```
   }
   cout << endl;
   for (i=0; i<=n-1; i++)
     cout << v[i] << " ";
   // sorteaza partea nearanjata cu elementele impare
   if (capat>0)
   {
     for (i=capat; i<=n-2; i++)
      for (j=i+1; j<=n-1; j++)
       if (v[j]>v[i])
       {
        temp=v[i]; v[i]=v[j]; v[j]=temp;
       }
   }
   cout << endl;
   for (i=0; i<=n-1; i++)
     cout << v[i] << " ";
}
```

Probleme propuse

Probleme suplimentare

▶ 1. (Bacalaureat iulie 2009, varianta 82 – enunț adaptat)

Scrieți un program care primește de la tastatură un tablou unidimensional cu `n` elemente numere întregi, fiecare element având cel mult patru cifre, și afișează pe ecran produsul elementelor impare din tablou, sau valoarea `0` dacă nu există elemente impare.

Exemplu: pentru tabloul `v=(-3,8,5,1,2,4)` programul va afișa numărul `-15 (-3*5*1)`, iar pentru tabloul `v=(12,0,4,16)` se va tipări valoarea `0`.

▶ 2. (Bacalaureat iulie 2009, varianta 16)

Scrieți un program care citește de la tastatură elementele unui tablou unidimensional cu exact `10` numere naturale, mai mici decât `1000`, determină și afișează pe ecran, separate prin câte un spațiu, numărul de elemente din șir care sunt multipli ai numărului `13` și, în continuare, pozițiile pe care acestea se găsesc în șir. Elementele tabloului sunt numerotate de la `0` la `9`.

Exemplu: dacă șirul citit este (`2, 6, 26, 14, 130, 11, 8, 23, 52, 39`), se vor afișa numerele `4 2 4 8 9`.

Indicații: *Dacă nu s-ar fi cerut să afișăm mai întâi numărul elementelor divizibile cu `13` și abia apoi pozițiile acestora, ar fi fost suficientă o singură parcurgere a vectorului, în care afișam pozițiile elementelor cu pricina și în același timp le număram în variabila `nr`, apoi, după ieșirea din ciclu, afișam `nr`. În contextul cerinței privind forma de afișare, sunt necesare două parcurgeri: cu prima parcurgere facem numărarea, afișăm `nr`, iar apoi cu a doua parcurgere "vizităm" din nou toate elementele `v[i]` (cu `i=0,1,...,n-1`) și afișăm pozițiile celor divizibile cu `13`.*

► **3. (Bacalaureat iulie 2009, varianta 67 – enunț adaptat)**

Scrieți un program care primește de la tastatură un tablou unidimensional cu `n` elemente numere întregi, fiecare element având cel mult nouă cifre, și afișează pe ecran numărul de numere prime din tablou.

Exemplu: pentru `n=5` și tabloul `(12,37,43,6,72)`, se va tipări valoarea 2.

► **4. (Bacalaureat iulie 2009 - varianta 44)**

Scrieți un program care citește de la tastatură numărul natural `n` (`0<n<100`) și un șir format din `n` numere întregi de cel mult `4` cifre fiecare, determină și afișează pe ecran numărul de pătrate perfecte din șir.

Exemplu: dacă `n=6`, iar șirul este format din elementele `(31,25,19,11,4,3)`, atunci pe ecran se va afișa 2.

► **5. (Bacalaureat iulie 2009 - varianta 33 - enunț adaptat)**

Scrieți un program care primește de la tastatură un tablou unidimensional cu `n` elemente numere întregi, și afișează mesajul `DA` în cazul în care printre elementele tabloului `x` se află cel puțin un număr impar, sau afișează mesajul `NU` în caz contrar.

Exemplu: pentru vectorul `v=(2,4,4,8,6,12,12)`, programul va afișa mesajul `NU`.

► **6.** Scrieți un program care verifică dacă cele `n` elemente ale unui șir dat de numere întregi sunt în ordine strict crescătoare de la stânga la dreapta.

Exemplu: Elementele șirului `(2,3,5,8)` sunt în ordine strict crescătoare iar șirul `(2,3,8,5)` nu îndeplinește această cerință.

***Indicații:** Avem de-a face cu o testare logică. Presupunem mai întâi condiția adevărată, inițializând cu `1` o variabilă `ok` (de tip întreg, dar cu sens logic). Apoi, într-un ciclu dirijat de contorul `i`, parcurgem pozițiile elementelor începând cu al doilea, și pentru fiecare element `v[i]` căutăm cazul contrar: dacă `v[i]<=v[i-1]`, înseamnă că la pasul respectiv am găsit un element `v[i]` care "strică" ordinea strict crescătoare, deci `ok` primește valoarea `0` (aferentă "stării" de condiție falsă). În final testăm valoarea cu care `ok` a ieșit din ciclu, și în funcție de aceasta afișăm un mesaj sugestiv.*

► **7.** Fiind dat un vector `v` cu `n` elemente numere întregi, citite de la tastatură, scrieți un program care afișează toate perechile de elemente consecutive egale între ele. Fiecare astfel de pereche va fi scrisă pe un rând de ecran, între două paranteze, cu elementele sale separate prin "virgulă".

Exemplu: Pentru vectorul `v=(2,3,3,8,5,5,11,11,7,9)` cu `n=10` elemente, se vor afișa perechile `(3,3)`, `(5,5)` și `(11,11)`.

► **8.** Se citește de la tastatură un șir de `n` numere întregi. Să se afișeze toate perechile de elemente consecutive cu proprietatea că al doilea element al perechii este egal cu suma cifrelor primului.

Exemplu: Pentru șirul `(124,7,12,44,8,9)` se afișează perechile `(124,7)`, `(44,4)`.

► **9. (Bacalaureat iulie 2009, varianta 10)**

Scrieţi un program care citeşte de la tastatură un număr natural `n` (`n≤100`) şi apoi cele `n` elemente, numere naturale cu cel mult patru cifre fiecare, ale unui tablou unidimensional `a`. Programul determină şi afişează pe prima linie a ecranului suma celor `n` elemente ale tabloului, pe a doua linie a ecranului suma primelor `n-1` elemente şi aşa mai departe, astfel încât pe linia `n-1` se va afişa suma primelor `2` elemente, iar pe linia `n` primul element al tabloului. Poziţiile elementelor tabloului sunt numerotate cu `0,1,...,n-1`.

Exemplu: dacă `n=4`, iar tabloul are elementele `a=(1,2,3,4)` programul va afişa valorile: `10,6,3,1`

Indicaţii: *Practic avem de calculat `n` sume, deci vom proiecta un ciclu în care contorul `p` va lua pe rând valorile `1,2,...,n`, şi la fiecare pas calculăm şi afişăm suma cu numărul de ordine `p`, care este de fapt suma primelor `p` elemente; pentru a efectua această sumă, într-un al doilea ciclu interior primului, parcurgem poziţiile `i` ale elementelor de la `0` la `p-1`, şi la fiecare pas adăugăm elementul `v[i]` la o variabilă `s` (iniţializată cu `0` la fiecare pas al primului ciclu, înainte de declanşarea celui de-al doilea). Putem folosi aceeaşi variabilă `s` la fiecare pas al primului ciclu, deoarece nu se cere salvarea acestor sume, ci doar afişarea fiecăreia dintre ele, imediat ce a fost calculată.*

► **10. (Bacalaureat iulie 2009, varianta 28)**

Scrieţi un program care citeşte de la tastatură un număr natural `n` din intervalul `[2,10000]` şi apoi `n` numere reale şi afişează pe ecran câte dintre cele `n` numere reale sunt egale cu media aritmetică a celorlalte `n-1` numere reale.

Exemplu: pentru şirul `(7,2,3,4)` se va afişa valoarea `1` (pentru că un singur element, şi anume `4`, este media aritmetică a celorlalte), iar pentru şirul `(1,1,1,1,1)` se va afişa valoarea `5`.

► **11. (Bacalaureat iulie 2009, varianta 54)**

Scrieţi un program care citeşte de la tastatură un număr natural nenul `n` (`n<=100`) şi apoi cele `n` numere naturale nenule, de maximum patru cifre, reprezentând elementele unui tablou unidimensional `v` (cu indici de la `0` la `n-1`) şi afişează câte dintre elementele v_i (`2≤i≤n-1`) sunt egale cu suma celor două elemente vecine. În cazul în care nu există niciun astfel de element în tabloul `v`, se va afişa valoarea `0`.

Exemplu: dacă `n=7` şi tabloul unidimensional `v` are conţinutul `(10,25,15,45,30,2,1)`, atunci se va afişa valoarea `2` (deoarece `25=10+15`, `45=15+30`).

► **12.** Pentru un şir dat de `n` elemente numere reale, să se afişeze toate tripletele de elemente consecutive cu proprietatea că al doilea element al tripletului este media aritmetică dintre primul şi al treilea element.

Exemplu: Pentru şirul `v=(2,3,5,10,15,20,9,-2)` cu `n=8` elemente, se vor afişa tripletele `(5,10,15)`, `(10,15,20)` şi `(20,9,-2)`.

Indicaţii: *Dacă memorăm şirul într-un vector `v=(v[0],v[1],...,v[n-1])` cu `n` elemente, atunci tripletele de elemente consecutive sunt de forma* *(v[i],v[i+1],v[i+2]), cu i=0,1,...,n-3. În cadrul unui ciclu, pentru fiecare astfel de triplet, testăm dacă v[i+1]==(v[i]+v[i+2])/2, iar în caz afirmativ afişăm tripletul.*

► **13.** Se citeşte de la tastatură un şir cu n elemente numere întregi. Să se construiască un alt şir cu elementele şirului dat citite invers, de la dreapta la stânga. Să se realizeze apoi aceeaşi inversare memorându-se însă noul şir în acelaşi vector în care s-a citit şirul iniţial (fără a folosi vreun vector suplimentar).

Exemplu: Pentru şirul (1,2,8,5) se va construi şirul (5,8,2,1).

► **14.** Se citeşte de la tastatură un şir de n elemente cu numere întregi. Scrieţi un program care şterge elementul minim din şir, apoi afişează şirul rămas.

Exemplu: pentru şirul (7,5,2,14,8,11) cu n=6 elemente, se va şterge minimul 2, rezultând astfel şirul (7,5,14,8,11).

Indicaţii: *Memorăm şirul într-un vector v=(v[0],v[1],..,v[n-1]). Mai întâi determinăm elementul minim min, precum şi poziţia p a acestuia, cu algoritmul explicat pe larg în acest capitol. Pentru a şterge elementul de pe poziţia p, vom muta cu o poziţie mai la stânga toate elementele aflate după el, adică cele de pe poziţiile p+1,...,n-1. Pentru aceasta, vom folosi un ciclu cu i=p+1,...,n-1, în care la fiecare pas elementul v[i] trece pe poziţia anterioară i-1, prin atribuirea v[i-1]=v[i].*

► **15.** Se citeşte de la tastatură un şir de n elemente numere întregi. Să se insereze la mijlocul şirului media aritmetică a elementelor sale.

Exemplu: Dacă de la tastatură se citeşte şirul (8,-3,12,4) cu n=4 elemente, atunci noul şir obţinut în urma inserării mediei 5 pe poziţia din mijloc va fi (8,-3,5.25,12,4).

► **16. (Bacalaureat iulie 2009 - varianta 24 - enunţ adaptat)**

Se citesc de la tastatură o valoare naturală nenulă n, (3≤n≤100), apoi cele n elemente, distincte, ale unui tablou unidimensional x. Fiecare dintre aceste elemente este un număr natural având cel mult patru cifre. Folosind un algoritm eficient, realizaţi un program care va calcula şi va afişa pe ecran media aritmetică a elementelor care ar rămâne în tabloul x dacă s-ar elimina valoarea minimă şi valoarea maximă din tablou. Valoarea afişată va avea cel mult 3 cifre după virgulă.

Exemplu: dacă se citesc pentru n valoarea 5, iar pentru tabloul x valorile (1,9,4,8,5), programul va afişa una dintre valorile 5.667 sau 5.666.

► **17.** Se citesc de la tastatură cele n elemente ale unui şir dat de numere întregi. Să se afişeze toate perechile de elemente ale şirului (nu neapărat consecutive) cu proprietatea că ambele elemente ale perechii au aceeaşi sumă a cifrelor.

Exemplu: În şirul (14,129,221,65,409,5,32) cu n=7 elemente, avem perechile (14,221), (14,5), (14,32), (221,5), (221,32) şi (5,32) cu suma cifrelor 5.

► **18.** Se dă un vector `v` cu `n` elemente numere întregi. Să se copieze într-un alt vector `u` elementele *strict pozitive* ale vectorului inițial.

Exemplu: Pentru `n=6` și vectorul `v=(7,-8,-15,2,0,13)`, vom obține vectorul `u=(7,2,13)`.

Indicații: *Algoritmul este foarte asemănător cu cel din problema rezolvată **R.V.9**. Elementele strict pozitive din vectorul `v` vor fi adăugate pe rând la sfârșitul vectorului `u` (inițial gol). Folosim un contor `k`, în care vom păstra în permanență poziția (indicele "căsuței") la care am ajuns cu adăugarea în vectorul `u`. Evident, inițial `k=0`, pentru că începem adăugarea cu prima poziție. Parcurgem într-un ciclu pozițiile `i=0,1,...,n-1` ale elementelor vectorul dat `v`, și la fiecare pas, dacă elementul `v[i]` este strict pozitiv, atunci:*

– îl adăugăm la sfârșitul vectorului `u`, pe poziția dată de valoarea curentă a lui `k` (`u[k]=v[i]`);

– incrementăm cu `1` variabila `k`, pentru a pregăti "căsuța" din vectorul `u` în care se va memora următorul număr strict pozitiv din vectorul `v` (la un pas următor al ciclului).

În final, numărul de elemente ale vectorului `u` este chiar `k`.

► **19.** Fiind dat un vector `v` cu `n` elemente numere întregi, să se construiască alți doi vectori: primul va conține numai elementele pare, iar al doilea numai elementele impare ale vectorului inițial.

Exemplu: Pentru `n=7` și `v=(8,-6,11,14,10,-5,-214)`, vor rezulta vectorii `x=(8,-6,14,10,-214)` și `y=(11,-5)`.

► **20. (Bacalaureat iulie 2009, varianta 93)**

Scrieți un program care citește de la tastatură un număr natural `n` (`1≤n≤99`), impar, și construiește în memorie un tablou unidimensional $A=(A_1, A_2,\ldots, A_n)$ cu elementele mulțimii `{1,2,...,n}` astfel încât elementele de pe poziții impare formează șirul crescător `1,2,...,[(n+1)/2]`, iar elementele de pe poziții pare șirul descrescător `n,n-1,...,[(n+1)/2]+1`.

Exemplu: pentru `n=11` se va construi tabloul `A=(1,11,2,10,3,9,4,8,5,7,6)`

► **21. (Bacalaureat iulie 2008, varianta 93)**

Să se introducă într-un vector, apoi să se afișeze ca atare, următorul șir de numere: `1,1,2,1,2,3,1,2,3,4,...`

Numărul de elemente câte vor fi introduse se va preciza anticipat de la tastatură.

Indicații: *Putem observa următoarea grupare a elementelor șirului: `(1)`, `(1,2)`, `(1,2,3)`, `(1,2,3,4)`, etc... Grupa `1` conține valorile de la `1` la `1`, grupa `2` valorile de la `1` la `2`, ș.a.m.d., deci o grupă oarecare `i` va conține valorile de la `1` la `i`, pe care le afișăm banal într-un ciclu. Numai că, neștiindu-se numărul grupelor ci numărul elementelor, pentru trecerea grupelor în revistă nu se poate folosi tot un ciclu `for`, ci un ciclu `while`, în care să se numere elementele afișate, și din care să se iese forțat atunci când s-au parcurs `n` valori. În rest, adăugarea elementelor pe rând într-un vector nou se încadrează în algoritmul clasic.*

► **22.** Se dau două mulțimi definite prin intermediul vectorilor `u` și `v`, cu `m` respectiv `n` elemente. Să se memoreze în vectorul `w` și apoi să se afișeze reuniunea celor două mulțimi (elementele comune și necomune luate o singură dată). La citirea fiecăruia dintre cei doi vectori se va asigura îndeplinirea condiției de mulțime: elementele să fie distincte și în ordine crescătoare.

Exemplu: Pentru mulțimile `u=(2,3,8,9)` și `v=(3,8,11)`, va rezulta vectorul-reuniune `(2,3,8,9,11)`.

► **23. (Bacalaureat iulie 2009, varianta 31 - enunț adaptat)**

Se citesc de la tastatură un tablou unidimensional `x` cu cel mult `100` de elemente, numere naturale cu cel mult `4` cifre fiecare, și un număr natural `n` (`n≤100`), ce reprezintă numărul efectiv de elemente ale tabloului `x`. Scrieți un program care va afișa tabloul obținut în urma schimbarii poziției doar a elementelor impare din tablou astfel încât acestea să apară în ordinea crescătoare a valorilor lor.

Exemplu: pentru `n=6` și `x=(7,11,2,-8,-3,10)` programul va afișa tabloul `(-3,7,2,-8,11,10)`.

Indicații: *Practic algoritmul de sortare prin metoda interschimbărilor directe trebuie modificat, astfel încât la fiecare comparare să se compare între ele doar două elemente `x[i]` și `x[j]` care sunt ambele impare. În cazul în care vreți să folosiți "metoda bulelor", modificările sunt mai profunde !*

► **24. (Bacalaureat iulie 2009, varianta 41 - enunț adaptat)**

Se citesc de la tastatură un număr natural `n` și un tablou unidimensional `a` cu `n` elemente numere întregi. Scrieți un program care afișează valoarea `1` dacă toate elementele tabloului `a` sunt distincte și dacă diferența absolută a oricăror două elemente vecine din tablou este diferită de `1`, respectiv `0` în caz contrar.

Exemplu: pentru tabloul `(7,2,8,11,13)` se va afișa valoarea `1`, iar pentru tablourile `(5,3,9,5)` și `(2,-3,7,6)` se va afișa valoarea `0`.

► **25. (Bacalaureat iulie 2009, varianta 52)**

Scrieți un program care citește de la tastatură un număr natural nenul `n` (`n≤100`) și apoi `n` numere naturale, de maximum patru cifre fiecare, reprezentând elementele unui tablou unidimensional. Programul afișează mesajul `DA` în cazul în care elementele tabloului pot fi rearanjate astfel încât să formeze o progresie aritmetică, iar în caz contrar afișează mesajul `NU`.

Exemplu: dacă `n=6` și tabloul unidimensional are conținutul alăturat, atunci se va afișa `DA`.

`5 10 30 15 25 20`

► **26. (Bacalaureat iulie 2008, varianta 24)**

Scrieți un program care citește de la tastatură un număr natural nenul `n` (`n≤1000`), apoi construiește în memorie și afișează pe ecran un tablou unidimensional `a`, având `n` elemente, ale cărui capete vor fi completate cu toate numerele din mulțimea `{1,2,...,n}` luate alternativ, astfel: valoarea `1` pe prima poziţie, valoarea `2` pe ultima poziţie, valoarea `3` pe a doua poziţie, valoarea `4` pe penultima poziţie, ş.a.m.d.. Elementele tabloului creat se vor afişa cu câte un spaţiu între ele.

Exemplu: pentru `n=9`, tabloul va fi `(1,3,5,7,9,8,6,4,2)`.

► **27.** Se citeşte de la tastatură un şir de `n` elemente numere întregi. Să se afişeze elementele şirului în "pagini" de câte `m` rânduri, fiecare rând conţinând câte `m` elemente separate prin spaţii sau virgule (cu excepţia ultimei pagini care poate conţine mai puţin de `m` rânduri, respectiv a ultimului rând al "ultimei pagini" care poate conţine mai puţin de `m` elemente). După afişarea fiecărei "pagini" programul va aştepta apăsarea unei taste pentru confirmarea trecerii la "pagina" următoare.

Exemplu: Pentru `m=3`, `n=22` şi şirul `(8,2,4,56,87,91,12,41,62,44,65,21,87,30,2,10,35,99,70,13,18,24)` ecranul va arăta astfel:

```
8,2,4
56,87,91
12,41,62
---------
44,65,21
87,30,2
10,35,99
---------
70,13,18
24
```

► **28. (Bacalaureat iulie 2009, varianta 55)**

Scrieţi un program care citeşte de la tastatură un număr natural nenul `n` `(n≤100)` şi `2*n` numere naturale de *maximum* trei cifre; primele `n` valori reprezintă elementele tabloului unidimensional `a`, iar următoarele `n` elementele tabloului unidimensional `b`; fiecare tablou are elementele numerotate începând de la `0`. Programul construieşte în memorie şi afişează pe ecran cele `n` elemente ale unui tablou unidimensional `c`, în care orice element c_i $(0 \le i \le n)$ se obţine conform definiţiei următoare:

$$c_i = \begin{cases} a_i \text{ concatenat cu } b_i, \text{ dacă } a_i < b_i \\ b_i \text{ concatenat cu } a_i, \text{ altfel} \end{cases}$$

Exemplu: dacă `n=3` şi tablourile `a` şi `b` au conţinutul `a=(12,123,345)` şi `b=(1,234,15)`, atunci conţinutul tabloului `c` este următorul: `(112,123234,15345)`.

Probleme de nota 10

► **29.** Fiind dat un vector `v` cu `n` elemente numere întregi, să se construiască un alt vector `u` care să conţină numai pătratele perfecte din `v`, în ordine crescătoare. Aceste pătrate perfecte vor fi copiate de la început în ordine crescătoare, fără a se aplica vectorului `u` nici un algoritm de sortare.

Exemplu: Pentru vectorul `v=(34,9,45,64,18,16,39)`, se va crea vectorul `u=(9,16,64)`.

Indicații: *Atenție, algoritmul prezentat ca indicații la problema 18 nu ne este de nici un folos în acest caz. Presupunând că la un pas oarecare* `i` *al ciclului de parcurgere a vectorului* `v` *(cu* `i=0,1,...,n-1`*) avem în* `u` *adăugate deja* `k` *elemente* `u[0],u[1],...,u[k-1]` *ordonate crescător, adăugarea elementului* `v[i]` *nu se va face la sfârșitul lui* `u`*! Va trebui să găsim poziția pe care ar trebui să o ocupe acea valoare* `v[i]` *în vectorul* `u`*, astfel încât, după introducerea ei, vectorul* `u` *să rămână ordonat crescător. Pentru aceasta, căutăm în* `u` *un indice* `j`*, astfel încât* `u[j]<=v[i]<=u[j+1]`*, și, după ce l-am găsit, inserăm pe* `v[i]` *pe poziția* `j` *în vectorul* `u`*.*

► 30. (Bacalaureat iulie 2009, varianta 12)

Se citesc de la tastatură două valori naturale `m` și `n` (`m<=100`, `n<=100`), iar apoi `m+n` numere întregi de cel mult nouă cifre fiecare. Dintre cele `m+n` numere citite, primele `m` sunt ordonate strict crescător, iar următoarele `n` sunt, de asemenea, ordonate strict crescător. Se cere să se afișeze pe ecran câte din cele `m+n` numere au fost citite o singură dată.

Exemplu: pentru `m=6` și `n=9` și valorile `1, 2, 3, 4, 7, 20, 3, 5, 7, 8, 9, 10, 12, 20, 24`, se va afișa valoarea `9` (numerele care au fost citite o singură dată sunt `1 2 4 5 8 9 10 12 24`).

a) Descrieți un algoritm de rezolvare a acestei probleme, eficient din punct de vedere al timpului de executare și al spațiului de memorie utilizat, explicând în ce constă eficiența acestuia.

b) Scrieți programul corespunzător algoritmului descris.

► 31. (Bacalaureat iulie 2008, varianta 91)

Pentru un număr natural `n` citit de la tastatură (`0<n<100`), se cere să se construiască un vector cu `n` componente numere naturale din mulțimea `{0,1,2}`, astfel încât să nu existe două elemente alăturate egale, iar suma oricăror trei elemente consecutive să fie egală cu `3`. Scrieți programul care generează în memorie vectorul, apoi scrie componentele acestuia pe ecran, în ordine, cu spații între ele.

Exemplu: pentru `n=4`, se vor afișa valorile `1 2 0 1`.

Indicații: *Problema pare grea, dar cu o idee genială ea devine banală. Practic, dacă multiplicăm secvența* `(1,2,0)` *de* `[n/3]` *ori, obținem un șir care respectă cele două cerințe. Iată un exemplu: pentru* `n=11` *avem șirul* `(1,2,0,1,2,0,1,2,0,1,2,0)`*. Obsevăm că sumele a trei elemente alăturate sunt fie* `1+2+0`*, fie* `2+0+1`*, fie* `0+1+2`*, deci întotdeauna 3.*

► 32. (Bacalaureat iulie 2009, varianta 66 - enunț adaptat)

Se citesc de la tastatură un număr natural nenul `n` (`1≤n≤10000`) precum și un tablou unidimensional `a` care conține `n` valori întregi, fiecare dintre aceste valori având cel mult nouă cifre. Realizați un program care determină și afișează pe ecran cel mai mare divizor comun al elementelor tabloului `a`.

Exemplu: în urma apelului, pentru `n=5` și tabloul unidimensional `(12,36,48,6,60)`, se va returna `6`.

▶ **33.** La ora de educație fizică, în vederea apelului de prezență, elevii unei clase s-au așezat într-un șir după înălțime, în ordine crescătoare de la stânga la dreapta. Profesoara dorește să inverseze ordinea elevilor în șir, astfel încât ei să fie aranjați tot după înălțime, dar în ordine *descrescătoare* de la stânga la dreapta. Simulați într-un program algoritmul pe care-l aplică profesoara. De la tastatură se citesc numărul de elevi `n`, precum și înălțimile elevilor. Programul va trebui să testeze dacă așezarea inițială a elevilor respectă condiția de ordine crescătoare, apoi să afișeze elevii după re-ordonarea dictată de profesoară.

Indicații: *Se interschimbă între ei primul elev din șir cu ultimul, al doilea cu penultimul, al treilea cu antepenultimul, ș.a.m.d.. Dacă numărul elevilor este impar, atunci elevul aflat în mijlocul șirului va rămâne pe locul inițial.*

▶ **34. (Bacalaureat iulie 2009, varianta 34)**

Scrieți programul care citește de la tastatură un număr natural `n` `(n<100)`, apoi `n` numere naturale de cel mult trei cifre fiecare și afișează pe ecran cel mai mare număr de valori pare (dintre cele `n` citite) care s-au citit consecutiv de la tastatură.

Exemplu: pentru `n=8` și numerele `12,7,4,16,10,3,6,6` se va afișa `3`.

▶ **35.** Se citesc de la tastatură `n` fracții de forma $(a_1/b_1, a_2/b_2, \ldots, a_n/b_n)$. Să se afișeze fracțiile simplificate "la maximum", precum și fracția ce reprezintă suma celor `n` fracții. Reamintim: o fracție se simplifică "la maximum" împărțind numărătorul și numitorul la cel mai mare divizor comun al lor.

Exemplu: Dacă cele `n=3` fracții de intrare sunt `(6/6,12/18,50/20)`, atunci se vor afișa fracțiile `(1/1,2/3,5/2)`, iar suma acestora este `(25/6)`.

Indicații: *Memorăm numărătorii respectiv numitorii celor `n` fracții în doi vectori (`a[0],a[1],..,a[n-1]`) respectiv (`b[0],b[1],..,b[n-1]`). Într-un ciclu cu `i=0,1,..,n-1`, simplificăm fiecare fracție `a[i]/b[i]` astfel: determinăm `c.m.m.d.c (a[i],b[i])`, apoi împărțim `a[i]` și `b[i]` la acest `c.m.m.d.c.` Pentru a însuma cele `n` fracții, trebuie mai întâi să determinăm cel mai mic multiplu comun al numitorilor (`b[0],b[1],..,b[n-1]`), care se poate calcula după relația `NC=(b[0]*b[1]*..*b[n-1])/c.m.m.d.c.(b[0],b[1],..,b[n-1])`. Calculul celui mai mare divizor comun al numitorilor (`b[0],b[1],..,b[n-1]`) folosește proprietatea de tranzitivitate: astfel, într-un ciclu, vom determina succesiv `d1=c.m.m.d.c. (b[0],b[1])`, `d2=c.m.m.d.c. (d1,b[2])`, ș.a.m.d. În final, într-un alt ciclu se determină numărătorul fracției sumă: `a[0]` se înmulțește cu `NC/b[0]`, `a[1]` cu `NC/b[1]`, etc., adunându-se aceste rezultate.*

▶ **36. (Bacalaureat iulie 2008, varianta 45 - enunț adaptat)**

Se citește de la tastatură un șir de numere naturale mai mici sau egale decât `10000`. Șirul are cel mult `100` de valori și se termină atunci când se introduce o valoare negativă (care nu face parte din el). Scrieți un program care afișează pe ecran toate numerele impare din șir, *în ordine crescătoare*, separate prin câte un spațiu. În cazul în care un anumit element se repetă în șir, el se va afișa o singură dată.

Exemplu: dacă s-a citit şirul (`7,2,1,9,4,0,7,3,22,3`), se vor afişa valorile `1 3 7 9`

►37. (Bacalaureat, iulie 2000, varianta 10)

Scrieţi un program care citeşte de la tastatură cele `10` numere reale ce compun vectorul `a` şi apoi cele `8` numere reale ce constitue componentele vectorului `b` şi afişează pe ecran câte dintre componentele vectorului `a` sunt strict mai mici decât toate componentele vectorului `b`.

Exemplu: dacă `a=(4,8,1,9,5,11,3,43,6,20)` şi `b=(9,9,6,9,9,8,6,9)`, atunci numărul căutat este `4`, deoarece valorile `4,1,5` şi `3` sunt mai mici decât toate elementele lui `b`.

►38. Un polinom $P(X)=a_0+a_1 \cdot X+a_2 \cdot X^2+...+a_n \cdot X^n$ poate fi reprezentat sub forma unui vector `a=(a[0],a[1],...,a[n])` care va memora coeficienţii săi.

a) Fiind dat un polinom `P(X)` şi un întreg `b`, să se calculeze `P(b)` (valoarea lui `P` pentru o valoare `b` a argumentului);

b) Fiind date două polinoame `P(X)` şi `Q(X)`, ambele de gradul `n`, să se determine suma celor două polinoame, `P(X)+Q(X)`, tipărindu-se coeficienţii polinomului-sumă;

c) Fiind date două polinoame `P(X)` şi `Q(X)`, ambele de gradul `n`, să se determine polinoamele ce reprezintă câtul şi restul împărţirii lui `P(x)` la `Q(x)`. Rezultatul se va afişa pe ecran sub forma a două rânduri: pe primul rând coeficienţii polinomului ce reprezintă câtul împărţirii, iar pe al doilea coeficienţii restului.

► 39. (Bacalaureat iulie 2008, varianta 60)

Scrieţi un program care citeşte un număr natural par nenul `n` (`n<100`), precum şi `n` numere naturale de cel mult patru cifre fiecare, apoi determină cea mai mare sumă care poate fi obţinută adunând numai jumătate din numerele citite. Rezultatul se va afişa pe ecran.

Exemplu: pentru `n=6` şi numerele `728, 10, 103, 44, 1000, 94`, se va afişa valoarea `1731` (reprezentând suma `728+103+1000`).

► 40. Se citeşte de la tastatură un vector cu `n` elemente numere întregi. Să se afişeze toate permutările circulare ale vectorului dat. Fiecare permutare circulară se obţine din permutarea precedentă (respectiv din vectorul dat, pentru prima permutare) prin aplicarea următoarelor operaţii: primele `n-1` elemente sunt mutate fiecare cu câte o poziţie mai la dreapta, iar ultimul element trece pe prima poziţie. Procedeul se repetă până când ajungem la permutarea care coincide cu vectorul iniţial. Fiecare dintre permutările rezultate se va afişa pe câte un rând, elementele ce alcătuiesc o permutare urmând a fi separate prin câte un spaţiu.

Exemplu: Permutările circulare ale vectorului `v=(2,5,-1,8)` sunt cele din figură.

```
8 2 5 -1
-1 8 2 5
5 -1 8 2
2 5 -1 8
```

Indicații: Pentru a permuta circular un vector `v=(v[0],v[1],...,v[n-1])` *procedăm astfel:*

– salvăm ultimul element `v[n-1]` *într-o variabilă intermediară* `x`*;*

– într-un ciclu, parcurgem pozițiile `i=0,1,...,n-2` *ale primelor* `n-1` *elemente, și la fiecare pas mutăm elementul* `v[i]` *pe poziția următoare prin atribuirea* `v[i+1]=v[i]`*;*

– în final, aducem ultimul element al configurației inițiale (salvat în `x`*) pe poziția* `0` *(*`v[0]=x`*).*

Evident tot acest algoritm trebuie cuprins într-un alt ciclu, pentru a obține toate permutările posibile.

► **41.** Se citesc de la tastatură `n` numere naturale. Fără a face înmulțirea celor `n` numere, să se determine în câte zerouri se termină produsul lor.

Exemplu: Pentru șirul `(25,6,100,11,15,4000)` cu `n=6` elemente, produsul elementelor este `9900000000`, iar programul va afișa valoarea `8`.

Indicații: *Orice număr se termină în* `0` *dacă este divizibil cu* `2` *și* `5`*. Pentru a afla în câte zerouri se termină produsul elementelor șirului, este suficient să aflăm puterea la care apar factorii* `2` *și* `5` *în descompunerea elementelor, iar numărul de zerouri este minimul dintre aceste puteri.*

► **42.** Se dă un vector `v` cu `n` elemente numere întregi. Fără a folosi un vector auxiliar, să se mute la sfârșitul lui `v` elementele sale nule, păstrând ordinea celorlalte elemente.

Exemplu: dacă inițial `v=(2,3,0,9,0,0,8)`, în final va rezulta `v=(2,3,9,8,0,0,0)`.

Indicații. *Parcurgem vectorul dat* `v` *și pentru fiecare element* `v[i]` *testăm dacă este* `0`*; în caz afirmativ, mutăm toate componentele din dreapta lui* `v[i]` *cu o poziție mai la stânga, micșorăm dimensiunea lui* `v` *cu o unitate și incrementăm un contor* `nr` *care numără elementele nule. În final adăugăm* `nr` *elemente cu valoarea* `0` *la sfârșitul lui* `v`*.*

► **43.** Se dă un șir cu `n` componente naturale ($1 \le n \le 100$). Să se afișeze cel mai mic număr natural care se poate alcătui luând prima cifră a fiecărui element al șirului. *Exemplu:* pentru `n=4` și elementele `(234,7650,19,2)` numărul este `1227`.

Indicații. *Se parcurge șirul memorat într-un vector* `v`*, și pentru fiecare element* `v[i]` *reținem prima cifră, pe care o extragem cu algoritmul clasic de împărțiri la* `10` *și o adăugăm la sfârșitul unui "vector de cifre"* `c`*, inițial gol. Sortăm crescător vectorul, după care prin parcurgerea vectorului sortat de la stânga la dreapta obținem numărul cerut.*

► **44.** Se citește de la tastatură un șir cu `n` elemente numere naturale, nu neapărat distincte. Să se afișeze elementul care apare de cele mai multe ori în șir. Dacă există mai multe astfel de elemente, se vor afișa toate.

Exemplu: pentru `n=5` și elementele `(23,7,11,7,19,7,11,11)` se vor afișa elementele `7` și `11` care apar fiecare de câte trei ori.

Indicații. *Se ordonează crescător vectorul* `v` *în care memorăm șirul. După ordonare, elementele identice vor fi succesive în vector. Folosim alți doi vectori* `u` *și* `f`*: fiecare* `u[i]` *va fi un element distinct al vectorului* `v`*, iar* `f[i]` *va reprezenta frecvența de* *apariţie a lui `u[i]` în vectorul `v`. Parcurgem vectorul sortat `v` şi la fiecare pas comparăm elementul curent `v[i]` cu cel următor `v[i+1]`; dacă sunt diferite, am identificat un nou element distinct al lui `v`, "în persoana" lui `v[i+1]`, pe care îl adăugăm la sfârşitul lui `u`, iniţializând totodată cu `1` un element nou la sfârşitul vectorului `f`; în caz contrar, incrementăm componenta din `f` care indică frecvenţa elementelor egale `v[i]` şi `v[i+1]`.*

► **45.** Se citesc de la tastatură două numere naturale foarte mari, care pot avea până la `70` de cifre fiecare. Să se afişeze suma celor două numere.

Indicaţii. *Evident că numere atât de mari nu pot fi reprezentate în memorie nici măcar cu ajutorul tipului `long`. Singura soluţie este să memorăm cifrele lor în vectori. Pentru adunarea celor două numere vom opera asupra vectorilor. Se parcurg "în paralel" cei doi vectori de cifre de la dreapta la stânga. La fiecare pas, se însumează elementele aflate pe poziţii identice în cei doi vectori, cu memorarea rezultatului pe aceeaşi poziţie într-un al treilea vector. Dacă o astfel de sumă depăşeşte `10`, înseamnă că elementul corespunzător din vectorul-sumă va fi diferenţa dintre suma obţinută şi `10`, iar pe de altă parte vom avea un transport egal cu `1` către poziţia următoare. Dacă suma nu depăşeşte `10`, atunci ea se va memora ca atare în vectorul sumă.*

► **46.** Pentru un concurs judeţean de lupte greco-romane, din partea unui anumit liceu s-au înscris `n` elevi, ale căror greutăţi (exprimate în kg) sunt cunoscute. Din păcate, regulamentul concursului prevede că nu pot participa sportivi a căror greutate se află "la graniţa" dintre două categorii (adică este egală cu limita superioară a unei categorii respectiv limita inferioară a categoriei următoare), aceştia urmând a fi tăiaţi de pe lista celor înscrişi. Realizaţi un program care întocmeşte lista finală a elevilor ce vor participa în concurs. Categoriile de greutate se citesc într-un vector `c` alcătuit din perechi succesive de câte două elemente, în care elementele fiecărei perechi reprezintă limita minimă şi cea maximă de greutate a unei categorii. De la tastatură se mai citesc numărul de elevi `n` precum şi greutăţile acestora.

Exemplu: Dacă vectorul ce conţine greutăţile extreme ale categoriilor de greutate este (`48,54,62,66,74,82,86,92`), atunci categoriile sunt: `[48,54)`, `[54,62)`, `[62,66)`, `[66,74)`, `[74,82)`, `[82,86)`, `[86,92)`. Dacă s-au înscris `n=8` elevi având greutăţile (`57,54,66,90,49,62,58,70`), atunci vor fi eliminaţi de pe listă elevii având greutăţile de `54` şi `66` kg.

Indicaţii: *Citirea vectorului `c` al categoriilor şi a vectorului `g` al greutăţilor se va face cu validare, în sensul că elementele trebuie introduse în ordine crescătoare. Ilustrăm algoritmul cu primul concurent înscris `g[0]`. Pentru început depistăm în care categorie se încadrează greutatea sa: în vectorul `c` al categoriilor parcurgem perechi de forma (`c[i],c[i-1]`) cu `i=n-1,n-2,...,1` şi pentru fiecare pereche testăm dacă greutatea se află în intervalul respectiv; dacă greutatea este chiar egală cu `c[i]` sau `c[i-1]`, atunci concurentul `g[0]` trebuie şters din vectorul `g`. Indiferent de rezultatul acestei testări, mai departe trebuie să trecem peste ceilalţi elevi aflaţi în aceeaşi categorie de greutate cu `g[0]`; acest lucru se realizează printr-o simplă parcurgere a vectorului `g`, bazându-ne pe faptul că dacă în `g` mai există elemente situate în acelaşi interval, ele vor fi cele aflate succesiv imediat după `g[0]`, din cauză că vectorul este sortat crescător. Ceea ce am descris mai sus pentru elevul `g[0]` trebuie implementat pentru toate celelalte elemente ale* *vectorului* `g`. *Ştergerea unui element oarecare* `g[k]` *din vectorul* `g` *se face prin mutarea cu o poziţie mai la stânga a tuturor elementelor aflate după el.*

► **47.** Fiind dat un vector `v` cu `n` elemente numere întregi, să se afişeze cea mai lungă secvenţă (succesiune) de elemente consecutive aflate în ordine crescătoare. Dacă există mai multe astfel de secvenţe, se va afişa una singură.

Exemplu: Pentru vectorul `v=(7,3,5,8,4,9,-1,2,5,6,11,10)` cu `n=12` elemente, programul va tipări secvenţa `(-1,2,5,6,11)`.

► **48.** Vom spune despre un şir că are "aspect de munte", dacă toate elementele aflate până la o poziţie oarecare `k` inclusiv sunt în ordine crescătoare, şi toate elementele situate după poziţia `k` sunt în ordine descrescătoare. În acest caz elementul aflat pe poziţia `k` se numeşte "vârful muntelui". Realizaţi un program care citeşte de la tastatură un şir cu `n` elemente numere întregi şi testează dacă acesta are sau nu "aspect de munte". În caz afirmativ se va tipări "vârful muntelui", iar în caz contrar se va afişa un mesaj.

Exemplu: şirul memorat în vectorul `(2,3,7,11,14,9,8,7,5)` este un "munte" cu vârful reprezentat de elementul `14`, iar vectorul `(2,3,7,11,14,9,8,10,5)` nu are "aspect de munte", din cauza elementului `10` care "strică" ordinea descrescătoare.

***Indicaţii**: Dacă şirul este un "munte", evident că "vârful" său nu poate fi altcineva decât elementul maxim. Se determină maximul* `max` *şi poziţia sa* `k`, *după care se fac două testări: dacă elementele aflate până la poziţia* `k` *sunt în ordine strict crescătoare, şi dacă elementele de după poziţia* `k` *sunt în ordine strict descrescătoare. În cazul în care ambele testări dau rezultatul* `"ADEVĂRAT"`, *am identificat muntele şi vârful său.*

► **49.** Se citeşte de la tastatură un şir de numere naturale. Să se afişeze toate grupurile de numere din şir care au aceeaşi divizori factori primi.

Exemplu: Fie şirul `(12,15,2,36,32)`. Grupurile afişate vor fi: `(12,36)` cu divizorii primi `2` şi `3`, `(15)` cu divizorii primi `3` şi `5`, `(2,32)` cu divizorul prim `2`.

***Indicaţii**. Dacă şirul are* `n` *numere, vom crea doi vectori* `v` *şi* `f`, *astfel: fiecare element* `v[i]` *va fi un număr din şirul dat, iar* `f[i]` *va reprezenta produsul factorilor primi ai lui* `v[i]`, *cu* `i=1,2,...,n`. *Evident că dacă două elemente ale şirului au acelaşi produs de factori primi, atunci elementele în cauză vor avea aceeaşi factori primi, deci în vectorul* `f` *se pot evidenţia grupurile cerute. Ordonăm crescător vectorul* `f`, *folosind oricare dintre algoritmii de sortare studiaţi. În final se afişează grupurile cerute prin parcurgerea vectorului* `f`, *grupurile cerute nefiind altceva decât platourile din vector.*

► **50.** Se dau doi vectori `a` şi `b`, cu `m` respectiv `n` elemente numere întregi (`n>m`). Să se verifice dacă toate componentele vectorului `a` sunt distincte două câte două, iar în caz afirmativ să se testeze dacă acestea, luate în orice ordine, se găsesc în vectorul `b` pe poziţii consecutive. În cazul în care testul a reuşit, să se afişeze fiecare poziţie din vectorul `b` de unde începe secvenţa alcătuită din elementele vectorului `a`.

Exemplu: dacă `a=(3,5,4)` şi `b=(2,5,4,3,9,6,3,4,5,12)`, atunci vectorul `a` se regăseşte de două ori în vectorul `b`: secvenţa `(5,4,3)` care începe de pe poziţia a doua, respectiv secvenţa `(3,4,5)` a cărei poziţie de început este `7`.

Indicații: *Dacă vectorul* **a** *are elemente identice, firește că în șirul sortat acestea se vor găsi pe poziții consecutive. Prin urmare, cea mai simplă modalitate (dar nu neapărat cea mai eficientă !) de a testa dacă elementele sale sunt distincte două câte două, este următoarea: sortăm crescător vectorul, apoi parcugem perechile de elemente de forma* (`v[i]`,`v[i+1]`) *cu* `i=0,1,...,n-2`, *și testăm dacă există vreo pereche cu proprietatea* `v[i]=v[i+1]`.

În cazul în care testul anterior generează ADEVĂRAT, parcurgem elementele vectorului `b` *și pentru fiecare* `v[k]` (*cu* `k=0,1,...,n-1`), *testăm dacă acesta este începutul unei secvențe alcătuite cu elementele lui* `a` *în orice ordine, fiecare luat o singură dată. Pentru aceasta:*

– *mai întâi testăm dacă* `b[k]` *se găsește în vectorul* `a`;

– *în caz afirmativ, verificăm într-un ciclu dacă următoarele* `m` *elemente ale lui* `b` *începând cu poziția* `k` *se găsesc la rândul lor în vectorul* `a`; *pentru aceasta parcurgem secvența alcătuită din elementele* `b[j]` *cu* `j=k,...,k+m-1`; *pentru a testa dacă secvența coincide cu vectorul* `a`, *eventual într-o altă ordine a elementelor, vom folosi un vector* `sem` *cu rol de "semafor".*

V.2. Matrici

V.2.1. Noțiunea de matrice

O matrice este un tabel cu elemente de același tip, dispuse pe linii și coloane. Datorită acestei așezări a elementelor, o matrice este de fapt un tablou bidimensional. Fiecare element al matricii se află pe o anumită linie și pe o anumită coloană. Poziția unui element pe linie se mai numește și **indice de linie**, iar poziția elementului pe coloană se mai numește și **indice de coloană**.

Dacă notăm variabila matrice cu `a`, *atunci elementul de pe linia* `i` *și coloana* `j` *în matricea* `a` *se notează* `a[i][j]`.

Ilustrăm în continuare o matrice `a` cu `3` linii și `4` coloane, având ca elemente numere întregi.

$$\begin{array}{r|l} & \begin{array}{cccc}1 & 2 & 3 & 4\end{array}\ \text{coloana} \\ \hline \begin{array}{r}\text{linia } 1\\ 2\\ 3\end{array} & \begin{bmatrix}-2 & 16 & 8 & 2\\ 13 & 5 & 10 & 6\\ 19 & -1 & 9 & 7\end{bmatrix}\end{array}$$

`a[3][2]=-1`
(elementul de pe linia 3 și coloana 2 este `-1`)

O variabilă-matrice se declară asemănător cu o variabilă-vector, cu deosebirea că în loc de numărul maxim de elemente, trebuie să precizăm două valori: numărul maxim de linii și numărul maxim de coloane.