Tema 2 - Stive, cozi, liste înlănţuite

1. Să se implementeze o stivă sau o coadă utilizând liste înlănţuite. Creaţi iniţial o structură NOD cu două câmpuri - un câmp de tip int pentru informaţie şi un camp de tip pointer la NOD pentru legătura la elementul următor. Creaţi apoi o structură STIVA, care să aibă un câmp de tip pointer la NOD, care reprezintă vârful stivei. De asemenea structura STIVA trebuie să conţină metodele PUSH, POP şi TOP. PUSH adaugă un element în vârful stivei, POP extrage elementul din vârful stivei, iar TOP returnează valoarea aflată în vârful stivei. Adăugaţi de asemenea o metodă ISEMPTY, care verifică dacă stiva este vidă. De preferinţă utilizaţi şi un constructor, care să iniţializeze vârful stivei cu NULL. În funcţia main a programului se va declara o stiva. Apoi se vor introduce n elemente, cu n citit de la tastatură. Aceste elemente se extrag apoi pe rând şi se afişază în ordinea de extragere. (1p)

2. Se citeşte dintr-un fişier o expresie aritmetică alcătuită din numere întregi fără semn, operatorii aritmetici $+$, $-$, $*$, $/$ şi paranteze rotunde. Atenţie, caracterele din şir NU sunt despărţite prin spaţii! Să se construiască forma poloneză postfixată pentru expresia aritmetică dată şi să se evalueze expresia. Utilizaţi 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 şi numere cu mai mult de o cifră, sunt semnalate erorile din expresie şi nu se blochează programul, dacă expresia nu este corectă.
   Exemple:

   - $2 + + + 3$ nu este o expresie corectă. Programul semnalează că există prea mulţi operatori.
   - $2 * ((3 + 4)$ nu este o expresie corectă. Programul semnalează faptul că parantezarea nu este corectă.
     Observaţie: 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 documentaţia de pe e-learning.

3. Să se implementeze o coadă circulară. Pentru aceasta se cere crearea unei structuri COADA care să conţină 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 şi END, care reprezintă poziţia în vectorul DATA a primului şi respectiv poziţia de după ultimul element din coadă. De asemenea se cer în structura COADA, metodele PUSH, POP, FRONT, ISEMPTY, unde PUSH adaugă un element după ultimul element din coada, POP extrage primul element din coadă, FRONT returnează valoarea aflată la începutul cozii şi ISEMPTY verifică dacă este vidă sau nu coada. În funcţia 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 şi se afişază în ordinea extragerii. (1p)

4. Să se implementeze algoritmul de sortare *Bucket-sort* utilizând liste dublu înlănţuite. Utilizaţi o structură LISTA_DUBLU, care să aibă ca membrii un pointer la NOD, reprezentând capul listei şi metodele:

   - INSERT_SORT - inserează o cheie într-o listă ordonată pe poziţia potrivită.
   - MERGE - uneşte lista cu altă listă transmisă ca parametru
   - INSERT - inserează în capul listei.
   - PRINT - afişază elementele stocate în listă.

   Utilizaţi un constructor pentru iniţializarea capului listei cu NULL.

   Structura NOD utilizată în lista dublu înlănţuită trebuie să aibă câmpurile INFO, NEXT şi 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 înlănţuită cu funcţionalităţile descrise în continuare. Se cere utilizarea unei structuri NOD care are două câmpuri: un câmp int pentru informaţie şi un câmp de tip pointer la NOD pentru legătura către următorul element. Se cere utilizarea unei structuri LISTA care are ca menbru un pointer la NOD reprezentând capul listei şi metodele:

   - INSERT - inserează un element în capul listei (0.25 p)

- SEARCH - caută o cheie k în listă (0.25 p)
- DELETE -şterge un element din listă, (0.25 p)
- ISEMPTY - verifică dacă lista e vidă (0.25 p)
- INVERSE - inversează lista în compelxitate $O(n)$, fără a folosi mai mult decât o zonă de memorie constantă, cu excepţia celei ocupate de listă (adică fără a construi o a două listă) (1 p)
- INSERT_AFTER - inserează o cheie k după un nod cu o anumită cheie m dată, sau la capătul listei, dacă m nu există în listă. (0.5 p)
- INSERT_N_POZ - inserează de n ori cheia k pe poziţia p, dacă lista are cel puţin p elemente, sau la sfârşitul listei altfel. (1p)
- PRINT_LIST - afişează elementele din listă (0.25 p)

Utilizaţi un constructor pentru a iniţializa capul listei cu NULL. (0.25p)

În funcţia *main* realizaţi un menu cu ajutorul unei instrucţiuni *switch*, prin care se oferă 8 opţiuni, fiecare corespunzătoare uneia dintre funcţiile de mai sus, precum şi o opţiune de EXIT. Într-o instrucţiune *while*, se citesc şi se execută opţiuni până la alegerea opţiunii de EXIT.

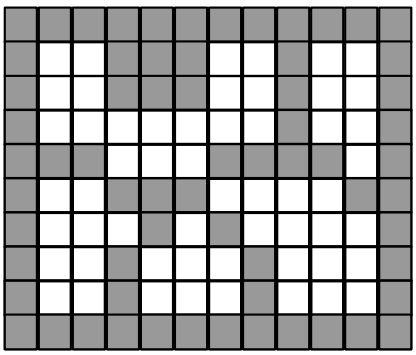

Figure 1: Reprezentarea etajului unei clădiri

6. Un anumit etaj al unei clădiri este reprezentat schematic sub forma unei matrice ce conţine valorile -1 şi 0, unde -1 reprezintă zid şi 0 reprezintă spaţiu liber. Pereţii sunt de grosime 1 şi uşile nu sunt marcate (se consideră tot perete).

   a. Determinaţi numărul de încăperi ale etajului respectiv. (1p)

   b. Determinaţi încăperea cu suprafaţa maxima. (1p)

c. Care perete poate fi dărâmat (o poziţie maractă cu -1 se va marca cu 0) a. î. să se obţină o încăpere de suprafaţă maximă? (2p)

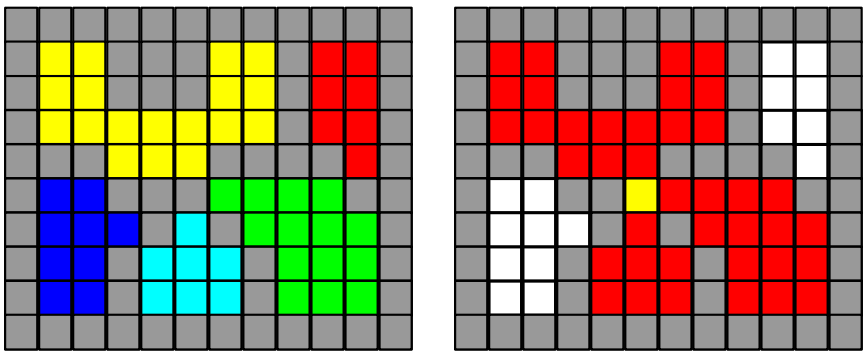

Figure 2: a. Camerele colorate cu culori diferite. b. Dacă se şterge elementul de zid marcat cu galben se obţine cea mai mare cameră posibilă.

În verificarea apartenenţei la aceeaşi cameră în matrice se consideră doar vecinătăţile pe orizontală şi verticală.

Nu utilizaţi funţii recursive!

Exemplu: Se consideră etajul reprezentat de către matricea din figură 1. Este o matrice de dimensiuni 10×12 în care pereţii sunt coloraţi cu gri (deci valorile -1) şi 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 suprafaţa de 18 unităţi, camera roşie de 7 unităţi, camera albastră de 9 unităţi, camera cyan de 7 unităţi şi camera verde de 14 unităţi.

În figura 2b. se observă faptul că, dacă se elimină zidul de pe poziţia marcată cu galben, se unesc trei încăperi şi se obţine ce mai mare încăpere posibilă, cu suprafaţa de 40 de unităţi.

7. Se consideră un labirint reprezentat printr-o matrice în care zidurile sunt marcate prin -1 şi drumurile prin 0. În labirint se află un şoricel pe poziţia (x_0, y_0) şi o bucată de brânză pe poziţia (x_1, y_1). Să se găsească un drum (de preferinţă de lungime minimă) de la şoricel la brânză. Nu utilizaţi recursivitate! Utilizaţi o coadă. (3p)

Evaluare: Rezolvaţi 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 învăţare.