PNG UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

CS220—Programming IV

SEMESTER 2 EXAMINATION

DATE: 26TH OCTOBER 2022

TIME ALLOWED: 3 HOURS

INSTRUCTIONS

1. Write your name and student number clearly on the front of the answer booklet.
2. There are eight (8) questions in the Examination Booklet.
3. You have 10 minutes to read this paper. Please **do not** write during this time.
4. All answers must be written in the examination answer booklet.
5. Attempt All Questions.
6. Notes, mobile phones, textbooks and dictionaries are NOT allowed in this exam.
7. Use blue or black ink—not pencil or red ink.
8. **Do not write** anything or **turn this page** until you are told to do so.

Marking Scheme

Questions	**Marks**
Question 1	/8
Question 2	/16
Question 3	/10
Question 4	/10
Question 5	/15
Question 6	/17
Question 7	/11
Question 8	/13
Total	**/100**

Question 1: [2 + 6 = 8 marks]

a) What is unit test in software development?

b) The *max* method given below returns a maximum number from a list of numbers. Create a test class called `TestClass.java` and write one test case (junit test) for the method.

```java
public int max(int[] list) {
    if(list.length==0) {
        return 0;
    }
    int max=list[0];
    for(int n:list) {
        if(n>max) {
            max=n;
        }
    }
    return max;
}
```

Question 2: [10 + 6 = 16 marks]

For this question refer to the following code. The class `C.java` has a generic max method that returns maximum element from a 2-dimensional array. Note that the *`p.compareTo(q)`* method returns -1 if `p<q`. It returns 1 if `p>q`

```java
import java.util.ArrayList;

public class C{

    public static <E extends Comparable<E>> E max(E[][] list) {
        E max=list[0][0];
        for(int i=0;i<list.length;i++) {
            for(int j=0;j<list[i].length;j++) {
                E e=list[i][j];
                if(max.compareTo(e)<0) {
                    max=e;
                }
            }
        }
        return max;
    }
}
```

Below is the JUnit Test class for the class C.java given.

```
import static org.junit.Assert.*;

public class UnitTest {
@Test
public void testMaxNumbers() {
        Integer[][] x= {{1,3,10,0},{9,8,2,7}};
        Integer ex=new Integer(10);
        assertEquals(ex,C.max(x)); //1
        assertEquals(C.max(x),ex); //2
        assertFalse(C.max(x)>0);   //3
        assertTrue(0==C.max(x));   //4
        assertNotEquals(C.max(x),new Integer(5)); //5

    }
}
```

a) For each assert statement given in the test class, state whether it will pass or fail. Also state whether each assert statement is valid or not. If not valid then correct it.

b) The assertions given in the test method *testMaxNumbers* is for testing array of numbers. Since the method is generic, you can also pass in 2-dimensional array of any reference types. Now, write a test method called *testMaxString*. Define 2-dimensional String array in the method and initialize it with some values and write one test case using assert statements.

Question 3:[2 + 2 + 6 = 10 marks]

For this question refer to the 3D array given.

```
int[][][] myNumbers={

                        {{5, 10,15,20,25}, {2, 4, 6, 8,10}},
                        {{44,55,66,77}, {400, 100,-100},{-120,90,800},{},{500}},
                        {{200, 100,-200}},
                        {{36, 80, 90, 200,20}, {100, 30}},
                        {{290, 30, 60,100},{100,-90, 110}}

                };
```

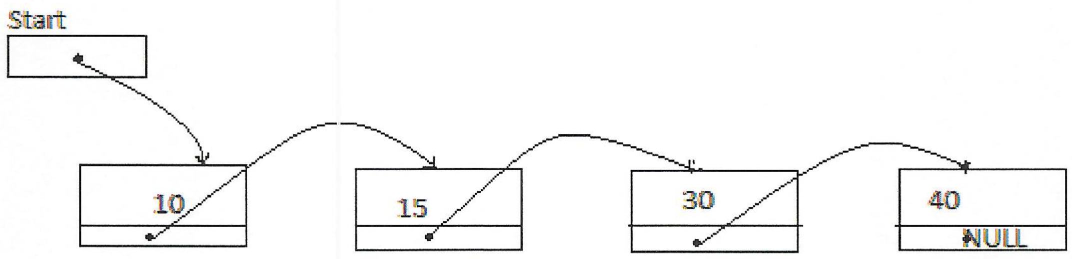
a) What is the length of `myNumbers[1]`?

b) What is the length of `myNumbers[1][3]`?

c) Write a code fragment to compute the sum of integers in myNumbers[1] that are 100 and above.

Question 3:[4 + 6 = 10 marks]

a) Here is a linked list containing integers in sorted fashion.



Explain the steps involved is inserting a new node with value 20 using a block diagram.

b) For this question, refer to program in ***Appendix A***. Implement the `addToLast()` method in line 52 of the program given.

Question 5: [3 + 2 + 5 + 5 = 15 marks]

a) What are the three parts to GUI programming in Java?

b) What is a container object in GUI programming in Java?

c) Create a frame in Java with following properties and then create a panel and add to the frame. Then create a button with a caption "OK" and add it to the panel.

Size:200 pixels by 200 pixels
Title: "New Form",
Visibility: The frame should be visible on the screen

d) Create a button with a label "MouseClick" and add listener to it.

Question 6: [3 + 2 + 4 + 5 + 3 = 17 marks]

a) State the characteristics of a recursive method.

b) Explain the difference between a recursion and iteration.

c) Write a recursive definition for computing the function $X^n + n$ for n≥0.

d) Write a recursive method for the recursive definition in (c) and identify the base case.

e) What is wrong with the following program?

```
public class Test {
  public static void main(String[] args) {
    computeD(1234567);
  }

  public static void computeD(double n) {
   if (n != 0) {
      System.out.println(n);
      computeD(n / 10);
    }
  }
}
```

Question 7: [2 + 3 + 3 + 3 = 11 marks]

a) What is the key benefit of using generics in Java?

b) What is erasure? Why Java generics are implemented using erasure?

c) What is wrong in the following code?

```
ArrayList dates = new ArrayList();
dates.add(new Date());
Date date = dates.get(0);
```

d) What is a bounded type in Java Generics?

Question 8: [3 + 3 + 3 + 4 = 13 marks]

For this question refer to **Appendix B**. In this question you are expected to write generic methods to adding soft drinks to a carton. The respective classes that represent different soft drinks, a generic class called `Softdrink<T>`, a `Carton` class and the `DrinkDemo` class are given in Appendix B. Study the program carefully and answer the following questions.

a) In the `Carton` class, write a method signature to add any type of soft drink to the carton.

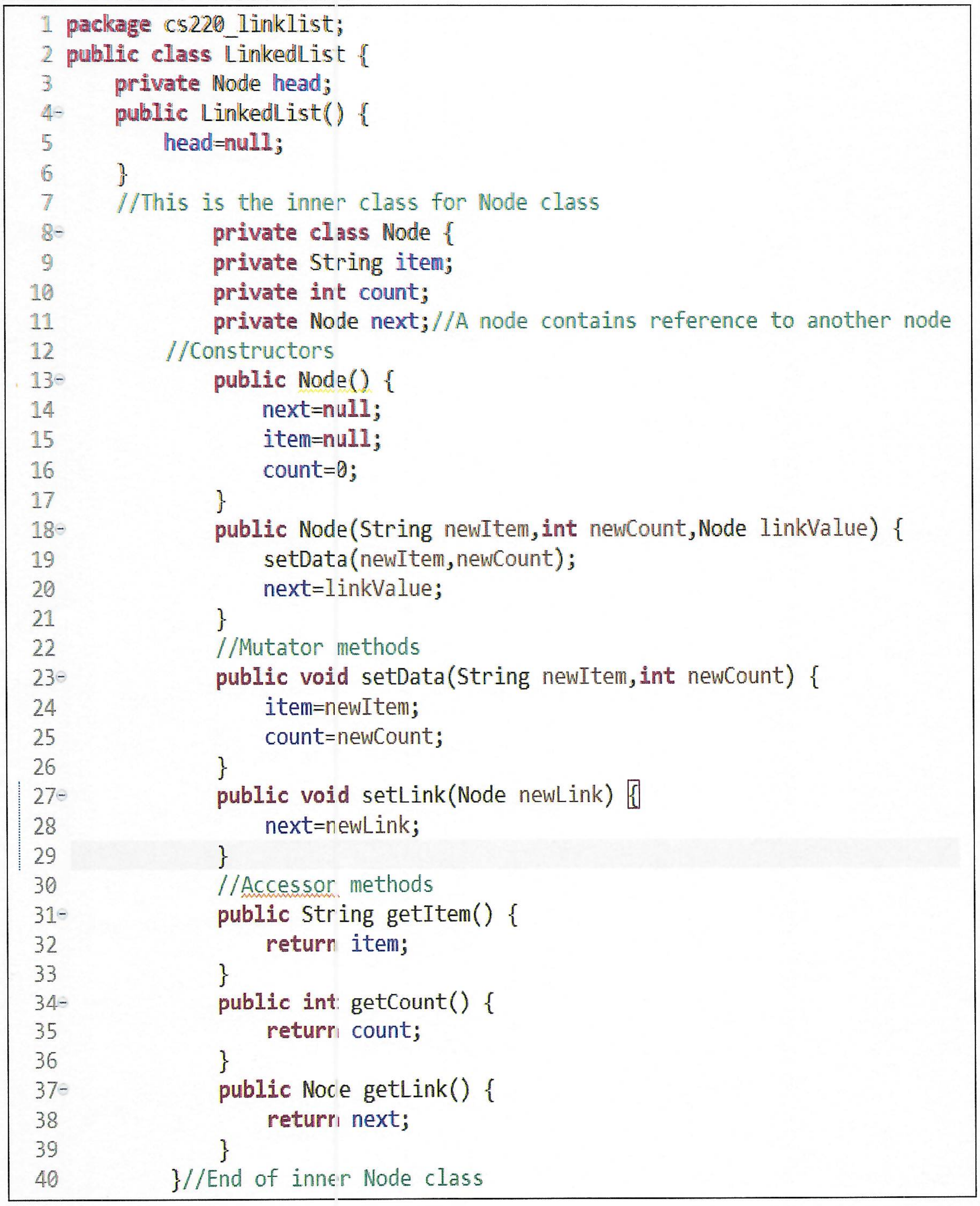
b) In the `Carton` class, write a method signature to add soft drink of type `Coke` only to the carton.

c) In the `Carton` class, write a method signature to add soft drink of type `CokeZero` as lower bound to the carton.

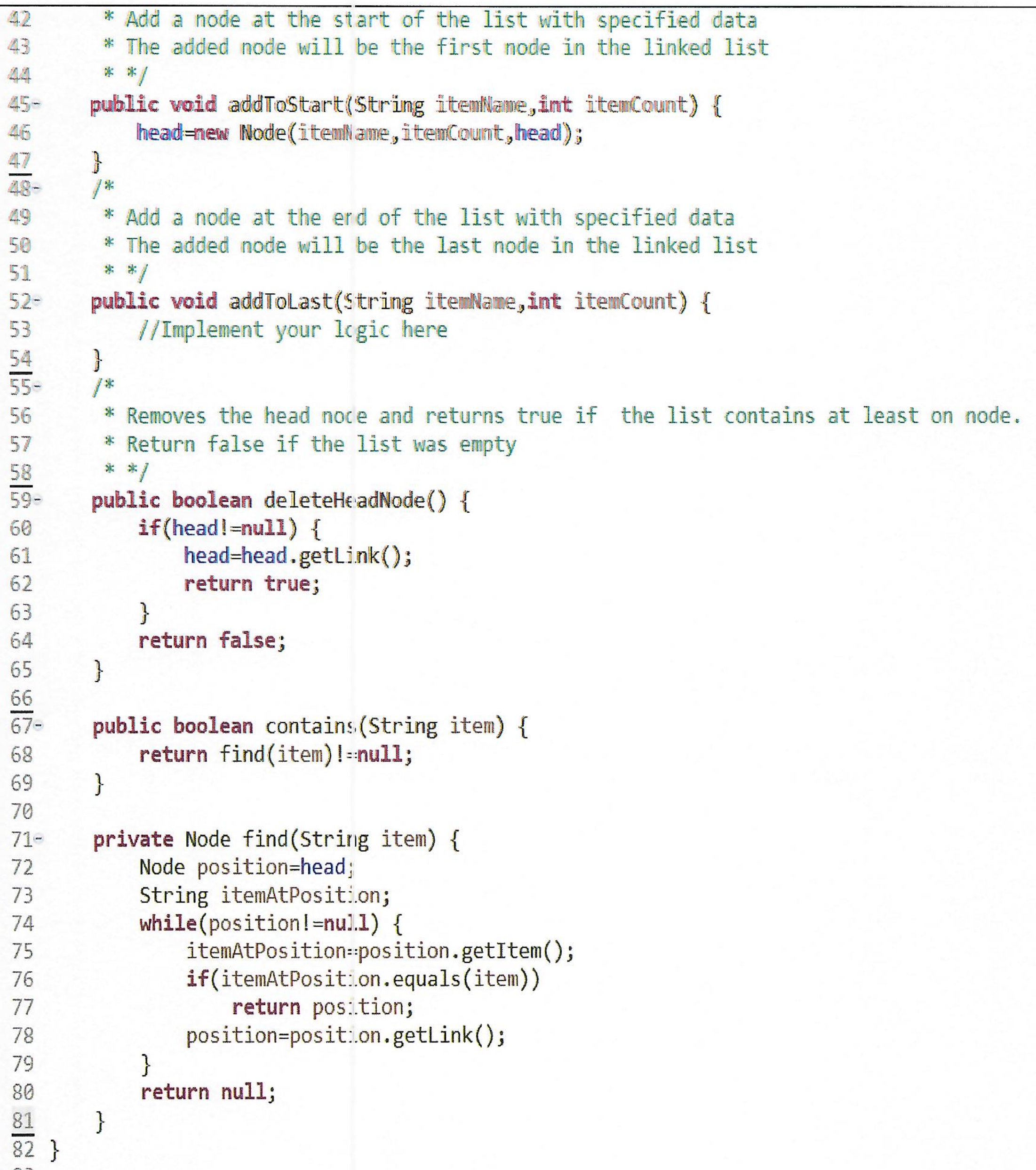
d) When you pass an instance of generic type `Softdrink<PepsiCola>` as an argument in the method in (b), it will give an error. However, if you pass the instance in (a), there will be no error. Explain why the error happens.

....**END OF EXAMINATION**..

Appendix A: Linkedlist Datastructure



```java
package cs220_linklist;
public class LinkedList {
    private Node head;
    public LinkedList() {
        head=null;
    }
    //This is the inner class for Node class
            private class Node {
            private String item;
            private int count;
            private Node next;//A node contains reference to another node
        //Constructors
            public Node() {
                next=null;
                item=null;
                count=0;
            }
            public Node(String newItem,int newCount,Node linkValue) {
                setData(newItem,newCount);
                next=linkValue;
            }
            //Mutator methods
            public void setData(String newItem,int newCount) {
                item=newItem;
                count=newCount;
            }
            public void setLink(Node newLink) {
                next=newLink;
            }
            //Accessor methods
            public String getItem() {
                return item;
            }
            public int getCount() {
                return count;
            }
            public Node getLink() {
                return next;
            }
        }//End of inner Node class
```

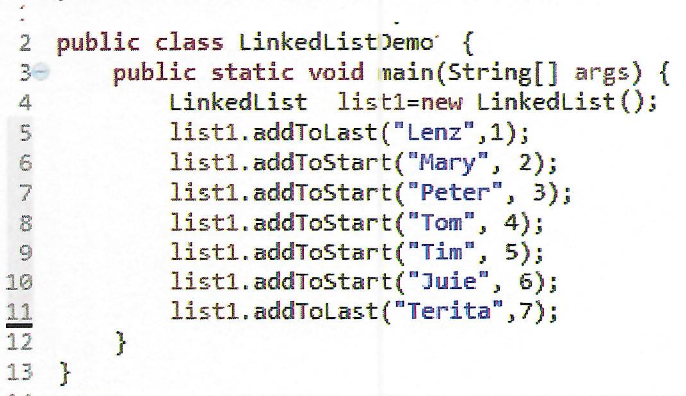


```java
     * Add a node at the start of the list with specified data
     * The added node will be the first node in the linked list
     * */
    public void addToStart(String itemName,int itemCount) {
        head=new Node(itemName,itemCount,head);
    }
    /*
     * Add a node at the end of the list with specified data
     * The added node will be the last node in the linked list
     * */
    public void addToLast(String itemName,int itemCount) {
        //Implement your logic here
    }
    /*
     * Removes the head node and returns true if  the list contains at least on node.
     * Return false if the list was empty
     * */
    public boolean deleteHeadNode() {
        if(head!=null) {
            head=head.getLink();
            return true;
        }
        return false;
    }

    public boolean contains(String item) {
        return find(item)!=null;
    }

    private Node find(String item) {
        Node position=head;
        String itemAtPosition;
        while(position!=null) {
            itemAtPosition=position.getItem();
            if(itemAtPosition.equals(item))
                return position;
            position=position.getLink();
        }
        return null;
    }
}
```

Usage of the *addToLast()* method in main class.



```java
public class LinkedListDemo  {
    public static void main(String[] args) {
        LinkedList  list1=new LinkedList();
        list1.addToLast("Lenz",1);
        list1.addToStart("Mary", 2);
        list1.addToStart("Peter", 3);
        list1.addToStart("Tom", 4);
        list1.addToStart("Tim", 5);
        list1.addToStart("Juie", 6);
        list1.addToLast("Terita",7);
    }
}
```

Appendix B: Generics Program

```java
public class Softdrink<T> {
    public T softdrink;
}
public class Fanta {}
public class Pepsi {}
public class PepsiCola extends Pepsi{}
public class Coke {}
public class CokeDiet extends Coke{}
public class CokeZero extends Coke{}
public class Carton {
    //1) Write a method signature to add any type of soft drink to carton
    //2) Write a method signature to add soft drink of type Coke only
    //3) Write a method to add software drink of type CokeZero as lower bound
}
public class DrinkDemo {
    public static void main(String[] args) {
        Softdrink<Pepsi> pepsiDrink=new Softdrink<>();
        Pepsi pepsi=new Pepsi();
        pepsiDrink.softdrink=pepsi;

        Softdrink<Coke> cokeDrink=new Softdrink<>();
        Coke coke=new Coke();
        cokeDrink.softdrink=coke;

        Softdrink<CokeZero> cokeZeroDrink=new Softdrink<>();
        CokeZero cokeZero=new CokeZero();
        cokeZeroDrink.softdrink=cokeZero;

        Softdrink<CokeDiet> cokeDietDrink=new Softdrink<>();
        CokeDiet cokeDiet=new CokeDiet();
        cokeDietDrink.softdrink=cokeDiet;

        Carton carton=new Carton();

        //Methods for adding drinks to Carton
        //Example.
        // carton.addAnyDrink(cokeDrink);
        // carton.addAnyDrink(pepsiDrink);
        // carton.addCokeDrink(cokeZeroDrink);
        // carton.addCokeZeroDrink(cokeDrink);
    }
}
```