Angela Zhang
amz2132
Professor Blaer
COMS 3134 Data Structures

Homework 2

1.
```
public static void printLots(List<Object> L,List<Integer> P)
{
        Iterator<Integer> itr2 = P.iterator();

        for (int x : P)
        {
                Object s = "";
                int count = itr2.next();
                Iterator<Object> itr1 = L.iterator();

                for (int i=0; i<count+1; i++)
                {
                        if (itr1.hasNext())
                        {
                                s= itr1.next();
                        }
                        else
                        {
                                s="That index is out of bounds";
                        }
                }
                System.out.println(s);
        }
}
}
```

2. (Problem 4 from Weiss)

```
public List intersection(List L1, List L2,)
{
        Iterator itr1 = L1.iterator();
        List<Object> L3 = new LinkedList<Object>();

        while (itr1.hasNext())
        {
                Iterator itr2 = L2.iterator();
                Object target = itr1.next();

                while (itr2.hasNext())
                {
                        if itr2.next().equals(target);
                        {
                                L3.add(target);
                        }
                }
            }
            return L3;
        }
```

3. (Problem 24 from Weiss)

```
public class Written3<AnyType>
{
        int top1;
        int top2;
        AnyType[] array;

        Written3(int n)
        {
        array = (AnyType[]) new Object[n];

        top1=-1;
        top2=array.length;
        }

        public boolean empty1()
        {
                if (top1==-1)
                {
                        return true;
                }
                else
                {
                        return false;
                }
        }

        public boolean empty2()
        {
                if(top2==array.length)
                {
                        return true;
                }
                else
                {
                        return false;
                }
        }

        public AnyType peek1()
        {
                return array[top1];
        }

        public AnyType peek2()
```

```java
{
        return array[top2];
}

public AnyType pop1()
{
        if (top1 > -1)
        {
                AnyType popped= array[top1];
                top1--;
                return popped;
        }
        else
        {
                System.out.println("Error: Cannot pop empty stack");
                return null;
        }
}

public AnyType pop2()
{
        if (top2<array.length)
        {
                AnyType popped = array[top2];
                top2--;
                return popped;
        }
        else
        {
                System.out.println("Error: Cannot pop empty stack");
                return null;
        }
}
public void push1(AnyType a)
{
        if (top2-top1>1)
        {
                top1++;
                array[top1]=a;
        }
        else
        {
                System.out.println("StackOverflow Error");
        }
}

public void push2(AnyType a)
{
        if(top2-top1>1)
        {
                top2--;
                array[top2]=a;
        }
```

```
            else
            {
                System.out.println("StackOverflow Error");
            }
        }
}
```

4.a) 1. Move 4 from the front of the input track to holding track s1 (output track: blank)
2. Move 3 from the front of the input track to holding track s1 (output track: blank)
3. Move 1 from the front of the input track to the output track (output track: 1)
4. Move 8 from the front of the input track to holding track s2 (output track: 1)
5. Move 2 from the front of the input track to the output track (output track: 21)
6. Move 7 from the front of the input track to holding track s2
7. Move 6 from the front of the input track to holding track s2
8. Move 9 from the front of the input track to holding track s3
9. Move 5 from the front of the input track to holding track s2
10. Move 3 from holding track s1 to the output track (output track: 321)
11. Move 4 from holding track s1 to the output track (output track: 4321)
12. Move 5 from holding track s2 to the output track (output track: 54321)
13. Move 6 from holding track s2 to the output track (output track: 654321)
14. Move 7 from holding track s2 to the output track (output track: 7654321)
15. Move 8 from holding track s2 to the output track (output track: 87654321)
16. Move 9 from holding track s1 to the output track (output track: 987654321)

b) An example would be 135798642