Karadeniz Technical University
Department of Computer Engineering
Lecturer Ömer ÇAKIR

COM 2005 Data Structures
Final Exam, 02.01.2018, 13:00, D-1
Duration : **90** Minutes

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

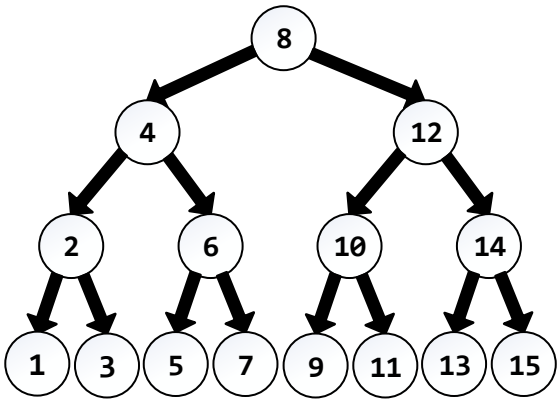
Enginnering Faculty Exam Execution Instructions should be obeyed. Questions are related to 1,4,12 of Program Learning Outcomes

```
void print(DoublyNode* node, bool first)
{
	if (first)
		cout << node->elem << endl;
	if (node->next != trailer)
		print(node->next, false);
	else
		cout << node->elem << endl;
}

int main()
{
	DoublyLinkedList list;
	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);
	list.print(list.header->next, true);
}
```

1. What's the output of the program above? **(25P)**

```
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->elt << " ";

      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 in the **main()** with the **root** of the tree below? **(25P)**



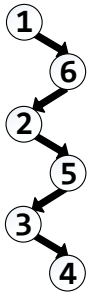
```
void insertOrdered(DoublyNode* newNode,
                   DoublyNode* current)
{
  if(..................... && .................)

        insertOrdered(newNode, current->next);
  else
  {
        newNode->next = current->next;
        newNode->prev = current;
        current->next->prev = newNode;
        current->next = 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 **`Trailer`**'s score is **`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)**