BAC INFO' 2003

În cadrul acestui articol vom prezenta detalii referitoare la răspunsurile corecte la întrebările testului grilă prezentat în GInfo 13/4 (aprilie 2003). Pentru fiecare întrebare în parte vom preciza care dintre variantele de răspuns este cea corectă, precum și motivele pentru care aceasta a fost aleasă.

Întrebarea #1:
Există mai multe definiții ale conceptului de algoritm. O variantă care include, în principiu, toate caracteristicile unui algoritm este următoarea:

Un algoritm este o succesiune finită de pași care trebuie executați într-o anumită ordine și care duce la rezolvarea unei probleme.

Din această definiție rezultă imediat că afirmațiile corespunzătoare variantelor a) și c) sunt adevărate.

Afirmația corespunzătoare variantei d) este și ea adevărată deoarece, în practică, pot fi găsite mai multe modalități de rezolvare a unei probleme, aceste metode putând fi descrise cu ajutorul unor algoritmi diferiți.

Așadar, singura variantă rămasă este b). Afirmația corespunzătoare acesteia este într-adevăr falsă datorită faptului că există anumite probleme care nu pot fi rezolvate. Exemplul clasic în acest sens îl reprezintă ***problema șchiopătării*** a lui *Turing*.

În concluzie, răspunsul corect este:

b) Poate fi găsit un algoritm pentru rezolvarea oricărei probleme.

Întrebarea #2:
Afirmația corespunzătoare variantei a) este falsă, deoarece există expresii care pot să nu conțină variabile. Un exemplu în acest sens este adunarea a două constante (expresia `2 + 3` este validă și nu conține nici o variabilă).

Afirmația corespunzătoare variantei b) este adevărată deoarece, încă din definirea conceptului de variabilă, aceasta își poate schimba valoarea (de fapt, în marea majoritate a cazurilor, variabilele sunt utilizate pentru ca valorile lor să fie modificate pe parcursul executării algoritmilor).

Valoarea de adevăr a afirmației corespunzătoare variantei c) este discutabilă. Există anumite definiții ale expresiilor în care se precizează că acestea pot fi variabile, constante sau pot fi formate din doi operanzi (care sunt expresii) și un operator. Această afirmație ar putea fi interpretată astfel încât să rezulte că o expresie poate conține cel mult un operator. Totuși, în cazul în care expresiile folosesc ca operanzi alte expresii, acestea din urmă pot conține, la rândul lor, operatori. Așadar, o expresie "compusă" din mai multe expresii poate conține mai mulți operatori. De exemplu, expresia `1 + 2 + 3` conține doi operatori. În plus, deoarece afirmația de la punctul b) este categoric adevărată, este clar că nu putem considera și această afirmație ca fiind adevărată.

Afirmația de la punctul d) este falsă, deoarece există o mulțime de situații în care operațiile nu constau în efectuarea de calcule folosind variabile. Și în acest caz, exemplul efectuării adunării `1 + 2` este suficient pentru a arăta că afirmația este falsă.

În concluzie, răspunsul corect este:

b) Valoarea unei variabile se poate modifica pe parcursul executării unui algoritm.

Întrebarea #3:
În cazul unei structuri repetitive anterior condiționate, execuția corpului este precedată de testarea unei condiții. Dacă aceasta nu este îndeplinită, corpul nu se execută deloc.

Evident, este posibil ca această condiție să fie îndeplinită la a doua iterație. În acest caz, corpul instrucțiunii se va executa o singură dată.

De asemenea, este posibil ca această condiție să fie îndeplinită și la următoarele testări, deci este posibil să fie executat corpul instrucțiunii de mai multe ori.

În concluzie, răspunsul corect este:

d) de zero, una sau mai multe ori.

Întrebarea #4:
Cuvintele `for`, `while` și `if` sunt rezervate atât în limbajul *Pascal*, cât și în limbajul *C*. Ele nu pot fi folosite în alte scopuri. În cazul limbajului *Pascal*, `write` este identificatorul unei proceduri. Chiar dacă acest lucru nu este recomandat, el poate fi utilizat în alte scopuri. În cazul limbajului *C*, `printf` este identificatorul unei funcții și poate fi și el utilizat în alte scopuri deși acest lucru nu este recomandat.

În concluzie, răspunsul corect este:

Versiune Pascal: *Versiune C:*
c) `write` c) `printf`.

Întrebarea #5:
Potrivit sintaxei limbajului *Pascal*, pentru a defini un nou tip de date trebuie utilizat cuvântul cheie `type`. În cazul limbajului *C*, cuvântul cheie utilizat în acest scop este `typedef`.

În concluzie, răspunsul corect este:
Versiune Pascal: *Versiune C:*
a) `type` a) `typedef`.

Întrebarea #6:
Identificatorul folosit pentru a preciza tipul caracter este `char` atât în cazul limbajului *Pascal*, cât și în cazul limbajului *C*.

Pentru a atribui o valoare unei anumite variabile trebuie utilizat operatorul de atribuire care este `":="` pentru limbajul *Pascal* și `"="` pentru limbajul *C*.

În cazul ambelor limbaje, pentru a specifica faptul că se dorește folosirea unui anumit caracter, acest caracter trebuie cuprins între apostrofuri.

În concluzie, răspunsul corect este:
Versiune Pascal:
b) Declarație: `a:char;` Atribuire: `a:='H';`.
Versiune C:
b) Declarație: `char a;` Atribuire: `a='H';`.

Întrebarea #7:
Afirmația nu este adevărată deoarece pot exista funcții a căror listă de parametri să fie vidă.

În concluzie, răspunsul corect este:
b) Nu.

Întrebarea #8:
În cazul limbajului *Pascal*, pentru a determina codul *ASCII* al unui caracter trebuie utilizată funcția `ord`.

În cazul limbajului *C*, nu se face distincție între o variabilă de tip caracter și o variabilă de tip numeric. Caracterele sunt reprezentate direct prin codurile *ASCII*, deci pentru a determina un astfel de cod trebuie doar să utilizăm caracterul inclus între apostrofuri.

În concluzie, răspunsul corect este:
Versiune Pascal: *Versiune C:*
a) `a := ord('*');` a) `a = '*';`.

Întrebarea #9:
Un număr este impar dacă cea mai nesemnificativă cifră binară a sa este `1`. Rezultatul operației `SI` logic între o anumită valoare și valoarea `1` este ultima cifră a reprezentării binare deoarece toți biții reprezentării binare (cu excepția celui mai nesemnificativ) a valorii `1` sunt `0`. Celelalte operații logice pe biți nu pot fi utilizate pentru obținerea valorii celui mai nesemnificativ bit al reprezentării unui număr, motiv pentru care nu pot fi folosite pentru verificarea parității unui număr.

În concluzie, răspunsul corect este:
Versiune Pascal: *Versiune C:*
a) `a` **`and`** `1 = 1` c) `a & 1`.
Notă:

Acest tip de test poate fi utilizat numai dacă valoarea variabilei `a` este un număr întreg pozitiv.

Întrebarea #10:
Din definiția tabloului rezultă că acesta este o structură statică de date ale cărei elemente au același tip. Așadar, afirmațiile corespunzătoare punctelor a) și b) sunt adevărate.

Afirmația corespunzătoare punctului c) este și ea adevărată deoarece, pentru declararea unui tablou în limbajul *Pascal*, trebuie utilizat cuvântul cheie `array`, în timp ce declararea unui tablou în limbajul *C* se realizează fără a utiliza acest cuvânt cheie.

Afirmația corespunzătoare punctului d) este falsă în cazul limbajului *Pascal*, deoarece domeniul unui tablou poate fi orice tip ordinal (cu restricțiile impuse de dimensiunea segmentului de date), așadar primul element poate avea indicele diferit de `1`. Ea este falsă și în cazul limbajului *C* deoarece în cazul acestui limbaj de programare indicele primului element al unui tablou este `0`.

În concluzie, răspunsul corect este:
Versiune Pascal:
d) Primul element al unui tablou are întotdeauna indicele 1.
Versiune C:
d) Primul element al unui tablou are întotdeauna indicele 1.

Întrebarea #11:
În cazul limbajului *Pascal*, funcția care returnează numărul de caractere dintr-un string este `length`.

În cazul limbajului *C*, funcția care returnează această valoare este `strlen`.

În concluzie, răspunsul corect este:
Versiune Pascal: *Versiune C:*
c) `length` c) `strlen`.

Întrebarea #12:
În cazul limbajului *Pascal* un articol este declarat cu ajutorul cuvântului cheie **`record`**. Câmpurile trebuie să apară după acest cuvânt și sunt declarate prin identificatorul și tipul câmpului. Urmează cuvântul cheie **`end`** care indică terminarea secvenței de declarare a articolului.

În cazul limbajului *C* un articol este declarat cu ajutorul cuvântului cheie **`struct`**. Câmpurile trebuie să apară după acest cuvânt, sunt declarate prin identificatorul și tipul câmpului și sunt cuprinse între acolade.

În concluzie, răspunsul corect este:
Versiune Pascal:
a) **`var`** `a:`**`record`** `x,y:real` **`end`**`;`
Versiune C:
a) **`struct`** `{`**`float`** `x,y;}a;`.

Întrebarea #13:
Un fișier text poate conține și alte caractere în afara literelor alfabetului latin (de exemplu cifre, semne de punctuație

etc.), motiv pentru care afirmația corespunzătoare punctului a) este falsă.

Pentru a citi date dintr-un fișier text acesta trebuie deschis pentru citire; așadar afirmația corespunzătoare punctului b) este adevărată.

Fișierele text pot fi deschise pentru adăugare la sfârșitul lor, așadar pot fi adăugate litere. Ca urmare, afirmația corespunzătoare punctului c) este falsă.

Dimensiunea unui fișier text este limitată doar de capacitatea de stocare a discului hard și de sistemul de fișiere utilizat. În orice caz, aceasta nu este limitată la 65535 de caractere, motiv pentru care afirmația corespunzătoare punctului d) este falsă.

În concluzie, răspunsul corect este:

b) Înainte de a prelua date dintr-un fișier text, acesta trebuie deschis pentru citire.

Întrebarea #14:

Tipul corespunzător fișierelor text în limbajul *Pascal* este `text`. Acesta este un identificator de tip predeclarat, nefiind un cuvânt rezervat.

În cazul limbajului *C* există mai multe posibilități de declarare a unei variabile care să reprezinte un fișier. Dintre variantele disponibile, doar construcția `FILE*` poate fi utilizată pentru a declara o astfel de variabilă. Nici această construcție nu reprezintă un cuvânt rezervat.

În concluzie, răspunsul corect este:

Versiune Pascal:

d) nu există nici un astfel de cuvânt cheie.

Versiune C:

d) nu există nici un astfel de cuvânt cheie.

Întrebarea #15:

În cazul limbajului *Pascal*, pentru a realiza corespondența dintre o variabilă și un anumit fișier se utilizează procedura `assign`.

În cazul limbajului *C* există mai multe modalități de a realiza această corespondență. Dintre variantele prezentate, poate fi utilizată funcția `fopen`.

În concluzie, răspunsul corect este:

Versiune Pascal:	*Versiune C:*
a) `assign`	a) `fopen`.

Întrebarea #16:

Secvența prezentată reprezintă descrierea algoritmului clasic de determinare a inversului unui număr. Așadar, răspunsul corect este:

c) determină inversul unui număr (numărul obținut prin citirea cifrelor de la dreapta la stânga).

Întrebarea #17:

Datorită condiției `d < rad` se vor testa divizorii primi strict mai mici decât rădăcina pătrată a numărului a cărui primalitate este verificată. Așadar, dacă un număr este multiplu al rădăcinii sale pătrate (și nu este multiplu al nici unui alt număr mai mic, diferit de 1), atunci numărul va fi considerat a fi prim, cu toate că are un divizor diferit de `1` și de el însuși. Ca urmare, algoritmul nu va funcționa corect în cazul în care se verifică primalitatea unor numere care sunt pătrate perfecte ale unor numere prime. Dintre cele trei numere, singurul care respectă această proprietate este `25`.

Algoritmul nu va intra în ciclu infinit deoarece valoarea numărului testat crește cu `2` la fiecare pas. Până la urmă acesta va ajunge mai mare sau egal cu rădăcina pătrată a numărului a cărui primalitate este testată sau se va determina un divizor.

În concluzie, răspunsul corect este:

d) `25`.

Întrebarea #18:

Primele elemente ale șirului lui *Fibonacci* sunt: `0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597`...

Cel mai mic element format din trei cifre este 144 care este par, deci nu este prim. Următorul element, 233, este prim, deci este cel mai mic element prim al șirului lui *Fibonacci* care are trei cifre.

În concluzie, răspunsul corect este:

a) `233`.

Întrebarea #19:

Elementul neutru al unei operații este acea valoare pentru un operand a cărei utilizare, indiferent de valoarea celuilalt operand, să determine ca rezultatul să aibă aceeași valoare cu acest al doilea operand. Pentru minimul a două numere valoarea corespunzătoare este ∞, deoarece orice număr este mai mic decât ∞.

În concluzie, răspunsul corect este:

b) ∞.

Întrebarea #20:

Algoritmul de sortare prin inserție constă în parcurgerea elementelor vectorului și determinarea poziției elementului curent în șirul parțial sortat. Așadar, răspunsul corect este:

a) poziția elementului curent în șirul parțial sortat la pașii anteriori.

Întrebarea #21:

Pentru a putea interclasa două șiruri este obligatoriu ca acestea să fie sortate. Așadar, răspunsul corect este:

b) șirurile trebuie să fie ordonate în funcție de un același criteriu.

Întrebarea #22:

În cazul căutării binare, la fiecare pas, lungimea șirului în care se caută numerele se înjumătățește. Așadar, dacă șirul are N elemente, după $[\log_2 N] + 1$ pași, lungimea șirului va fi `1`, deci căutarea se va încheia.

Datorită faptului că avem $N = 1000$ este evident că varianta corectă este:

b) cuprins între `7` și `13`.

Întrebarea #23:
Există două modalități de transmitere a parametrilor și anume: prin valoare și prin referință. Așadar, răspunsul corect este:
c) valoare sau referință.

Întrebarea #24:
Afirmația corespunzătoare punctului a) este falsă, deoarece numărul parametrilor unui subprogram nu este limitat la `12`.

Afirmația corespunzătoare punctului b) este falsă, pentru că există posibilitatea de a scrie subprograme care nu conțin structuri repetitive.

Afirmația corespunzătoare punctului c) este falsă, deoarece un subprogram poate conține orice tip de instrucțiuni, inclusiv una prin care se apelează un alt subprogram.

În concluzie, răspunsul corect este:
d) nici una dintre cele trei.

Întrebarea #25:
În cazul limbajului *Pascal*, varianta corespunzătoare punctului a) este corectă deoarece se respectă sintaxa limbajului.

Varianta corespunzătoare punctului b) este incorectă pentru că tipul unui parametru nu poate fi un vector decât dacă se definește acel vector ca tip.

Varianta corespunzătoare punctului c) nu este corectă deoarece procedurile nu au tip.

Varianta corespunzătoare punctului d) nu este corectă deoarece identificatorul procedurii nu poate fi `procedure`, acesta fiind un cuvânt rezervat.

În cazul limbajului *C* situația este mult mai clară, în acest limbaj neexistând proceduri.

În concluzie, răspunsul corect este:
Versiune Pascal:
a) `procedure proc(var a:byte);`.
Versiune C:
a) în limbajul *C* nu există proceduri.

Întrebarea #26:
Datorită faptului că variabila `a` este folosită și ca variabilă locală în cadrul subprogramului `aduna`, modificarea acestei valori în interiorul suprogramului nu are efect în afara acestuia. Ca urmare, în urma apelului, valoarea variabilei globale `a` va rămâne `3`.

Variabila `b` nu este definită în interiorul subprogramului, deci în momentul în care valoarea ei se modifică, operația poate fi efectuată doar asupra variabilei globale. Ca urmare, variabila globală `b` va avea valoarea `5` în urma execuției apelului.

În concluzie, răspunsul corect este:

Versiune Pascal:	*Versiune C:*
c) `3 5`	c) `3 5`.

Întrebarea #27:
Există două posibiltăți de implementare a recursivității: prin autoapelul unui subprogram (recursivitate directă) și prin apelul unui subprogram care la rândul său, apelează primul subprogram (recursivitate indirectă).

În concluzie, răspunsul corect este:
a) directă sau indirectă.
Notă:

Recursivitatea indirectă poate fi realizată și cu ajutorul a trei sau mai multe subprograme care se apelează succesiv, iar ultimul îl apelează pe primul.

Întrebarea #28:
Datorită faptului că funcția determină un produs și în cazul în care valoarea parametrului *n* este `0`, rezultatul este `0`, rezultatul final va fi `0` deoarece se vor realiza doar înmulțiri cu `0`.

Nu se va intra în ciclu infinit deoarece valoarea parametrului scade la fiecare pas, deci va deveni `0` până la urmă.

În concluzie, răspunsul corect este:

Versiune Pascal:	*Versiune C:*
c) `0`	c) `0`.

Notă:

Dacă valoarea returnată ar fi fost `1` pentru cazul în care parametrul ar avea valoarea `0`, funcția ar fi determinat factorialul unui număr.

Întrebarea #29:
În cazul în care problema este definită recursiv nu este obligatoriu (de obicei nici necesar) să fie utilizată metoda *backtracking*. Folosirea ei este indicată într-o mulțime de situații în care trebuie determinate toate soluțiile unei probleme. De obicei, algoritmii care utilizează această metodă sunt foarte lenți, deci ea nu este recomandată în cazul în care trebuie găsit cel mai rapid algoritm (decât dacă nu există o alternativă care duce la obținerea unui algoritm eficient).

În concluzie, răspunsul corect este:
b) când trebuie determinate toate soluțiile unei probleme.

Întrebarea #30:
Algoritmul se oprește după executarea a `m` pași (la începutul celui de-al (`m + 1`) - lea pas), și la fiecare pas se încearcă adăugarea de elemente care sunt mai mari decât elementul de la pasul anterior. Așadar, elementele șirului care reprezintă o soluție sunt ordonate crescător.

Ca urmare, pentru o valoare `n`, se determină toate șirurile formate din `m` elemente distincte, ordonate crescător. Datorită acestei ordonări, nu există posibilitatea de a găsi două șiruri care să conțină aceleași elemente, dar în altă ordine. Deci, aceste șiruri pot fi considerate ca fiind mulțimi.

În concluzie, răspunsul corect este:
Versiune Pascal:
d) determină toate submulțimile mulțimii `{1, 2, …, n}` care au `m` elemente.
Versiune C:
d) determină toate submulțimile mulțimii `{1, 2, …, n}` care au `m` elemente.

Întrebarea #31:
Este evident că pe fiecare linie se poate așeza un singur turn. Așadar, va trebui să determinăm posibilitățile de a alege coloanele pe care se vor așeza cele opt turnuri. Datorită faptului că pe fiecare coloană poate apărea un singur turn și nu există restricții suplimentare, soluțiile vor fi date de toate posibilitățile de a alege o ordine în care sunt scrise numerele cuprinse între `1` și `8`. Așadar, problema este echivalentă cu determinarea tuturor permutărilor șirului (`1, 2, 3, 4, 5, 6, 7, 8`).

În concluzie, răspunsul corect este:
b) problema determinării tuturor permutărilor mulțimii `{1, 2, 3, 4, 5, 6, 7, 8}`.

Întrebarea #32:
Eroarea apare în linia `21` pentru versiunea *Pascal*, respectiv linia `17` pentru versiunea *C*, deoarece condiția `pas < n` permite executarea a cel mult `n - 1` pași. Așadar, o submulțime cu `n` elemente (deci submulțimea care conține toate elementele mulțimii) nu va putea fi determinată.

În concluzie, răspunsul corect este:

Versiune Pascal:	*Versiune C:*
c) `21`	c) `17`.

Întrebarea #33:
Se observă că subprogramul `Back` este apelat cu valoarea `n` pentru parametrul `pas` și, prin autoapelare, valoarea parametrului scade.

O eventuală soluție este afișată doar atunci când valoarea parametrului `pas` este `n + 1`, ceea ce nu este posibil datorită modului în care variază valoarea acestui parametru.

Așadar, subprogramul `ScrieSolutie` nu va fi apelat niciodată, deci pe ecran nu se va scrie nimic.

În concluzie, răspunsul corect este:

Versiune Pascal:	*Versiune C:*
a) nimic	a) nimic.

Întrebarea #34:
Numărul permutărilor unui șir cu N elemente este $N!$. Numărul submulțimilor unei mulțimi cu N elemente este 2^N. Numărul submulțimilor care au exact M elemente este C_N^M.

Submulțimile care au cel puțin 3 elemente, au evident, 3, 4, 5 sau 6 elemente. Numărul total al acestora va fi:

$$C_6^3 + C_6^4 + C_6^5 + C_6^6 = 20 + 15 + 6 + 1 = 42.$$

Pentru $N = 6$ vom avea $6! = 720$ permutări și $2^6 = 64$ submulțimi.

În concluzie, răspunsul corect este:
b) `a = 720, b = 64, c = 42`.

Întrebarea #35:
Afirmația corespunzătoare punctului a) este adevărată pentru că stiva funcționează pe baza principiului ultimul sosit, primul servit. Așadar, ultimele elemente inserate în stivă sunt eliminate primele. Ca urmare, elementele sunt eliminate în ordinea inversă inserării lor.

Afirmația corespunzătoare punctului b) este adevărată deoarece coada funcționează pe baza principiului primul sosit, primul servit. Așadar, primele elemente inserate în coadă sunt eliminate primele. Ca urmare, elementele sunt eliminate în ordinea inserării lor.

Afirmația corespunzătoare punctului c) este adevărată pentru că recursivitatea constă în copierea pe o stivă a diferitelor informații necesare unui apel a unui subprogram (parametri, adresă de retur) și eliminarea lor în momentul încheierii execuției apelului. Ca urmare, aceste operații pot fi simulate dacă se utilizează o altă stivă.

Afirmația corespunzătoare punctului d) este falsă deoarece recursivitatea (atât cea directă, cât și cea indirectă) poate fi simulată folosind o stivă și nu o coadă.

În concluzie, răspunsul corect este:
d) cozile pot fi utilizate pentru a simula recursivitatea indirectă.

Întrebarea #36:
Pentru a insera un element într-o listă dublu înlănțuită trebuie modificate referințele sale spre predecesorul și succesorul său. Succesorul predecesorului va fi noul element, iar predecesorul succesorului va fi tot noul element, deci trebuie modificate încă două referințe.

Așadar, se modifică un număr total de patru referințe, deci răspunsul corect este:
b) patru referințe.

Întrebarea #37:
În urma execuției operațiilor `Adaugă_în_stivă(1);` și `Adaugă_în_stivă(2);` stiva va conține elementele `1` și `2`, vârful stivei fiind `2`. Acesta este eliminat în urma execuției operației `Elimină_primul_element_al_stivei;` deci stiva va conține un singur element: `1`.

În continuare se adaugă elementele `3` și `4` (prin operațiile `Adaugă_în_stivă(3);` și `Adaugă_în_stivă(4);`), stiva conținând elementele `1`, `3` și `4`, iar vârful fiind `4`. Acesta va fi scris de două ori deoarece se execută de două ori operația `Scrie_primul_element_al_stivei;`. Pe ecran va apărea (în acest moment) **4 4**.

Acum, prin intermediul operațiilor `Adaugă_în_stivă(5);` și `Adaugă_în_stivă(6);`, vom adăuga în stivă elementele `5` și `6`, aceasta conținând acum cinci elemente: `1`, `3`, `4`, `5` și `6`, vârful fiind `6`.

Prin execuția operației `Elimină_primul_element_al_stivei;` acest vârf este eliminat, stiva va rămâne cu patru elemente: `1`, `3`, `4` și `5` , iar noul vârf va fi `5`.

Acest nou vârf este și el eliminat în urma execuției următoarei operații `Elimină_primul_element_al_stivei;` deci vom avea trei elemente în stivă (`1`, `3` și `4`) iar vârful va fi `4`.

Operațiile `Adaugă_în_stivă(7);` și `Adaugă_în_stivă(8);` vor duce la adăugarea elementelor `7` și `8`; stiva va conține elementele `1`, `3`, `4`, `7` și `8`, vârful fiind `8`. Acesta este

afișat în urma execuției operației `Scrie_primul_element_al_stivei;`, deci pe ecran va apărea acum **4 4 8**.

Prin execuția operației `Adaugă_în_stivă(9);` se adaugă elementul `9` care devine noul vârf al stivei care conține acum elementele `1`, `3`, `4`, `7`, `8` și `9`.

Noul vârf este imediat eliminat datorită operației `Elimină_primul_element_al_stivei;` deci revenim în situația în care avem în stivă elementele `1`, `3`, `4`, `7` și `8`, iar vârful este 8.

Prin operația `Scrie_primul_element_al_stivei;` se va afișa valoarea acestui vârf (`8`), deci pe ecran vom avea **4 4 8 8**.

Următoarea operație (`Adaugă_în_stivă(10);`) duce la adăugarea elementului `10`. Vom avea o stivă cu șase elemente (`1`, `3`, `4`, `7`, `8` și `10`) al cărei vârf este `10`.

Operația `Elimină_primul_element_al_stivei;` va duce la eliminarea elementului `10` din vârful stivei, deci, din nou, aceasta va conține elementele `1`, `3`, `4`, `7` și `8`, vârful fiind `8`.

Următoarea operație (`Elimină_primul_element_al_stivei;`) va duce la eliminarea vârfului (`8`), deci vom rămâne cu o stivă care conține patru elemente (`1`, `3`, `4` și `7`) care are în vârf elementul `7`.

Această valoare va fi afișată datorită execuției operației `Scrie_primul_element_al_stivei;`. În acest moment, pe ecran, vom avea **4 4 8 8 7**.

Operația `Elimină_primul_element_al_stivei;` va duce la eliminarea elementului `7` din vârful stivei. Acum, aceasta va conține elementele `1`, `3` și `4`, vârful fiind `4`.

Prin următoarea operație (`Elimină_primul_element_al_stivei;`) se va elimina și elementul `4`, deci vom avea o stivă cu două elemente (`1` și `3`) care are vârful `3`.

Valoarea este scrisă datorită operației `Scrie_primul_element_al_stivei;` deci pe ecran va apărea acum **4 4 8 8 7 3**.

Vom rămâne cu un singur element (`1`) după execuția operației `Elimină_primul_element_al_stivei;` care va fi afișat datorită operației `Scrie_primul_element_al_stivei;`. În acest moment pe ecran vom avea **4 4 8 8 7 3 1**.

Acest ultim element este eliminat datorită execuției instrucțiunii (`Elimină_primul_element_al_stivei;`), deci stiva va rămâne vidă în urma executării tuturor operațiilor.

În concluzie, suma elementelor afișate este `4 + 4 + 8 + 8 + 7 + 3 + 1 = 35`, deci răspunsul corect este:

a) `35`.

Întrebarea #38:

Potrivit definiției conceptului de santinelă, acesta este un element fictiv care este utilizat pentru a ușura modul în care se realizează anumite operații cu structurile dinamice de date el nefăcând parte din structură din punct de vedere teoretic. Ca urmare, răspunsul corect este:

a) un element fictiv care nu face parte din structura propriu-zisă și care este utilizat pentru simplificarea implementării operațiilor de inserare și eliminare.

Întrebarea #39:

Afirmația corespunzătoare punctului a) este falsă datorită faptului că din definiția grafurilor neorientate nu rezultă obligativitatea existenței unei muchii. De exemplu, un graf poate fi format dintr-un singur vârf.

Afirmația corespunzătoare punctului b) este adevărată deoarece definiția grafurilor neorientate nu impune nici o limită asupra numărului nodurilor izolate.

Afirmația corespunzătoare punctului c) este adevărată deoarece o muchie leagă două vârfuri, deci este identificată în mod unic prin perechea de noduri care reprezintă extremitățile ei.

Afirmația corespunzătoare punctului d) este adevărată deoarece respectă definiția lungimii unui ciclu.

În concluzie, răspunsul corect este:

a) un graf neorientat trebuie să conțină cel puțin o muchie.

Întrebarea #40:

Afirmația corespunzătoare punctului a) este falsă pentru că un subgraf se obține prin eliminarea unor noduri ale unui graf, deci numărul acestora va fi mai mic în subgraf decât în graful inițial.

Afirmația corespunzătoare punctului b) este falsă deoarece conexitatea unui graf nu este o condiție suficientă (deși este necesară) pentru ca în graful respectiv să existe un ciclu eulerian.

Afirmația corespunzătoare punctului c) este adevărată deoarece, într-un graf complet cu N noduri, fiecare vârf este legat prin câte o muchie de toate celelalte N - 1 noduri. Așadar, toate nodurile au gradul N - 1, deci un graf complet este întotdeauna și un graf regulat.

Afirmația corespunzătoare punctului d) este falsă deoarece există o mulțime de grafuri hamiltoniene care nu sunt și grafuri regulate.

În concluzie, răspunsul corect este:

c) un graf complet este întotdeauna și un graf regulat.

Întrebarea #41:

O linie (sau o coloană) a matricei de adiacență va conține un număr de elemente cu valoarea `1` egal cu numărul muchiilor care au ca extremitate nodul corespunzător liniei respective. Așadar, gradul unui nod este dat de numărul elementelor cu valoarea `1` de pe linia (coloana) corespunzătoare.

Ca urmare, cele `10` noduri ale grafului vor avea gradele `5`, `8`, `4`, `5`, `6`, `4`, `4`, `4`, `4`, respectiv `4`.

În concluzie, cel mai mare grad este 8, deci răspunsul corect este:

c) `8`.

Întrebarea #42:

Într-un graf neorientat gradul unui vârf este dat de numărul total al arcelor care "intră" sau "ies" din nodul respectiv. Numărul arcelor care "intră" într-un nod reprezintă gradul interior, iar numărul celor care "ies" reprezintă gradul exterior.

Așadar, gradul unui nod va fi suma dintre gradul interior și cel exterior, deci răspunsul corect este:
c) suma dintre gradul interior și gradul exterior.

Întrebarea #43:
Variantele de la punctele a), b) și c) sunt corecte, ele descriind exact corespondențele dintre noțiuni (muchie - arc, ciclu - circuit, lanț - drum).

Varianta de la punctul d) nu este corectă deoarece noțiunea de adiacență este aceeași atât pentru grafurile orientate, cât și pentru cele neorientate. Noțiunea de incidență are o altă semnificație.

În concluzie, răspunsul corect este:
d) noțiunii de adiacență în grafurile neorientate îi corespunde noțiunea de incidență în grafurile orientate.

Întrebarea #44:
Afirmația corespunzătoare punctului a) este adevărată pentru că o frunză este legată printr-o muchie doar de părintele său, deci are gradul `1`.

Afirmația corespunzătoare punctului b) este adevărată dat fiind faptul că toate nodurile (cu excepția rădăcinii) sunt fii ai rădăcinii, ai fiilor rădăcinii, ai fiilor fiilor rădăcinii etc., deci au un părinte.

Afirmația corespunzătoare punctului c) este falsă deoarece într-un arbore un nod poate avea mai mulți fii (care sunt toți frați). Un nod poate avea cel mult un frate doar în arborii binari.

În concluzie, răspunsul corect este:
c) orice nod al unui arbore cu rădăcină poate avea cel mult un frate.

Întrebarea #45:
Datorită faptului că există muchiile (1, 2) și (1, 5), nodurile 2 și 5 vor avea ca părinte nodul 1.

Singurul nod care este legat de nodul 2 printr-o muchie și pentru care nu am stabilit încă părinții este 7 (există o muchie între 2 și 7). Așadar, părintele nodului 7 va fi 2.

De nodul 5 sunt legate prin muchiile (4, 5) și (3, 5) nodurile 3 și 4, deci acestea îl vor avea ca părinte pe 5.

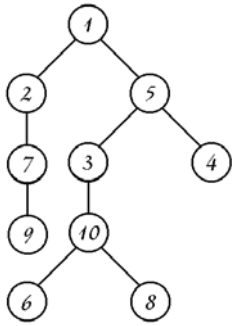
Nodul 9 este legat de nodul 7 prin muchia (7, 9), deci 7 va fi părintele nodului 9.

Nodul 10 este legat de nodul 3 prin muchia (3, 10), deci părintele nodului 10 va fi 3.

În sfârșit, de nodul 10 sunt legate prin muchiile (6, 10) și (8, 10) nodurile 6 și 8 care îl vor avea ca părinte.

Rezultă că:
- părintele nodului 2 este 1;
- părintele nodului 3 este 5;
- părintele nodului 4 este 5;
- părintele nodului 5 este 1;
- părintele nodului 6 este 10;
- părintele nodului 7 este 2;
- părintele nodului 8 este 10;
- părintele nodului 9 este 7;
- părintele nodului 10 este 3.

Arborele cu rădăcină definit prin lista de muchii dată este reprezentat grafic în figura următoare:



În concluzie, răspunsul corect este:
a) `1 5 5 1 10 2 10 7 3`.

Despre testul GInfo

Deși a avut doar un caracter orientativ, testul grilă publicat în *GInfo* a atras atât atenția cititorilor, cât și a multor elevi care nici măcar nu au auzit până acum de existența revistei noastre.

Versiunea *online* a testului a fost folosită de mii de elevi care au dorit să își evalueze cunoștințele.

Testul a fost bine primit și de profesori, deoarece în momentul publicării lui nu exista un alt test care să poată fi utilizat pentru familiarizarea elevilor cu modul în care se va desfășura în acest an examenul de *Bacalaureat* la disciplina *Informatică*.

Între timp, au apărut și teste furnizate de *Ministerul Educației și Cercetării*. Deși ar putea părea bizar, și caracterul acestora este doar orientativ. De exemplu, metoda *backtracking* lipsește din aceste teste chiar dacă, credem noi, ea va apărea cu siguranță la examen.

Încheiem prin a mulțumi tuturor celor care s-au arătat interesați de testul nostru, mulțumim tuturor celor care ne-au felicitat și urăm mult succes tuturor elevilor care se pregătesc în vederea susținerii examenului de Bacalaureat la începutul acestei veri.

Grila corectă

1. a.☐ b.☒ c.☐ d.☐	16. a.☐ b.☐ c.☒ d.☐	31. a.☐ b.☒ c.☐ d.☐
2. a.☐ b.☒ c.☐ d.☐	17. a.☐ b.☐ c.☐ d.☒	32. a.☐ b.☐ c.☒ d.☐
3. a.☐ b.☐ c.☐ d.☒	18. a.☒ b.☐ c.☐ d.☐	33. a.☒ b.☐ c.☐ d.☐
4. a.☐ b.☐ c.☒ d.☐	19. a.☐ b.☒ c.☐ d.☐	34. a.☐ b.☒ c.☐ d.☐
5. a.☒ b.☐ c.☐ d.☐	20. a.☒ b.☐ c.☐ d.☐	35. a.☐ b.☐ c.☐ d.☒
6. a.☐ b.☒ c.☐ d.☐	21. a.☐ b.☒ c.☐ d.☐	36. a.☐ b.☒ c.☐ d.☐
7. a.☐ b.☒ c.☐ d.☐	22. a.☐ b.☒ c.☐ d.☐	37. a.☒ b.☐ c.☐ d.☐
8. a.☒ b.☐ c.☐ d.☐	23. a.☐ b.☐ c.☒ d.☐	38. a.☒ b.☐ c.☐ d.☐
9. a.☐ b.☐ c.☒ d.☐	24. a.☐ b.☐ c.☐ d.☒	39. a.☒ b.☐ c.☐ d.☐
10. a.☐ b.☐ c.☐ d.☒	25. a.☒ b.☐ c.☐ d.☐	40. a.☐ b.☐ c.☒ d.☐
11. a.☐ b.☐ c.☒ d.☐	26. a.☐ b.☐ c.☒ d.☐	41. a.☐ b.☐ c.☒ d.☐
12. a.☒ b.☐ c.☐ d.☐	27. a.☒ b.☐ c.☐ d.☐	42. a.☐ b.☐ c.☒ d.☐
13. a.☐ b.☒ c.☐ d.☐	28. a.☐ b.☐ c.☒ d.☐	43. a.☐ b.☐ c.☐ d.☒
14. a.☐ b.☐ c.☐ d.☒	29. a.☐ b.☒ c.☐ d.☐	44. a.☐ b.☐ c.☒ d.☐
15. a.☒ b.☐ c.☐ d.☐	30. a.☐ b.☐ c.☐ d.☒	45. a.☒ b.☐ c.☐ d.☐