Karadeniz Technical University
Department of Computer Engineering
Lecturer Ömer ÇAKIR

COM 1004 Data Structures
Final Exam, 03.01.2016, 10:00, D-1, D-8
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);

  cout << "Reversed List :" << endl;

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

  list->printH2T();
}
```

1. Which of the following choises is the code of the `.....` lines of the function **`reverseList()`** that reverses the elements of a doubly linked list? **(30P)**

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

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

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

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

(D) `tPrev`
`tPrev->prev`

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

```
void LinkedBinaryTree::traverse(Node* p)
{
  while (root != NULL)
  {
    while ((p->left != NULL) || (p->right != NULL))
    {
      if (p->left != NULL)
        p = p->left;
      else
        p = p->right;
    }

    cout << p->elt << endl;
    deleteNode(root, p->elt);

    p = root;
  }
}

void main()                                // Output →
{
  LinkedBinaryTree binaryTree;

  binaryTree.addRoot();
  binaryTree.root->elt = 8;
  binaryTree.addBelowRoot(binaryTree.root, 4);
  binaryTree.addBelowRoot(binaryTree.root, 12);
  binaryTree.addBelowRoot(binaryTree.root, 2);
  binaryTree.addBelowRoot(binaryTree.root, 6);
  binaryTree.addBelowRoot(binaryTree.root, 10);
  binaryTree.addBelowRoot(binaryTree.root, 14);
  binaryTree.addBelowRoot(binaryTree.root, 1);
  binaryTree.addBelowRoot(binaryTree.root, 3);
  binaryTree.addBelowRoot(binaryTree.root, 5);
  binaryTree.addBelowRoot(binaryTree.root, 7);
  binaryTree.addBelowRoot(binaryTree.root, 9);
  binaryTree.addBelowRoot(binaryTree.root, 11);
  binaryTree.addBelowRoot(binaryTree.root, 13);
  binaryTree.addBelowRoot(binaryTree.root, 15);

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

2. **a)** What is the output of the program above? **(20P)**

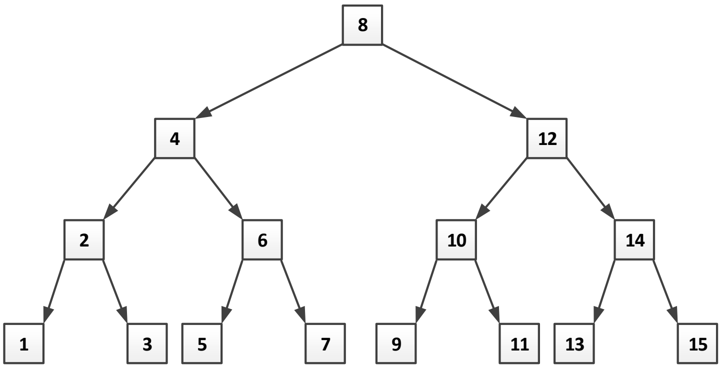
b) Which tree traversal method is the output of the program equivalent to? **(15P)**

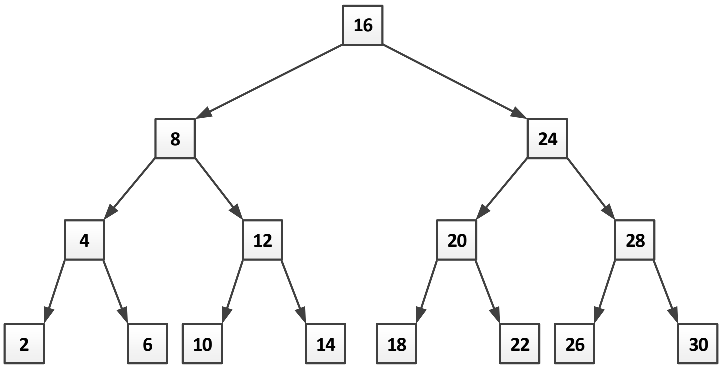
*You'll loose **5P** from wrong answer.*

(A) inorder

(B) preorder

(C) postorder





3. Delete **16** from the **2-3-4** tree above. **(35P)**