Karadeniz Technical University
Department of Computer Engineering
Lecturer Ömer ÇAKIR

COM 2005 Data Structures
Resit Exam, 26.01.2018, 15:00, D-1
Duration : **90** Minutes

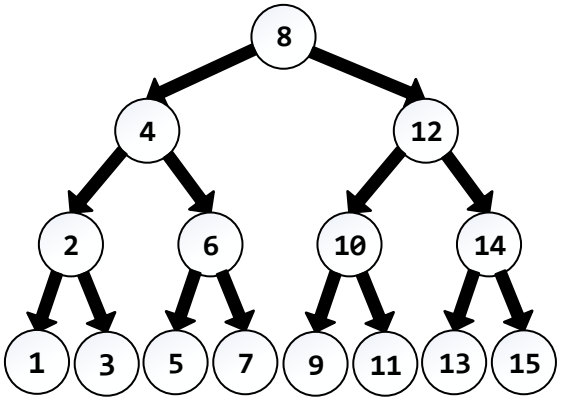
NUMBER : ..... NAME : ..... SIGNATURE :	EXAM GRADE	
	[.....]	.....

Enginnering Faculty Exam Execution Instructions should be obeyed. Questions are related to 1,4,12 of Program Learning Outcomes

```
void traverse(Node* v)
{
	if (v->left != NULL)
	{
		traverse(v->left);
	}

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

1. What is the output of the function **traverse()** that is called in the **main()** with the **root** of the tree below? **(25P)**




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

   while (true)
   {
       if (current != NULL)
       {
              stl_stack.push(current);
              current = current->left;
       }
       else
       {
              if (stl_stack.empty())
              {
                  return;
              }
              else
              {
                  current = stl_stack.top();

                  if ((current->right != NULL)
                     && (current->left != NULL))
                     cout << current->elt << " ";

                  stl_stack.pop();
                  current = current->right;
              }
       }
   }
}
```

2. What is the output of the function **traverse()** that is called in the **main()** with the **root** of the tree on the left? **(25P)**

```
void insertOrdered(DoublyNode* newNode,
                   DoublyNode* current)
{
  if(..................... && ..................)

        insertOrdered(newNode, current->next);
  else
  {
        newNode->next  = current;
        newNode->prev  = current->prev;
        current->prev->next = newNode;
        current->prev  = newNode;
  }
}

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 score are **0**.
*You'll loose **5P** from wrong answer.*

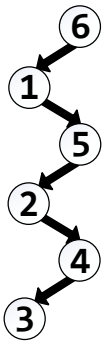
```
(A)  if ((newNode->score >= current->score)
        && (current != trailer))

(B)  if ((newNode->score >= current->next->score)
        && (current != trailer))

(C)  if ((newNode->score >= current->score)
        && (current->next != trailer))

(D)  if ((newNode->score >= current->next->score)
        && (current->next != trailer))
```

Zig
Zig-Zig
Zig-Zig
Zig
Zig-Zig
Zig-Zig
Zig



4. Find the element insertion order for the Splay Tree above using splay operations in the table. **(25P)**

→

					6