



CEVAPLAR - A

```
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. Liste elemanlarını reverse yapan `reverseList()` fonksiyonunda ile temsil edilen yerlere ait kodlar sırasıyla aşağıdakilerden hangisidir? (30P)

Yanlış cevaptan 5P kılacaktır.

- (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()
{
    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);
}

```

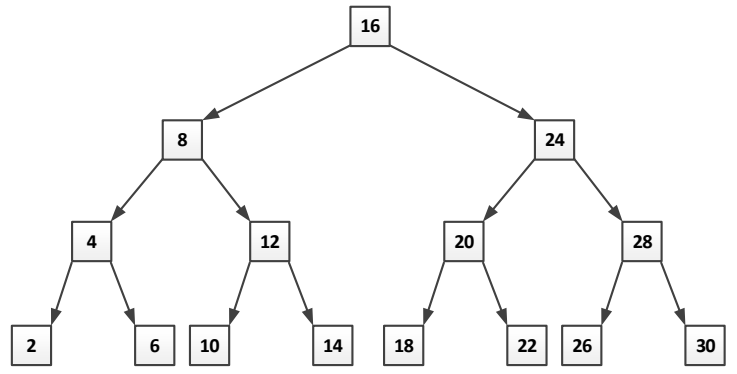
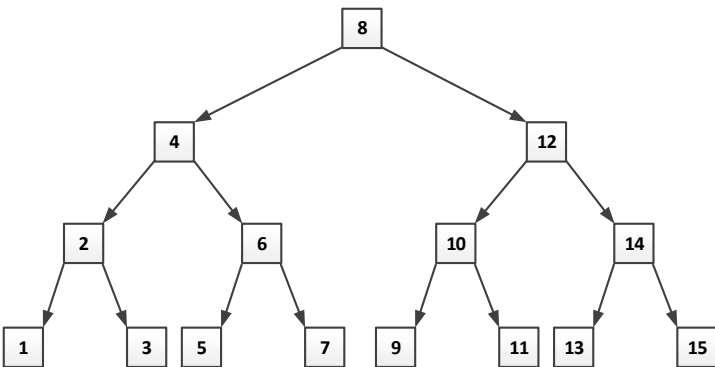
// Çıktı →

- 1
- 3
- 2
- 5
- 7
- 6
- 4
- 9
- 11
- 10
- 13
- 15
- 14
- 12
- 8

2. a) Yukarıdaki programın çıktısı nedir? (20P)

b) Yukarıdaki programın çıktısı hangi ağaç gezinme yöntemine eşdeğerdir? (15P) *Yanlış cevaptan 5P kırılacaktır.*

- (A) inorder
- (B) preorder
- (C) postorder



3. Yukarıdaki 2-3-4 ağacından 16'yı siliniz. (35P)

(35P)

