Karadeniz Technical University
Department of Computer Engineering
Lecturer Ömer ÇAKIR

COM 2005 Data Structures
Final Exam, 09.01.2026, 08:30
Duration : **75** Minutes

NUMBER : ..... NAME : ..... SIGNATURE :	EVALUATION 1[.....] 2[.....] 3[.....] 4[.....] SUM[.....]

Faculty of Engineering Exam Directives must be followed. Questions are related to 1,4,12 of the Program Learning Outcomes

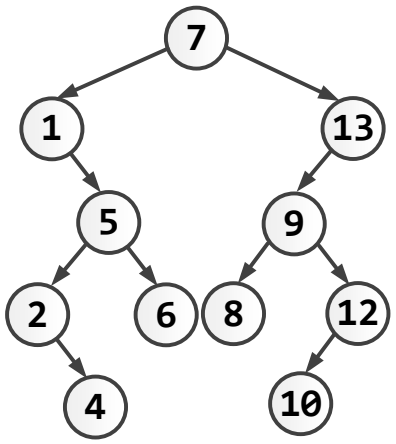
```
void removeOrdered(string& e, int& i,
                           DoublyNode* current)
{
  if (empty())
  {
     cout << "List is empty !" << endl;
     return;
  }

  if (current != trailer)
  {
     if ((e.compare(current->elem) == 0)
                && (current->score == i))
     {
         current->prev->next = current->next;
         current->next->prev = current->prev;
         delete current;
         return;
     }

     removeOrdered(e, i, current->next);
  }
  else
     cout << e << " is not found" << 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);

    cout << "List after insertions :" << endl;
    list.printH2T();

    list.removeOrdered("Jack", 510,
                           list.header->next);
    list.removeOrdered("Mike", 1105,
                           list.header->next);
    list.removeOrdered("Paul", 720,
                           list.header->next);

    cout << "\nList after removals:" << endl;
    list.printH2T();
}
```

1. How many times does the function **removeOrdered()** call itself recursively when removing **Jack, Mike and Paul**? **(30P)**

```
void BinaryTree::eulerLike(TreeNode* v) const
{
	if (v->left != NULL)
	{
		eulerLike(v->left);
	}

	cout << v->elem;

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

int main()
{
	BinaryTree binaryTree;
	binaryTree.addNode(binaryTree.root, 7);
	binaryTree.addNode(binaryTree.root, 3);
	binaryTree.addNode(binaryTree.root, 11);
	binaryTree.addNode(binaryTree.root, 1);
	binaryTree.addNode(binaryTree.root, 5);
	binaryTree.addNode(binaryTree.root, 9);
	binaryTree.addNode(binaryTree.root, 13);

	binaryTree.eulerLike(binaryTree.root);
}
```

2. What is the output of the program above? **(20P)**

3. How many times does the function **eulerLike()** call itself recursively? **(20P)**



4. Add **14** to the Splay Tree above. **(30P)**