Examen Structuri de Date, 13 iunie 2015

NUME:

PRENUME:

GRUPA:

| | Ex1 | Ex2 | Ex3 | Ex4 |
|---|---|---|---|---|
| a | | | | |
| b | | | | |
| c | | | | |
| d | | | | |

*Punctajele se acorda astfel: a=1p, b=2p, c=3p, d=4p, total 4x(1+2+3+4) = 40p*
*Punctajul pentru d se acorda doar daca ati rezolvat corect punctul c corespunzator*

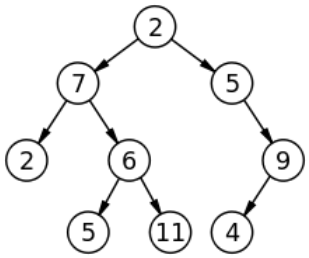
1. **Lista si Coada**
   a. Care sunt tipurile de liste inlantuite studiate la curs? Dati un exemplu pentru fiecare si explicati ce pointeri ati folosi pentru implementarea fiecareia.
   b. Se da urmatoarea functie pentru a parcurge o lista simplu inlantuita:

```
void traverse(struct Node *head)
{
   while (head->next != NULL)
   {
       printf("%d  ", head->data);
       head = head->next;
   }
}
```

   Care raspuns / raspunsuri sunt adevarate si de ce?
   - Functia poate esua cand lista este goala
   - Functia nu afiseaza ultimul nod al listei
   - Functia nu este implementata corect pentru ca modifica pointerul head

   c. Scrieti o functie pentru inserarea de elemente intr-o lista simplu inlantuita circulara.
   d. Modificati functia de la punctul precedent pentru ca lista sa se mentina sortata. Considerand ca aveti clasa SortedLinkedList pentru lista simplu inlantuita circulara cu aceasta functie pentru inserare, implementati clasa PriorityQueue folosind drept container de date un obiect SortedLinkedList.

2. **Arbori Binari si Arbori Binari de Cautare**
   a. Ce este un arbore? Dar un arbore binar?
   b. Intr-un arbore binar, nivelul unui nod este dat de numarul de muchii care il leaga de radacina. Astfel, radacina este pe nivelul 0 si copii sai pe nivelul 1. Care este numarul maxim de noduri de pe nivelul i? De ce?
   c. Fie structura de date BinaryTree, implementati functiile de parcurgere in preordine, inordine si postordine pentru aceasta structura de date. Ce va afisa fiecare din aceste functii pentru arborele din figura?



d. Pornind de la implementarea punctului precedent, scrieti si explicati o functie care sa determine daca arborele este un arbore binar de cautare.

**3. Heap**

a. Explicati cum functioneaza un heap si la ce poate fi folosit
b. Fie un sir de elemente, cu urmatoarele chei: 13, 10, 7, 23, 9, 18, 8, 16, 25, 4, 17. Explicati procesul de adaugare a acestor elemente in heap (prin desen si cateva explicatii)
c. Implementati o clasa MinHeap care sa ofere posibilitatea inserarii si extragerii de elemente
d. Folositi clasa MinHeap de la punctul anterior pentru a gasi al k-lea cel mai mare element (pentru punctaj maxim, ganditi o solutie in care sa nu trebuiasca sa introduceti toate elementele in heap)

**4. Graf**

a. Ce puteti spune despre graful de mai jos? Este orientat sau neorientat? Cate componente conexe are si care sunt acestea?

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 4 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

```
void functia1(int v)
{
    bool *visited = new bool[V];
    for(int i = 0; i < V; i++)
        visited[i] = false;

    functia2(v, visited);
}

void functia2(int v, bool visited[])
{
    visited[v] = true;
    cout << v << " ";

    list<int>::iterator i;
    for(i = adj[v].begin(); i != adj[v].end(); ++i)
        if(!visited[*i])
            functia2(*i, visited);
}
```

b. Ce face codul de mai sus? Explicati pe scurt functionarea sa. Daca graful de mai sus ar fi stocat sub forma de lista de vecini, ce rezultat ar produce functia?
c. Implementati o functie care sa realizeze parcurgerea BFS. Ce rezultat va produce pentru graful din figura de mai sus?
d. Modificati functia de parcurgere BFS de la punctul precedent pentru a masura distanta de la un nod catre nodurile din componenta sa conexa. Folositi aceasta functie pentru a determina care este cea mai mare distanta dintre oricare doua noduri din graf.