CSC212

Tutorial 7

Recursion

Problem 1:

Write the static recursive method ***reverseQueue*** that changes the order of the elements in Queue ***q*** and puts them in reverse order. Don’t use auxiliary data structures.

The function’s signature: *public static <T> void reverseQueue(Queue<T> q)*

Solution:

```
public static <T> void reverseQueue(Queue<T> q){
        if(q.length() == 0)
                return;
        T x = q.serve();
        reverseQueue(q);
        q.enqueue(x);
}
```

Problem 2:

Write the recursive static method ***copyStack***, that takes two Stacks ***s1*** and ***s2*** and copies all the elements in ***s1*** into ***s2*** in the same order. Don’t use auxiliary data structures. ***s1*** should not change at the end of the method.

The function’s signature: *public static <T> void copyStack(Stack<T> s1, Stack<T> s2)*

Solution:

```
public static <T> void copyStack(Stack<T> s1, Stack<T> s2){
        if(s1.empty())
                return;
        T x = s1.pop();
        copyStack(s1, s2);
        s1.push(x);
        s2.push(x);
}
```

Problem 3:

Write the recursive method ***search,*** member of the class ***Linkedlist,*** that searches for an element ***e*** and returns true if found or false otherwise. Don’t use auxiliary data structures and don’t call any of the ***LinkedList*** methods.

The function’s signature: *public Boolean search(T e)*

Solution:

```
public boolean search(T e) {
        return recSearch(head, e);
}
private boolean recSearch(Node<T> p, T e){
        if(p == null)
                return false;
        if(p.data.equals(e))
                return true;
        return recSearch(p.next, e);
}
```

Problem 4:

Write the static recursive method ***searchList*** that searches for an element ***e*** in a List ***l*** and returns true if found or false otherwise. Don’t use auxiliary data structures.

The function’s signature: *public static <T> boolean searchList(List<T> l, T e)*

Solution:

```
public static <T> boolean searchList(List<T> l, T e){
        if(l.empty())
                return false;
        l.findFirst();
        return recSearch(l, e);
}
private static <T> boolean recSearch(List<T> l, T e){
        if(l.last())
                return l.retrive().equals(e);
        if(l.retrive().equals(e))
                return true;
        l.findNext();
        return recSearch(l, e);
}
```