Peter Drake (pjd2132)
Discrete Math

Homework 2

Written 1

```
public class Problem1{
   public static void printLots(List<Object> L, List<Integer> P){
      Iterator<Object> itrL = L.iterator();
      Iterator<Integer> itrP = P.iterator();

      if (itrP.hasNext()){
         int position = itrP.next();

         int i = 0;

         while (itrL.hasNext()){
            if (i == position){
               System.out.println(itrL.next());
               if (itrP.hasNext()){
                  position = itrP.next();
               }
               else{
                  break;
               }
            }
            else{
               itrL.next();
            }
            i++;
         }
      }
   }
```

Written 2

```
public static list intersection(list L1, list L2){

        pos1 = L1.next();               //pos1 assigned to first value in L1
        pos2 = L2.next();               //pos2 assigned to first value in L2
        list result;


        while(pos1 != null && pos2 != null){

                if (pos1 < pos2){
                        pos1 = pos1.next();
                }
                else if (pos1 > pos2){
                        pos2 = pos2.next();
                }
                else{
                        result.add(pos1);
                        pos1 = L1.next();
                        pos2 = L2.next();
                }
        }
        return result;
```

Note: this code takes advantage of the fact that the two lists are sorted.

Written 3

```
public class TwoStacksArray {
        int top1;
        int top2;
        int size;
        AnyType[] arr;

        public TwoStacksArray(int n){
                size = n;
                arr = new AnyType[n];
                top1 = 0;
                top2 = n – 1;

        public void push1(AnyType x){
                if (top1 <= top2){
                        arr[top1] = x;
                        top1++;
                }
                else{
                        print(“Overflow”);
                        throw new OverflowException;
                }

        public void push2(AnyType x){
                if (top1 <= top2){
                        arr[top2] = x;
                        top2--;
                }
                else{
                        print(“Overflow”);
                        throw new OverflowException;
                }

        public AnyType pop1(){
                if (top1 > 0){
                        AnyType temp = arr[top1 – 1];
                        top1--;
                        return (temp);
                }
                else{
                        print(“Underflow”);
                        throw new UnderflowException;
                }
```

```
public AnyType pop2(){
        if (top2 < n – 1){
                AnyType temp = arr[top2 + 1]
                top2++;
        }
        else{
                print("Underflow");
                throw new UnderflowException;
        }
```

Written 4

a)

1. Move 4 to S1
2. Move 3 to S1
3. Move 1 to Back
4. Move 8 to S2
5. Move 2 to Back
6. Move 3 from S1 to Back
7. Move 4 from S1 to Back
8. Move 7 to S2
9. Move 6 to S2
10. Move 9 to S1
11. Move 5 to Back
12. Move 6 from S2 to Back
13. Move 7 from S2 to Back
14. Move 8 from S2 to Back
15. Move 9 from S1 to Back

b)

A train with train cars: 178965432 cannot be rearranged in increasing order using 3 holding stations. Since cars can only be stacked on cars of greater value to guarantee this rearrangement, once the 5 car is reached, it is guaranteed that a lower number car will be stacked by a higher number car, making the rearrangement impossible. In the case described, 2 goes to S1. 3 cannot stack on 2 so it will be placed in S2. 4 can neither stack on 3 nor 2, so 4 goes to S3. However, once 5 is reached, it has nowhere to go and must hide a lower number.