QUESTION 1

**MYSingleLinkedList<T>{**

**Iterator<T> iterator();**

**public class MyIterator <T> implements Iterator<T>**{///ITERATOR SUBCLASS.

//HAS next() hasnext() remove() METHODS }.

**public String reverseToString(**){/CALLS REVERSER.

**private String reverser(String str)** /////RECURSIVE METHOD TO REVERSE A STRING.

**private node<T> reUse(Object data){///**REUSE’S THE DELETED NODES FOR NEW NODES.

**private void delete(Object del){///**ADDS REMOVED NODES TO ANOTHER LINKED LIST.

**public String deletedToString(){///”**toSTRING()” METHOD FOR DELETED STRING LIST

IF WE REUSE A NODE,IT WILL NO LONGER BE USED IN THIS METHOD.

**public void add(Object data){//**APPENDS THE NEW ELEMENT TO THE LIST

**T get(int index)/**/RETURNS THE DATA OF THE NODE BY INDEX NUMBER

**void remove**(int index)///REMOVES NODES BY IT’S INDEX NUMBER.CALLS “delete(Object del)” METHOD TO DELETE

**void remove(T element)//** REMOVES NODES BY IT’S NAME.

BOTH OF ABOVE REMOVE METHODS ITERATE THROUGH THE LIST FROM THE BEGİNNİNG.

ITERATOR’S REMOVE METHOD IS FASTER.

**private class node<T>{///NODE SUBCLASS OF SINGLE LINKED LIST.**

T data;

node<T> next;

node(Object o){

data=(T) o;

**}**

**We create one this class each time we make append a new node.**

**public String toString()///OVERIDDEN “TOSTRING()” METHOD.WORKS LIKE AN ITERATOR.THEORETİCALY SAME SPEED.**

**{**

QUESTION 3

**public AbstractCollection <E> appendAnything(AbstractCollection <E> base,AbstractCollection <E> ekle){**

I HAVE USED POLYMORPHİSM .EVERY LIST IS INHERITED FROM AbstractCollection CLASS.

AbstractCollection IS INHERITED FROM ITERABLE INTERFACE.SO I CAN USE ITERATOR TO PARSE THROUGH EVERY TYPE OF LIST.

USING “**NEXT()”** AND “**HASNEXT()” AND ADD()** METHODS I WAS ABLE MAKE A VERY SHORT CLASS TO CONCATANATE TWO LIST.SAME ELEMENTS WON’T BE CONCANATED.**ADD()** METHODS RETURN

QUESTION 4

I HAVE CREATED A NEW LINKED LIST FOR DELETED NODES.SO WHEN I REMOVE A NODE FROM THE NORMAL LINKED LIST ,IT GET’S TRANSFERRED TO THIS NEW LIST.

DELETE() METHOD IS FOR THAT.

REUSE() METHOD USES THESE NODES IN THIS LIST.IF NOTHING LEFTS WE JUST RETURN NULL SO

OUR ADD() METHOD JUST CREATES NEW ONES.

DeletedToString() METHOD RETURNS A STRING OF ELEMENTS OF THIS LIST.BUT WHEN WE REUSE THESE NODES THEY WONT BE USED TO MAKE A STRING.

I HAVE MADE A MAIN CLASS TO TEST THESE.