Karadeniz Technical University
Department of Computer Engineering
Lecturer Ömer ÇAKIR

COM 1004 Data Structures
Resit Exam, 22.01.2016, 10:00, D-1
Exam Duration : **90** Minutes

NUMBER : ..... **NAME :** .....

Rules to be Obeyed During the Exam **SIGNATURE :** .....

EXAM GRADE	
[.....]	.....

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 reverseList(DoublyLinkedList* list,
                 DoublyNode* hNext,
                 DoublyNode* tPrev)
{
  if (hNext == tPrev) return;

  if (hNext->next == tPrev)
  {
    list->add(hNext, tPrev->elem, tPrev->score);
    list->remove(tPrev);
    return;
  }
  else
  {
    list->add(hNext, tPrev->elem, tPrev->score);
    tPrev = tPrev->prev;
    list->remove(tPrev->next);

    list->add( ....., hNext->elem, hNext->score);
    hNext = hNext->next;
    list->remove(hNext->prev);

    reverseList(list, hNext, ..... );
  }
}

void main()
{
  DoublyLinkedList* list = new DoublyLinkedList();
  list->insertOrdered("Paul",  720);
  list->insertOrdered("Rose",  590);
  list->insertOrdered("Anna",  660);
  list->insertOrdered("Mike", 1105);
  list->insertOrdered("Rob",   750);
  list->insertOrdered("Jack",  510);
  list->insertOrdered("Jill",  740);

  reverseList(list,
              list->header->next,
              list->trailer->prev);
  list->printH2T();
}
```

1. Which two of the following choises for the **.....** lines of the function **reverseList()** have the same outputs when the function **printH2T()** is called? What is the output? **(40P)**

(A)
```
tPrev
tPrev
```

(B)
```
tPrev->next
tPrev->next
```

(C)
```
tPrev->next
tPrev->prev
```

(D)
```
tPrev->next
tPrev
```

(E)
```
tPrev
tPrev->prev
```

(F)
```
tPrev->prev
tPrev
```

Equivalent choises → ( ) and ( )

Output :


```
void LinkedBinaryTree::traverse(Node* p)
{
   while (root != NULL)
   {
      while (p->left != NULL) p = p->left;
      cout << p->elt << endl;
      deleteNode(root, p->elt);
      p = root;
   }
}

void main()
{                                        // Output:
   LinkedBinaryTree Tree;
   Tree.addRoot();
   Tree.root->elt = 8;
   Tree.addBelowRoot(Tree.root,  4);
   Tree.addBelowRoot(Tree.root, 12);
   Tree.addBelowRoot(Tree.root,  2);
   Tree.addBelowRoot(Tree.root,  6);
   Tree.addBelowRoot(Tree.root, 10);
   Tree.addBelowRoot(Tree.root, 14);
   Tree.addBelowRoot(Tree.root,  1);
   Tree.addBelowRoot(Tree.root,  3);
   Tree.addBelowRoot(Tree.root,  5);
   Tree.addBelowRoot(Tree.root,  7);
   Tree.addBelowRoot(Tree.root,  9);
   Tree.addBelowRoot(Tree.root, 11);
   Tree.addBelowRoot(Tree.root, 13);
   Tree.addBelowRoot(Tree.root, 15);

   binaryTree.traverse(binaryTree.root);
}
```

2. **a)** What is the output of the program above? **(20P)**
Note → Assume that the deleted node is replaced with the next higher one.

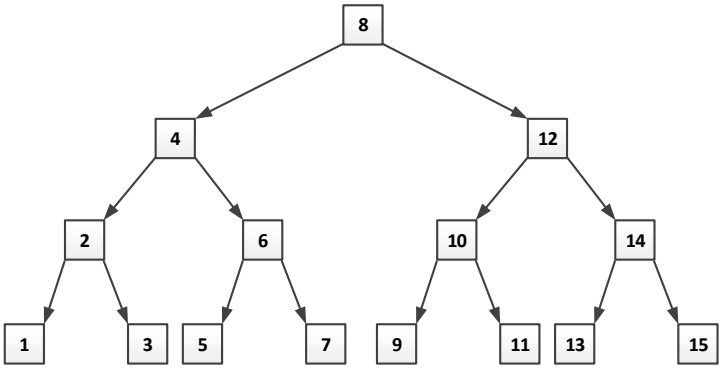
b) Which tree traversal method is the output of the program equivalent to? **(20P)**

*You'll loose **5P** from wrong answer.*

(A) inorder

(B) preorder

(C) postorder




```
void insertOrdered(string& e, int& i)
{
   CircularlyNode* newNode = new CircularlyNode;
   newNode->elem           = e;
   newNode->score          = i;

   if (cursor == NULL)
   {
      newNode->next  = newNode;
      cursor         = newNode;
      return;
   }

   CircularlyNode* front   = cursor->next;
   CircularlyNode* back    = cursor;

   while( newNode->score > front->score )
   {
      back = front;
      front = front->next;

      if (.......................) break;
   }

   back->next        = newNode;
   newNode->next     = front;

   if (newNode->score > cursor->score)
       cursor = cursor->next;
}
```

3. Write **R** for "Right" or **W** for "Wrong" before the codes below for the **.....** line of the function **insertOrdered()** that inserts nodes in ascending order into a <u>circularly</u> linked list by **score** value.
(20P)

*You will loose **5P** from <u>each</u> wrong answer.*

(...) back == cursor

(...) back == cursor->next

(...) front == cursor

(...) front == cursor->next