Karadeniz Technical University
Department of Computer Engineering
Lecturer Ömer ÇAKIR

COM 2005 Data Structures
Midterm Exam, 20.11.2018, 15:00
Duration : **100** Minutes

NUMBER : .... **NAME :** .... **SIGNATURE :**	**EXAM GRADE**	
	[........]	....

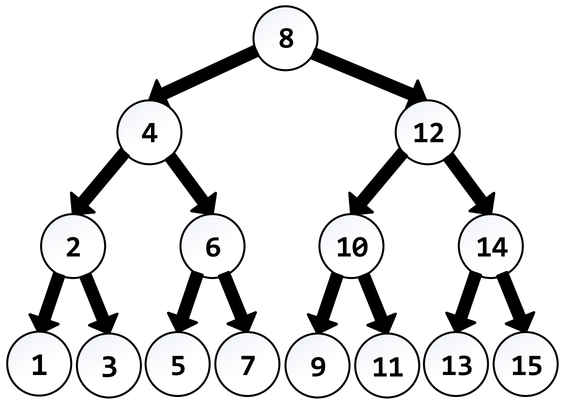
Students have to obey Engineering Faculty Exam Execution Instructions.
Questions are related to 1,4,12 of Program Learning Outcomes

```
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. What is the output of the function **ZigZag()** that is called with the **root** of the tree below as the argument? **(25P)**

→

8										4
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. What is the output of the function **traverse()** that is called with the **root** of the tree on the left as the argument? **(25P)**


```
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. Complete the function **insertOrdered()**. **(25P)**
Assume that **header**'s and **trailer**'s scores are **0**.

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

```
(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. What is the output of the program above? **(25P)**