 integer numbers - m & n - the number of rows and columns in the assigned area, separated through space. If m=0 - this is considered the end of the input. In other cases: $1 < m < 100$ and $1 < n < 100$. M lines containing n characters each will follow. Each character is linked to a rig: if it's '*' - the rig does not contain petroleum, while '@' marks a rig with petroleum.

Assign the number of individual petroleum deposits to each area.

Example of input:

```
1 1

*
3 5
*@*@*
**@**
*@*@*
1 8
```

```
@@****@*
5 5
****@
*@@*@
*@**@
@@@*@
@@**@
0 0
```

Example of output:

```
0
1
2
2
```

3B

Considerăm un şir de N numere naturale distincte $a_1, a_2, \ldots, a_N$. Pentru fiecare termen a_i definim predecesorul său, dacă există, ca fiind cel mai din dreapta termen a_j, cu $j < i$ şi $a_j < a_i$. De exemplu, pentru şirul 8, 12, 2, 4, 3, 10, 9, 7, 5, 6, numărul 8 este predecesorul lui 12, 3 este predecesorul lui 5, 10 nu este predecesorul niciunui număr, la fel 4 nu este predecesor pentru alt număr, 2 nu are predecesor.

Cerinţă

Scrieţi un program care determină câte numere din şir nu sunt predecesori ai niciunui alt număr.

Date de intrare

Fişierul de intrare predecesor.in conţine pe prima linie numărul natural N reprezentând numărul de termeni ai şirului. Pe următoarea linie se găsesc, separaţi prin câte un spaţiu, termenii $a_1, a_2, \ldots, a_N$ ai şirului.

Date de ieşire

Fişierul de ieşire predecesor.out va conţine o singura linie pe care va fi scris un singur număr natural reprezentând numărul de termeni ai şirului care nu sunt predecesori ai niciunui alt număr.

Restricţii

- 3 <= N <= 500 000
- 1 <= a_i <= 1 000 000 000, pentru orice 1 <= i <= N

Pred.in	Pred.out
10 8 12 2 4 3 10 9 7 5 6	6

Consider a series of N distinct natural numbers a_1, a_2, …, a_N. For each element a_i determine its predecessor, if another element a_j exists, being the rightmost one, with $j < i$ and $a_j < a_i$. For example, for the series 8, 12, 2, 4, 3, 10, 9, 7, 5, 6. Here, 8 is the predecessor of 12; 3 is the predecessor of 5; 10 is **not** the predecessor of any number, so is 4; 2 does not have any predecessors.

Task
Write a program which determines how many numbers in the series are not predecessors to any other numbers.

<table>
<tr><td></td><td>

Input data
The input file *predecessor.in* contains a natural number N on the first line, which would represent the number of elements in the series. The next line contains the elements a_1, a_2, …, a_N of the series, separated through space.

Output data
The output file *predecessor.out* contains a single line, on which one natural number N is written, representing how many numbers in the series are not predecessors to any other numbers.

Restrictions
$3 \le N \le 500\,000$
$1 \le a_i \le 1000\,000\,000$, for any $1 \le i \le N$

<table>
<tr><th>Pred.in</th><th>Pred.out</th></tr>
<tr><td>10
8 12 2 4 3 10 9 7 5 6</td><td>6</td></tr>
</table>

</td></tr>
<tr><td>4B</td><td>Elaboraţi un algoritm pentru generarea formelor normale disjunctive (conjunctive) perfecte ale funcţiilor logice. Funcţia este descrisă direct în program. Max. 5 variabile independente.

Design an algorithm which generates normal, perfect disjunctive (conjunctive) forms of the logic functions. The function is described directly by the program. Use 5 independent variables at most.</td></tr>
<tr><td>5B</td><td>Write a program to manage binary search trees. The program will have a simple user interface to select a required operation : add node, search node, tree traversal, remove node.

The program will have the option to draw tree by levels and connections in text mode, using ASCII symbols. Do not use additional libraries, except yours!</td></tr>
</table>

Category C (10 max)

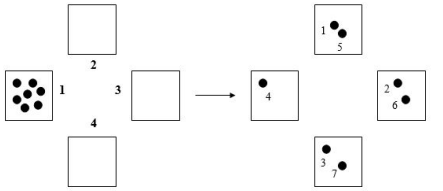
1C

ANTIKALAH

[Enunț]

Pentru jocul kalah se folosesc câteva cutii, amplasate în cerc, în care se află bile. O mișcare este realizată în felul următor: se iau toate bilele dintr-o cutie și se repartizează consecutiv câte una, în cutii, începând cu următoarea cutie în direcția mișcării acelor de ceasornic. Dacă numărul bilelor este mai mare decât cel al cutiilor, repartizarea continuă până la epuizarea bilelor. În acest caz bile se pun și în cutia din care inițial au fost extrase.

Un exemplu de mișcare este prezentat pe desen. În partea dreaptă bilele sunt numerotate în ordinea în care au fost puse în cutii



În timpul unui antrenament Ionuț a amplasat aleatoriu bilele prin cutii, după care a început să facă mișcări. După fiecare mișcare el înscria numărul cutiei în care era pusă ultima bilă. La un moment dat Ionuț a hotărât să restabilească repartizarea inițială a bilelor în cutii, folosind repartizarea finală și notițele făcute la mișcările precedente. Scrieți un program, care îl va ajuta să facă acest lucru.

[Date de intrare]

Prima linie a fișierului de intrare va conține două numere naturale: N ≤ 100 — numărul de cutii și M ≤ 100 — numărul de mișcări, efectuate de Ionuț. Cutiile sunt numerotate consecutiv de la 1 la N în ordinea amplasării lor după acele de ceasornic.

Următoarele N linii conțin câte un număr — cantitatea de bile în prima, a doua, a treia, ... , a N-a cutie în repartizarea finală.

Urmează M linii, care conțin câte un număr, care corespunde indicelui cutiei în care a fost pusă ultima bilă la prima, a doua, a treia, ..., a M -a mișcare.

[Restricții]

Numărul total de bile nu depășește 10^3.

N ≤ 100

M ≤ 100

[Exemplu]

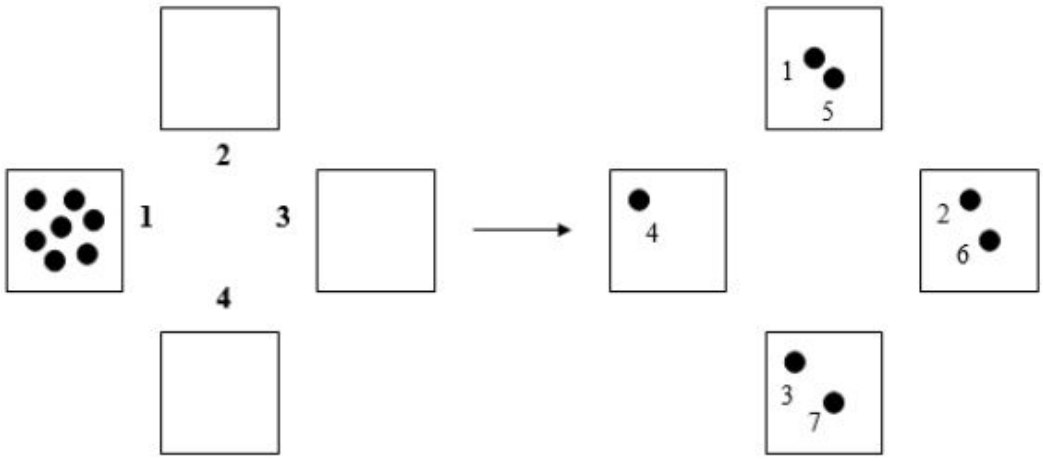
*.in	*.out
4 1 1 2 2 2 4	7 0 0 0

*.in	*.out
2 2 1 1 2 2	1 1

ANTIKALAH

The game Kalah requires a few boxes, placed in a circle, containing pebbles. A move is made in the following way: all pebbles from one of the boxes are taken out and dropped consecutively, one by one, in the boxes, starting from the next box, moving clockwise. If the number of pebbles is bigger than the number of boxes, the repartition continues until no pebbles are left. In this case, pebbles are dropped into the original box, too.

An example of moves is displayed in the picture below. The pebbles from the right side are numbered according to the order they were dropped in the boxes.



During one of his trainings, Ionuț dropped the pebbles in the boxes in random order, after which he started his moves. After each move, he wrote down the number of the box in which the pebble was placed. At one moment, he decided to restore the initial repartition of the pebbles in boxes, by using the last repartition and the notes on the previous moves. Write a program which would help him do that.

Input data:

The first line of the input file will consist of natural numbers: N ≤ 100 - the number of boxes; M ≤ 100 - the number of moves Ionuț made. The boxes are numbered consecutively from 1 to N according to their clockwise layout.

The next N lines will each contain a number that represents the quantity of pebbles in the first, second, third, ... , N^{th} box in the final layout.

M lines that follow will each contain a number that corresponds to the index on the box in which the last pebble was placed during the first, second, third, …, M^{th} move.

Restrictions:

The number of pebbles does not exceed 10^9.
N ≤ 100
M ≤ 100

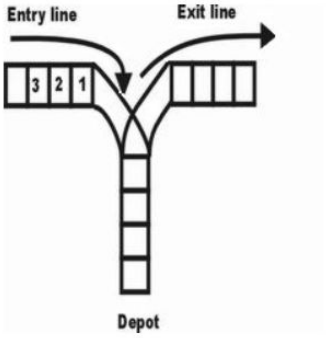
Example:

*.in	*.out
4 1 1 2 2 2 4	7 0 0 0

*.in	*.out
2 2 1 1 2 2	1 1

2C

Sa consideram un depou ca in figura de mai jos. Observati ca exista o linie de intrare, pe care intra `n` vagoane, numerotate de la `1` la `n` in ordinea intrarii.
Vagoanele intra in depou, urmand apoi sa iasa intr-o ordine oarecare pe linia de iesire.



Cerinta

Dat fiind numarul de vagoane sa se determine numarul de modalitati distincte de aranjare a vagoanelor pe linia de iesire.

Date de intrare

Fisierul de intrare `depou.in` contine o singura linie pe care este scris un numar natural `n` reprezentand numarul de vagoane.

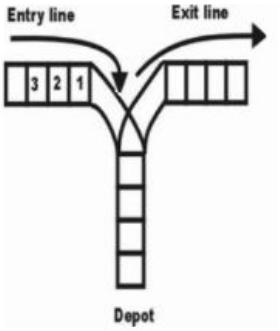
Date de iesire

Fisierul de iesire `depou.out` va contine o singura linie pe care va fi scris un numar natural reprezentand numarul de modalitati distincte de aranjare a vagoanelor pe linia de iesire.

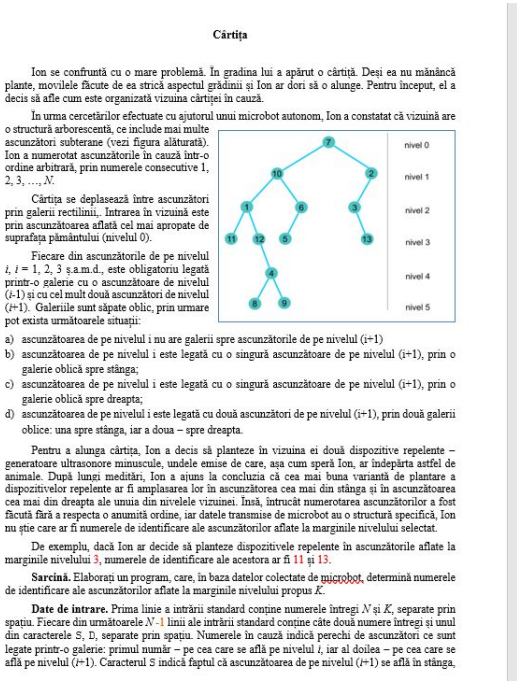
Restrictii

- `1 <= n <= 150`

Exemplu input
3

	Output 5 Consider a depot similar to the one drawn below. Notice that there is an entrance line, which is used by the wagons to enter the depot. The wagons are numbered from 1 to n according to the order they came in. The wagons enter the depot, then exit through the designated line according to a specific order.  **Task** Given the number of wagons, determine in how many distinct ways we can arrange the wagons in the exit line. **Input data** The input file *depot.in* (*depou.in*) contains a single line on which a natural number n (representing the number of wagons) is written. **Output data** The output file *depot.out* (*depou.out*) on which a natural number (representing the number of distinct ways we can arrange the wagons in the exit line) is written. **Restrictions** 1 ≤ n ≤ 150 **Example of input** 3 **Example of output** 5
3C	Elaboraţi un algoritm pentru generarea formelor normale disjunctive (conjunctive) perfecte ale funcţiilor logice. Funcţia este descrisă într-un şir de caractere introdus din input standard sau fişier. Max. 5 variabile independente Define an algorithm that generates perfect normal disjunctive (conjunctive) forms of logical functions. The function is described by a string from standard input of a file. You may use 5 independent variables at most.

4C



Cărtița

Ion se confruntă cu o mare problemă. În gradina lui a apărut o cârtiță. Deși ea nu mănâncă plante, movilele făcute de ea strică aspectul grădinii și Ion ar dori să o alunge. Pentru început, el a decis să afle cum este organizată vizuina cârtiței în cauză.

În urma cercetărilor efectuate cu ajutorul unui microbot autonom, Ion a constatat că vizuină are o structură arborescentă, ce include mai multe ascunzători subterane (vezi figura alăturată). Ion a numerotat ascunzătorile în cauză într-o ordine arbitrară, prin numerele consecutive 1, 2, 3, …, N.

Cărtița se deplasează între ascunzători prin galerii rectilinii,. Intrarea în vizuină este prin ascunzătoarea aflată cel mai apropate de suprafața pământului (nivelul 0).

Fiecare din ascunzătorile de pe nivelul i, i = 1, 2, 3 ș.a.m.d., este obligatoriu legată printr-o galerie cu o ascunzătoare de nivelul (i-1) și cu cel mult două ascunzători de nivelul (i+1). Galeriile sunt săpate oblic, prin urmare pot exista următoarele situații:

a) ascunzătoarea de pe nivelul i nu are galerii spre ascunzătorile de pe nivelul (i+1)
b) ascunzătoarea de pe nivelul i este legată cu o singură ascunzătoare de pe nivelul (i+1), prin o galerie oblică spre stânga;
c) ascunzătoarea de pe nivelul i este legată cu o singură ascunzătoare de pe nivelul (i+1), prin o galerie oblică spre dreapta;
d) ascunzătoarea de pe nivelul i este legată cu două ascunzători de pe nivelul (i+1), prin două galerii oblice: una spre stânga, iar a doua – spre dreapta.

Pentru a alunga cârtița, Ion a decis să planteze în vizuina ei două dispozitive repelente – generatoare ultrasonore minuscule, undele emise de care, așa cum speră Ion, ar îndepărta astfel de animale. După lungi meditări, Ion a ajuns la concluzia că cea mai buna variantă de plantare a dispozitivelor repelente ar fi amplasarea lor în ascunzătorea cea mai din stânga și în ascunzătoarea cea mai din dreapta ale unuia din nivelele vizuinei. Însă, întrucât numerotarea ascunzătorilor a fost făcută fără a respecta o anumită ordine, iar datele transmise de microbot au o structură specifică, Ion nu știe care ar fi numerele de identificare ale ascunzătorilor aflate la marginile nivelului selectat.

De exemplu, dacă Ion ar decide să planteze dispozitivele repelente în ascunzătorile aflate la marginile nivelului 3, numerele de identificare ale acestora ar fi 11 și 13.

Sarcină. Elaborați un program, care, în baza datelor colectate de microbot, determină numerele de identificare ale ascunzătorilor aflate la marginile nivelului propus K.

Date de intrare. Prima linie a intrării standard conține numerele întregi N și K, separate prin spațiu. Fiecare din următoarele N-1 linii ale intrării standard conține câte două numere întregi și unul din caracterele S, D, separate prin spațiu. Numerele în cauză indică perechi de ascunzători ce sunt legate printr-o galerie: primul număr – pe cea care se află pe nivelul i, iar al doilea – pe cea care se află pe nivelul (i+1). Caracterul S indică faptul că ascunzătoarea de pe nivelul (i+1) se află în stânga, iar caracterul D – că ascunzătoarea în cauză se află în dreapta în raport cu ascunzătoarea de pe nivelul i.

Date de ieșire. Ieșirea standard va conține pe o singură linie două numere întregi separate prin spațiu – numărul de identificare a ascunzătorii din stânga și numărul de identificare a ascunzătorii din dreapta ale nivelului K.

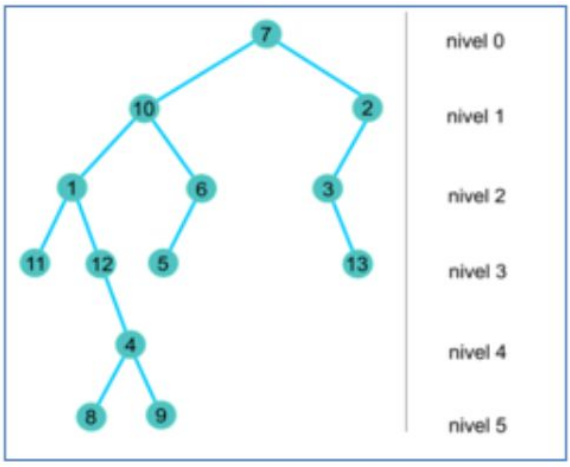
Restricții. $3 \le N \le 1000000$; $0 \le K \le 500$. Timpul de execuție nu va depăși 0,5 secunde. Programul va folosi cel mult 20 Megaocteți de memorie operativă. Fișierul sursă va avea denumirea cartita.pas, cartita.c sau cartita.cpp.

Exemplu.

Intrare	*Ieșire*
13 3 4 8 S 10 6 D 7 2 D 4 9 D 1 12 D 6 5 S 12 4 D 2 3 S 3 13 D 1 11 S 10 1 S 7 10 S	11 13

Ion struggles with a big problem. A mole settled in his garden. Although it doesn't eat plants, the molehills affects the aspect of the garden, so Ion would like to get rid of it. Firstly, he decided to find out about how the mole's burrow is structured.

His research was done by using an autonomous microbot. The obtained data helped him conclude that the burrow has a tree-like structure, containing several underground hiding places (see the image). Ion numbered these places from 1 to N, in arbitrary order.



The mole moves through the hiding pockets through rectilinear tunnels. The entrance to the burrow is through the pocket closest to the surface (level 0).

Each of the hiding pockets pockets level i, i equals 1, 2, 3, and so on, must be linked to an i - 1 level hiding pocket through a tunnel and it can be linked to no more than two i + 1 level hiding pockets. The tunnels are diagonally dug. Thus, the following situations are possible:

a) the hiding pocket on level i does not have a tunnel to link it to the hiding pocket on the level i + 1;
b) The hiding pocket on level i is linked to a single hiding pocket on the level i + 1 through a diagonal tunnel directed to the left;
c) the hiding pocket on the level i is linked to a single hiding pocket on the level i + 1 through a diagonal tunnel directed to the right;
d) the hiding pocket on the level II is linked to chew hiding pocket on the level II plus one through two diagonal tunnels: one going to the left, one to the right.

In order to get rid of the mole, Ion decided to plant 2 repellent devices: I need ultrasound of generators, the waves of which, as Ion hopes, would help him to get rid of the pests. After thinking for a long time he decided that the best way to plant the repellent devices would be to set them in the leftmost hiding pocket and also to the rightmost one from the level of the burrow. However, because the numbering of the hiding Pockets was done arbitrary, whilst the microbot sends data in a specific structured way, Ion does not know the number of the hiding pockets that would help him identify the ones from the edges of the selected hiding pocket.

For instance, if Ion decides to plant the repellent devices in the pockets on the edges of the level 3, the identification numbers corresponding to them would be 11 and 13.

Task

Write a program which determines the identification numbers of the hiding pockets in the specified level K, using the data collected by the microbot.

Input data

The first line of the standard input contains the integer numbers N and K, separated through space. Each of the following N-1 lines of the standard input will each contain 2 integer numbers and one of the characters S, D, separated through space. These numbers indicate couples of hiding pockets that are linked through tunnels: the first number indicates the one on the level i, the second one - the pocket on the level i+1. “S” indicates that the pocket on the level i+1 is located on the left, while “D” indicates that the pocket is located on the right of the level i.

Output data

The standard output will consist of a single line, containing 2 integer numbers separated through space - the identification number of the hiding pockets from the left and the identification number of the hiding pocket on the right of the level K.

Restrictions

$3 \leq N \leq 1000000$; $0 \leq K \leq 500$. The execution time should not exceed 0.5 seconds. The program should use 20 Megaoctets operative memory at most. The source file should be named *cartita.pas*, *cartita.c* or *cartita.cpp*.

	Example **Input** 13 3 4 8 S 10 6 D 7 2 D 4 9 D 1 12 D 6 5 S 12 4 D 2 3 S 3 13 D 1 11 S 10 1 S 7 10 S **Output** 11 13
5C	