SOLUTIONS

```
void CircularlyLinkedList::
          newAdd(const string& e, const int& i)
{
  CircularlyNode* newNode= new CircularlyNode;
  newNode->elem          = e;
  newNode->score         = i;

  if (cursor == NULL)
  {
     newNode->next  = newNode;
     cursor         = newNode;
     return;
  }

  CircularlyNode* back = cursor;

  while (newNode->score < back->next->score)
  {
     if (back->next == cursor)
     {
        newNode->next     = back->next->next;
        back->next->next  = newNode;
        cursor            = cursor->next;
        return;
     }
     back = back->next;
  }

  newNode->next    = back->next;
  back->next       = newNode;
}

int main()
{
  CircularlyLinkedList list;

  list.newAdd("Paul", 720);
  list.newAdd("Rose", 590);
  list.newAdd("Anna", 660);
  list.newAdd("Mike", 1105);
  list.newAdd("Rob",  750);
  list.newAdd("Jack", 510);
  list.newAdd("Jill", 740);

  list.print(); // prints cursor->next first !
}
```

1. What is the output of the program above? **(30P)**

Mike	1105
Rob	750
Jill	740
Paul	720
Anna	660
Rose	590
Jack	510

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

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

	cout << v->elem;
}

int main()
{
	BinaryTree binaryTree;
	binaryTree.addNode(binaryTree.root, 4);
	binaryTree.addNode(binaryTree.root, 2);
	binaryTree.addNode(binaryTree.root, 6);
	binaryTree.addNode(binaryTree.root, 1);
	binaryTree.addNode(binaryTree.root, 3);
	binaryTree.addNode(binaryTree.root, 5);
	binaryTree.addNode(binaryTree.root, 7);

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

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

4	2	1	2	3	2	4	6	5	6	7	6	4

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

A)4 *B)6* C)8 D)10 E)12

```
SinglyLinkedList SinglyLinkedList::
uniLists(SinglyLinkedList list2)
{
   SinglyLinkedList tekList;
   SinglyNode* plist1 = head;
   SinglyNode* plist2 = list2.head;

   while ((plist1 != NULL) || (plist2 != NULL))
   {
      if (plist1 == NULL)
      {
         tekList.addBack(plist2->score);
         plist2 = plist2->next; continue;
      }

      if (plist2 == NULL)
      {
         tekList.addBack(plist1->score);
         plist1 = plist1->next; continue;
      }

      if (plist1->score < plist2->score)
      {
         tekList.addBack(plist1->score);
         plist1 = plist1->next;
      }

      if (plist2->score < plist1->score)
      {
         tekList.addBack(plist2->score);
         plist2 = plist2->next;
      }

      if (plist1->score == plist2->score)
      {
         plist1 = plist1->next;
         plist2 = plist2->next;
      }
   }
   return tekList;
}

int main()
{
   SinglyLinkedList list1;
   list1.addBack(1);
   list1.addBack(2);
   list1.addBack(3);
   list1.addBack(4);
   list1.addBack(5);

   SinglyLinkedList list2;
   list2.addBack(1);
   list2.addBack(3);
   list2.addBack(5);
   list2.addBack(7);
   list2.addBack(9);

   SinglyLinkedList tekList =
                        list1.uniLists(list2);
   tekList.print();
}
```

4. What is the output of the program above? **(30P)**

2
4
7
9