Karadeniz Technical University
Department of Computer Engineering
Lecturer Omer CAKIR

COM 205 Data Structures
Midterm Exam, 11.23.2012, 15:00, D-1, D-8
Time : **100** Minutes

NUMBER : ..... **NAME :**	**EXAM GRADE**	
Rules to be Obeyed During the Exam **SIGNATURE:**	[.....]	.....

1. Cell phones are not allowed to be used as a calculator or a watch. They must be switched off and placed in the pocket.
2. Brief information about the exam will be given at the begining, then no one is not allowed to ask a question during the exam.
3. Do not to forget to sign this paper after writing your number and name.

```
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. `insertOrdered()` that insert elements with ascending order into a doubly linked list gives an error because line sequence of one or more code block written in **bold** is changed. Write down the correct sequence of this or these blocks. **(20P)**

BLOCK 1 CORRECT SEQUENCE (write nothing if already correct):
BLOCK 2 CORRECT SEQUENCE (write nothing if already correct):
BLOCK 3 CORRECT SEQUENCE (write nothing if already correct):

```
CircularlyLinkedList C;

void CircularlyLinkedQueue::enqueue(const string& e)
{       // Write code here !



        n++;
}

void CircularlyLinkedQueue::dequeue()
{
        if (empty())
        {
                cout << "dequeue of empty queue" << endl;
                return;
        }
        // Write code here !



        n--;
}
```

2. Complete `enqueue()` and `dequeue()` functions belong to queue data structure with `add()`, `advance()` and `remove()` functions of C circularly linked list. Explain your code. **(20P)**

`enqueue()` implementation using C circularly linked list functions :
`dequeue()` implementation using C circularly linked list functions :

```
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. Write down the out of the program above?(30P)

Hint → Program prints out **Sum = ... , 13** times.
Sum of the numbers **[1..27]** is **378**.
Be careful not to sum numbers incorrectly!

Sum =
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) What does **rList()** function above if it is called like **rList(header->next, trailer->prev)** in the main() function? **(15P)**

b) Why **rList()** function is recursively called like **rList(tPrev->next, hNext->prev)**? **(15P)**