

## Tema 2 - Stive, cozi, liste înlănțuite

1. Să se implementeze o stivă sau o coadă utilizând liste înlănțuite. Creați inițial o structură NOD cu două câmpuri - un câmp de tip `int` pentru informație și un câmp de tip pointer la NOD pentru legătura la elementul următor. Creați apoi o structură STIVA, care să aibă un câmp de tip pointer la NOD, care reprezintă vârful stivei. De asemenea structura STIVA trebuie să conțină metodele PUSH, POP și TOP. PUSH adaugă un element în vârful stivei, POP extrage elementul din vârful stivei, iar TOP returnează valoarea aflată în vârful stivei. Adăugați de asemenea o metodă ISEMPY, care verifică dacă stiva este vidă. De preferință utilizați și un constructor, care să inițializeze vârful stivei cu NULL. În funcția `main` a programului se va declara o stivă. Apoi se vor introduce  $n$  elemente, cu  $n$  citit de la tastatură. Aceste elemente se extrag apoi pe rând și se afișază în ordinea de extragere. (1p)
2. Se citește dintr-un fișier o expresie aritmetică alcătuită din numere întregi fără semn, operatorii aritmetici  $+$ ,  $-$ ,  $*$ ,  $/$  și paranteze rotunde. Atenție, caracterele din șir NU sunt despărțite prin spații! Să se construiască forma poloneză postfixată pentru expresia aritmetică dată și să se evalueze expresia. Utilizați stive. (Este de preferat utilizarea unui vector pentru stocarea elementelor stivei. Se cere utilizarea unei structuri STIVA!) (4p)  
Punctajul maxim se acordă dacă: se pot utiliza și numere cu mai mult de o cifră, sunt semnalate erorile din expresie și nu se blochează programul, dacă expresia nu este corectă.

### Exemple:

- $2 + + + 3$  nu este o expresie corectă. Programul semnalează că există prea mulți operatori.
- $2 * ((3 + 4)$  nu este o expresie corectă. Programul semnalează faptul că parantezarea nu este corectă.  
**Observație:** chiar dacă numărul de paranteze deschise este egal cu cel de paranteze închise, parantezarea poate să nu fie corectă. De exemplu  $))(($  sau  $))(($ .
- $23\# + a$  nu este o expresie corectă. Programul semnalează faptul că apar caractere nepermise  $\#$ ,  $a$ .

- $3 * 4 - 3 * (24 - 12) - 7$ . Programul returnează valoarea  $-31$ . Forma poloneză postfixată este  $34 * 32412 - * - 7-$

Algoritmul este descris în documentația de pe e-learning.

3. Să se implementeze o coadă circulară. Pentru aceasta se cere crearea unei structuri COADA care să conțină un câmp de tip vector de int numit DATA, care stochează elementele din coadă, un câmp SIZE\_MAX de tip int, care reprezintă capacitatea maximă a cozii, două câmpuri de tip int BEGIN și END, care reprezintă poziția în vectorul DATA a primului și respectiv poziția de după ultimul element din coadă. De asemenea se cer în structura COADA, metodele PUSH, POP, FRONT, ISEMPY, unde PUSH adaugă un element după ultimul element din coada, POP extrage primul element din coadă, FRONT returnează valoarea aflată la începutul cozii și ISEMPY verifică dacă este vidă sau nu coada. În funcția main se declară o variabilă de tip COADA, se inserează pe rând  $n$  elemente, cu  $n$  citit de la tastatură, apoi se extrag aceste elemente pe rând și se afișază în ordinea extragerii. (1p)
4. Să se implementeze algoritmul de sortare *Bucket-sort* utilizând liste dublu înlanțuite. Utilizați o structură LISTA\_DUBLU, care să aibă ca membrii un pointer la NOD, reprezentând capul listei și metodele:
  - INSERT\_SORT - inserează o cheie într-o listă ordonată pe poziția potrivită.
  - MERGE - unește lista cu altă listă transmisă ca parametru
  - INSERT - inserează în capul listei.
  - PRINT - afișază elementele stocate în listă.

Utilizați un constructor pentru inițializarea capului listei cu NULL.

Structura NOD utilizată în lista dublu înlanțuită trebuie să aibă câmpurile INFO, NEXT și PREV.

Algoritmul poate fi găsit în cartea "Introduction to Algorithms third edition" de T.C. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein (2p)

5. Să se implementeze o listă simplu înlanțuită cu funcționalitățile descrise în continuare. Se cere utilizarea unei structuri NOD care are două câmpuri: un câmp int pentru informație și un câmp de tip pointer la NOD pentru legătura către următorul element. Se cere utilizarea unei structuri LISTA care are ca membru un pointer la NOD reprezentând capul listei și metodele:
  - INSERT(k) - inserează un element cu cheia  $k$  în capul listei (0.25 p)

- SEARCH( $k$ ) - caută o cheie  $k$  în listă - returnează nodul cu cheia  $k$  sau NULL (0.25 p)
- DELETE( $k$ ) -șterge un element  $k$  din listă, (0.25 p)
- ISEMPY - verifică dacă lista e vidă (0.25 p)
- SWAP( $x,y$ ) - schimbă ordinea elementelor  $x$  și  $y$  în listă ( $x, y$  sunt două noduri) (0.5 p)
- SORT - sortează elementele listei în ordine crescătoare (1 p)
- INSERT\_BEFORE( $k,m$ ) - inserează o cheie  $k$  înaintea un nod cu o anumită cheie  $m$  dată, sau la sfârșitul listei, dacă  $m$  nu există în listă. (0.5 p)
- PRINT\_LIST - afișează elementele din listă (0.25 p)

Utilizați un constructor pentru a inițializa capul listei cu NULL. (0.25p)

În funcția *main* realizați un menu cu ajutorul unei instrucțiuni *switch*, prin care se oferă 8 opțiuni, fiecare corespunzătoare uneia dintre funcțiile de mai sus, precum și o opțiune de EXIT. Într-o instrucțiune *while*, se citesc și se execută opțiuni până la alegerea opțiunii de EXIT.

**ATENȚIE:** Nici o funcție nu trebuie să dea eroare de execuție, dacă se apelează pe o listă vidă!!!

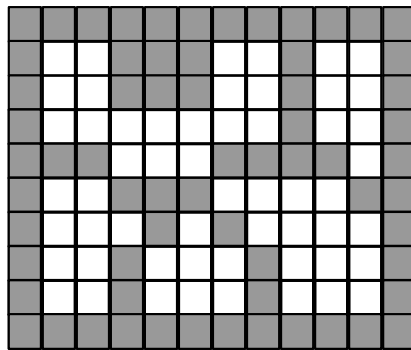


Figure 1: Reprezentarea etajului unei clădiri

6. Un anumit etaj al unei clădiri este reprezentat schematic sub forma unei matrice ce conține valorile -1 și 0, unde -1 reprezintă zid și 0 reprezintă spațiu liber. Pereții sunt de grosime 1 și ușile nu sunt marcate (se consideră tot perete).

- Determinați numărul de încăperi ale etajului respectiv. (1p)
- Determinați încăperea cu suprafața maximă. (1p)
- Care perete poate fi dărâmat (o poziție marcată cu -1 se va marca cu 0)
  - să se obțină o încăpere de suprafață maximă? (2p)

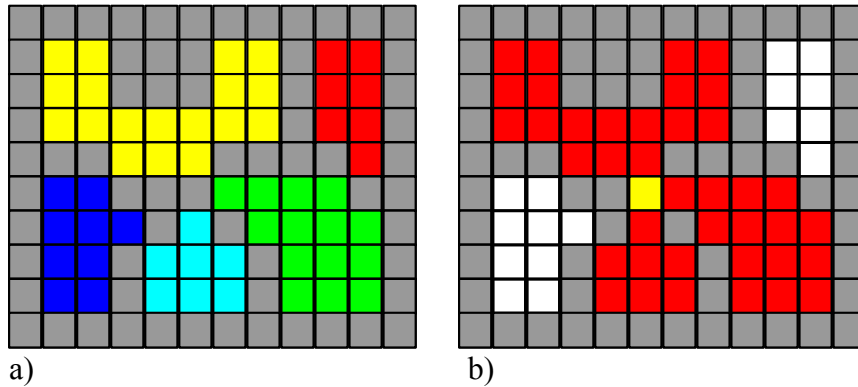


Figure 2: a. Camerele colorate cu culori diferite. b. Dacă se șterge elementul de zid marcat cu galben se obține cea mai mare cameră posibilă.

În verificarea apartenenței la aceeași cameră în matrice se consideră doar vecinătățile pe orizontală și verticală.

Nu utilizați funcții recursive!

**Exemplu:** Se consideră etajul reprezentat de către matricea din figură 1. Este o matrice de dimensiuni  $10 \times 12$  în care pereții sunt colorați cu gri (deci valorile -1) și golurile corespunătoare camerelor cu alb.

Se observă că există 5 camere, fiecare marcate cu altă culoare în figura 2 a. Camera marcată cu galben are suprafața de 18 unități, camera roșie de 7 unități, camera albastră de 9 unități, camera cyan de 7 unități și camera verde de 14 unități.

În figura 2b. se observă faptul că, dacă se elimină zidul de pe poziția marcată cu galben, se unesc trei încăperi și se obține cea mai mare încăpere posibilă, cu suprafața de 40 de unități.

- Se consideră un labirint reprezentat printr-o matrice în care zidurile sunt marcate prin  $-1$  și drumurile prin  $0$ . În labirint se află un șoricel pe poziția  $(x_0, y_0)$  și o bucată de brânză pe poziția  $(x_1, y_1)$ . Să se găsească un drum (de preferință de lungime minimă) de la șoricel la brânză. Nu utilizați recursivitate! Utilizați o coadă. (2p)

**Evaluare:** Rezolvați la alegere probleme. Fiecare problemă are alături punctajul aferent. Se acordă pentru această temă suma punctajelor problemelor rezolvate, dar maxim nota 10. Un punct este din oficiu. Se tine cont de criteriile generale de evaluare, prezentate în lista de criterii de pe platformă de e-learning de la prima unitate de învățare.