



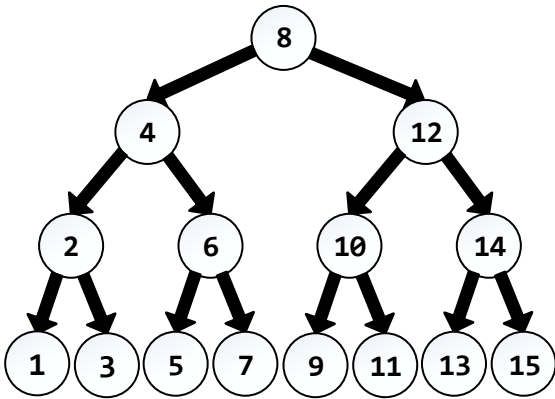
NUMARA : AD SOYAD :	DEĞERLENDİRME	
	[.....]
Öğrenciler, Mühendislik Fakültesi Sınav Uygulama Yönergesi 'ndeki kurallara uymalıdır. Sınav Soruları Bölüm Program Çıktıları 'ndan 1,4,12 ile ilişkilidir.		

```
void traverse(TreeNode* v)
{
    if (v->left != NULL)
    {
        traverse(v->left);
        cout << v->elem << " ";
    }

    if (v->right != NULL)
    {
        traverse(v->right);
    }
}
```

1. main()'de aşağıdaki ağacın **root**u ile çağrıldığı varsayılan **traverse()** fonksiyonunun çıktısı nedir? (25P)

--	--	--	--	--	--	--	--



1 2 3 4 5 6 7 8

2. Yukarıdaki verilerin ikili ağaca eklendiği varsayılınsın. Bu ağacın inorder, preorder ve postorder gezinme çıktılarından hangisi diğer ikisinden farklıdır? (25P)
Yanlış cevaptan 5P kılacaktır.

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

```

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

    DoublyNode* current      = header;

    while (current->next != trailer)
    {
        if (newNode->score >= current->next->score)
            current = current->next;
        else
            break;
    }

    newNode->next             = current->next;
    newNode->prev             = current;
    .....                   = .....;
    .....                   = .....;
}

```

3. insertOrdered() fonksiyonundaki satırları için aşağıda verilen kodlardan hangisi listeye hatalı ekleme yapar? (25P) *Yanlış cevaptan 5P kılacaktır.*

- (A) newNode->prev->next = newNode;
newNode->next->prev = newNode;
- (B) newNode->next->prev = newNode;
newNode->prev->next = newNode;
- (C) current->next->prev = newNode;
current->next = newNode;
- (D) current->next->prev = newNode;
newNode->prev->next = newNode;
- (E) newNode->prev->next = newNode;
current->next->prev = newNode;

```

SinglyLinkedList* mergeLists(SinglyLinkedList*
                             list2)
{
    SinglyLinkedList* mergedList =
        new SinglyLinkedList();
    SinglyNode* plist1 = this->head;
    SinglyNode* plist2 = list2->head;

    while ((plist1 != NULL) || (plist2 != NULL))
    {
        if (plist1 == NULL)
        {
            mergedList->addBack(plist2->elem,
                                plist2->score);
            plist2 = plist2->next;    continue;
        }

        if (plist2 == NULL)
        {
            mergedList->addBack(plist1->elem,
                                plist1->score);
            plist1 = plist1->next;    continue;
        }

        if (plist1->score <= plist2->score)
        {
            .....
            .....
        }
        else
        {
            .....
            .....
        }
    }
    return mergedList;
}

```

4. Yukarıdaki mergeLists() fonksiyonunu tamamlayınız. (25P)