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

BIL 205 Veri Yapıları
1. Arasınav, 23.11.2012, 15:00, D-2, D-9
Süre : **100** Dakika

		DEĞERLENDİRME	
NUMARA :	**AD SOYAD :**		
Sınavda Uyulması Gereken Kurallar	**İMZA :**	[.....]	.....

1. Cep telefonlarının saate bakmak için bile olsa herhangi bir amaçla kullanılması yasaktır. Telefon kapalı ve cepte olmalıdır.
2. Sınavın başında sorular kısaca açıklanacaktır. Öğrencilerin soruları cevaplandıktan sonra sınav boyunca soru sormak yasaktır.
3. Soru kağıdına numaranızı ve isminizi yazıp imzalamayı unutmayınız.

```
void insertOrdered(const string& e, const int& i)
{
  DoublyNode* newNode  = new DoublyNode;
  newNode->elem        = e;    newNode->score = i;
  newNode->next        = NULL; newNode->prev  = NULL;

  if(header->next == trailer)
  {//BLOCK 1
    newNode->next      = trailer;
    newNode->prev      = header;
    header->next       = newNode;
    trailer->prev      = newNode;
    return;
  }

  DoublyNode* current  = header->next;
  if(newNode->score < current->score)
  {//BLOCK 2
    newNode->next      = current;
    newNode->prev      = current->prev;
    current->prev      = newNode;
    current->prev->next= newNode;
    return;
  }

  DoublyNode* founded  = NULL;
  while (current != trailer)
  {
    if(newNode->score>=current->score) founded=current;
    else break;
    current = current->next;
  }
  //BLOCK 3
  newNode->next        = founded->next;
  newNode->prev        = founded;
  founded->next        = newNode;
  founded->next->prev  = newNode;
}
```

1. Elemanları çift yönlü bağlı listeye sıralı ekleyen yukarıdaki `insertOrdered()` adlı fonksiyonda **koyu** yazılmış kod bloklarının birinde veya birkaçında satırların yerleri değiştirildiği için program koştuğunda hata vermektedir. Bu blok veya blokların doğrusunu aşağıya yazınız. **(20P)**

BLOCK 1'in doğrusu (blokta hata yoksa birşey yazmayınız) :
BLOCK 2'nin doğrusu (blokta hata yoksa birşey yazmayınız) :
BLOCK 3'ün doğrusu (blokta hata yoksa birşey yazmayınız) :

```
CircularlyLinkedList C;

void CircularlyLinkedQueue::enqueue(const string& e)
{       // Kodu buraya yazınız !



        n++;
}

void CircularlyLinkedQueue::dequeue()
{
        if (empty())
        {
                cout << "dequeue of empty queue" << endl;
                return;
        }
        // Kodu buraya yazınız !



        n--;
}
```

2. Kuyruk veri yapısına ait `enqueue()` ve `dequeue()` fonksiyonlarını, C dairesel bağlı listesinin `add()`, `advance()` ve `remove()` fonksiyonları ile gerçeklemek üzere yukarıya gerekli kodları yazınız. Yazdığınız kodları açıklayınız. **(20P)**

`enqueue()`fonksiyonunun C dairesel bağlı listesi ile gerçeklenişi
`dequeue()`fonksiyonunun C dairesel bağlı listesi ile gerçeklenişi

```
int tripleSum(int A[], int i, int n)
{
   if (n == 1) return A[i];
   else
   {
       int Sum =     tripleSum(A, i + 2*n/3, n/3 )
                   + tripleSum(A, i + n/3, n/3 )
                   + tripleSum(A, i, n/3 );
       cout << "Sum = " << Sum << endl;
       return Sum;
   }
}

void main()
{
   int A[27] = { 1,2,3,4,5,6,7,8,9,
                 10,11,12,13,14,15,16,17,18,
                 19,20,21,22,23,24,25,26,27 };

   tripleSum(A, 0, 27);
}
```

3. Yukarıdaki programın çıktısı nedir? **(30P)**

İpucu→ Program ekrana **13** kere `Sum = ...` yazar.
[1..27] arası sayıların toplamı **378**'dir.
Toplarken işlem hatası yapmamaya dikkat ediniz !

Sum =
Sum =
Sum =
Sum =
Sum =
Sum =
Sum =
Sum =
Sum =
Sum =
Sum =
Sum =
Sum =

```
void rList(DoublyNode* hNext, DoublyNode* tPrev)
{
   if ( hNext == tPrev ) return;

   if(hNext->next == tPrev)
   {
      hNext->next = tPrev->next;
      tPrev->next->prev = hNext;

      tPrev->prev = hNext->prev;
      hNext->prev->next = tPrev;

      hNext->prev = tPrev;
      tPrev->next = hNext;

      return;
   }
   else
   {
      DoublyNode* hNextNext = hNext->next;
      DoublyNode* hNextPrev = hNext->prev;

      hNext->next = tPrev->next;
      hNext->prev = tPrev->prev;
      tPrev->prev->next = hNext;
      tPrev->next->prev = hNext;

      tPrev->next = hNextNext;
      tPrev->prev = hNextPrev;
      hNextPrev->next = tPrev;
      hNextNext->prev = tPrev;

      return rList( tPrev->next, hNext->prev);
   }
}
```

4.

a) Yukarıda verilen **rList()** fonksiyonu main() fonksiyonunda **rList(header->next, trailer->prev)** parametreleri ile çağrıldığında ne iş yapar? **(15P)**

b) **rList()** fonksiyonu kendini recursive olarak neden **rList(tPrev->next, hNext->prev)** parametreleri ile çağırmıştır? **(15P)**