Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
Öğr.Gör. Ömer ÇAKIR

BIL 2001 Veri Yapıları
Arasınav, 20.11.2018, 15:00
Süre : **100** Dakika

CEVAPLAR

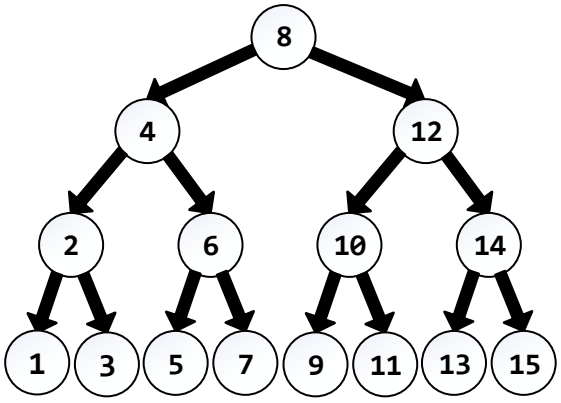
```
void ZigZag(TreeNode* v)
{
	cout << v->elem << " ";

	if (v->left != NULL)
	{
		ZigZag(v->left);
	}

	if (v->right != NULL)
	{
		ZigZag(v->right);
		cout << v->elem << " ";
	}
}
```

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

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




```
void traverse(Node* v)
{
   stack<Node*> stl_stack;

   stl_stack.push(v);

   while (!stl_stack.empty())
   {
      Node* current = stl_stack.top();

      if ((current->right != NULL)
         || (current->left != NULL))
         cout << current->elem << " ";

      stl_stack.pop();

      if (current->right != NULL)
            stl_stack.push(current->right);

      if (current->left != NULL)
            stl_stack.push(current->left);
   }
}
```

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

8	4	2	6	12	10	14

```
void insertOrdered(DoublyNode* newNode,
                   DoublyNode* current)
{
  if(..................... || ..................)
  {
     newNode->next          = current->next;
     newNode->prev          = current;
     current->next->prev    = newNode;
     current->next          = newNode;
  }
  else
    insertOrdered(newNode, current->next);
}

int main()
{
  DoublyLinkedList list; DoublyNode* newNode;

  newNode = new DoublyNode;
  newNode->elem  = "Paul";  newNode->score = 720;
  list.insertOrdered(newNode, list.header);

  newNode = new DoublyNode;
  newNode->elem = "Rose";   newNode->score = 590;
  list.insertOrdered(newNode, list.header);

  newNode = new DoublyNode;
  newNode->elem = "Anna";   newNode->score = 660;
  list.insertOrdered(newNode, list.header);

  newNode = new DoublyNode;
  newNode->elem = "Mike";   newNode->score =1105;
  list.insertOrdered(newNode, list.header);
}
```

3. `insertOrdered()` fonksiyonunu tamamlayınız. **(25P)**
Not → `header` ve **`trailer`**'ın score değerini **`0`** varsayınız.
*Yanlış cevaptan **5P** kırılacaktır.*

```
(A)  if ((current == trailer)
     || (newNode->score <= current->score))

(B)  if ((current->next == trailer)
     || (newNode->score <= current->score))

(C)  if ((current == trailer)
     || (newNode->score <= current->next->score))

(D)  if ((current->next == trailer)
     || (newNode->score <= current->next->score))
```

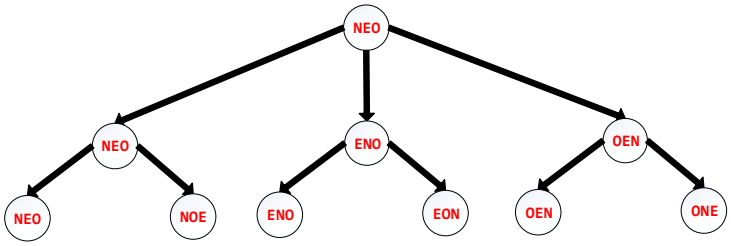
```
void strings(string str, int i, int n)
{
      if (i == n - 1)
      {
            cout << str << endl;
            return;
      }

      for (int j = i; j < n; j++)
      {
            swap(str[i], str[j]);
            strings(str, i + 1, n);
            swap(str[i], str[j]);
      }
}

void main()
{
      string str = "NEO";
      strings(str, 0, str.length());
}
```

4. Yukarıdaki programın çıktısı nedir? **(25P)**

```
NEO
NOE
ENO
EON
OEN
ONE
```



İlan edilen programın olduğu klasörde "**solution.cpp**" adlı bir dosya var. Aslında bu bir kod dosyası değil. Recursive çağrıların detaylı anlatıldığı bir belge olarak düşünebilirsiniz.