Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
Öğr.Gör. Ömer ÇAKIR

BIL 205 Veri Yapıları
Bütünleme Sınavı, 31.01.2013, 13:00, D1
Süre : **75** Dakika

CEVAPLAR

```
void print(DoublyNode* node)
{
   if (node == trailer) return;
   else print(node->next);
   cout << node->elem << endl;
}

void main()
{
   DoublyLinkedList list;

   list.insertOrdered("Paul", 720);
   list.insertOrdered("Rose", 590);
   list.insertOrdered("Anna", 660);

   list.print(list.header->next);
}
```

1. **a)** Yukarıdaki programın çıktısı nedir? **(10P)**

Paul
Anna
Rose

b) **if(node==trailer)** kısmı **if(node->next==trailer)** şeklinde olsaydı çıktı ne olurdu? Sebebini açıklayınız. **(30P)**

ÇIKTI

Anna
Rose

print(node->next) recursive çağrıları node==trailer olana dek yapılır. node == trailer olunca return yapıldığında son satıra gelinir cout ile listenin son elemanı olan Paul yazılır. Çünkü son çağrıda node=Paul idi. Onun next'i de trailerdır. Diğer elemanlar da benzeri şekilde sondan başa yazılır.

node->next==trailer olduğunda recursive çağrılar sonucu ilk satır return yaparken node bu sefer Anna'dır. Çünkü onun next'i Paul if'in için node olarak gider ve onun da next'i trailer olduğundan return yapılır. Sondan başa listelendiğinden Anna'nın peşine de Rose yazılır.

7 5 4 3 2 1 6

2. Yukarıdaki verileri splay ağacına yerleştiriniz? **(25P)**

Bu sorunun cevabı 2. Sayfadadır |
v

```
void removeOrdered(const string& e, const int& i)
{
  SinglyNode* current        = head;
  SinglyNode* previous       = head;

  if((current->elem == e) && (current->score == i) )
  {
     head = current->next;
     delete current;
     return;
  }

  current = current->next;
  while (current != NULL)
  {
     if((current->elem == e)&&(current->score == i))
     {
        previous->next = current->next;
        delete current;
        return;
     }

     previous = current;
     current  = current->next;
  }

  if(current==NULL) cout<<e<<" is not found"<<endl;
}
```

3. Tek yönlü bağlı listeden eleman silen **removeOrdered()** adlı fonksiyondaki boş satırlara gerekli kodları yazınız. **(15P)**

4. Kuyruk veri yapısına ait enqueue() ve dequeue() fonksiyonları, dairesel bağlı listenin add(), advance() ve remove() fonksiyonları ile gerçeklenirken dequeue() için remove() yeterli olurken enqueue() için neden add()'in peşine advance() işlemi de yapmak gerekir? **(20P)**

(Bu sorunun benzeri vizede sorulmuştu)

Dairesel bağlı listede cursor listenin sonuna (back) işaret ediyordu. Hem ekleme (add) hem de silme (remove) işlemi cursor->next ile yapılıyordu. Yani cursor'ın peşine ekleniyor veya peşine gelen siliniyordu. Bu bilgiler ışığında :

Kuyruk veri yapısında veri ekleme (enqueue) kuyruk sonuna yapıldığından dairesel bağlı listenin add fonksiyonunun peşine, cursor yeni eklenene son eleman olarak işaret etsin diye advance de çağrılmıştır.

Kuyruk veri yapısında veri silme (dequeue) kuyruk başından yapıldığından dairesel bağlı listenin remove fonksiyonu ile birebir örtüşür. Çünkü dairesel listede de cursor->next yani ilk eleman silinmektedir.

2. Sorunun Cevabı :