Lydia Jiang
lyj2001

Homework 2

#1

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class HW2 {

        public static void printLots(List<Integer> L, List<Integer> P) {
                List<Integer> result = new ArrayList<Integer> ();

                Iterator<Integer> iterL = L.iterator();
                Iterator<Integer> iterP = P.iterator();

                int posL = 0;


                while (iterL!=null && iterP!= null && iterP.hasNext()) {

                        int index = iterP.next();
                        System.out.println("Index: " + index);

                        //position L is not in list P
                        while (index > posL) {
                                posL++;
                                iterL.next();
                        }
                        if (index == posL)
                        {
                                result.add(iterL.next());
                                posL++;
                        }

                }

                for (Object o: result) {
                        System.out.println(o);
                }

        }
```

```
	public static void main (String[] args) {

		List<Integer> L = new ArrayList<Integer>();
		L.add(0);
		L.add(1);
		L.add(2);
		L.add(3);
		L.add(4);
		L.add(5);
		for (Object o: L) {
			System.out.println(o);
		}
		List<Integer> P = new ArrayList<Integer>();
		P.add(0);
		P.add(2);
		P.add(5);
		printLots(L,P);

	}

}
```

#2: 3.4

```
List result;
ListIterator iter1 = L1.iterator();
ListIterator iter2 = L2.iterator();
ListIterator resultPos = result.iterator();


while (iter1 != null && iter2 !== null)
{
	if( iter1 → element < iter2 → element )
		iter1 = iter1 → next;
	else if (iter1 → element > iter2 → element )
		iter2 = iter2 → next;
	else
	{
		insert(iter1 → element, result, resultPos);
		iter1 = iter1 → next;
		iter2 = iter2 → next;
		resultPos = resultPos → next;
	}
}
```

return result;

3. 24

```
class twoStacks {

        int[] arr;
        int size;                               //size of array
        int top1;
        int  top2;

        public twoStacks (int n)                //constructor

                size = n
                arr = new int[n];
                top1 = -1;
                top2 =  size;


        public void push1(int x)                //adds x to stack1
                if (top1 < top2 -1)             //at least one empty space to add x
                        top1++;                 //increment counter
                        arr[top1] = x           //add x to array
                else
                        Stack Overflow          //Stack Overflow when there are no empty spaces
                        Exit

        public void push2 (int x)               //adds x to stack2
                if (top1< top2-1)               //if there is space
                        top2--;
                        arr[top2] = x;          //add x to array
                else
                        StackOverflow           //Stack Overflow when there are no empty spaces
                        exit

        public int pop1()                       //remove first element of first stack
                if (top1>=0)                    //if not empty
                        int x  = arr[top1];
                        top1--;
                        return x;
                else
                        StackOverflow
                        exit

        public int pop2()                       //remove first element of second stack
                if (top2 < size)                //if not empty
                        int  x =arr[top2];
                        top2++;
                        return x;
                else
                        StackOverflow
                        exit


}
```

4a)

Input	Holdings			Output
5 9 6 7 2 8 1 3 4				
5 9 6 7 2 8 1 3	4			
5 9 6 7 2 8 1	3 4			
5 9 6 7 2 8	1 3 4			
5 9 6 7 2	1 3 4	8		
5 9 6 7	1 3 4	 2 8		
5 9 6	1 3 4	 2 8	 7	
5 9	1 3 4	 2 8	 6 7	
5 9	3 4	2 8	6 7	1
5 9	3 4	 8	6 7	2 1
5 9	 4	 8	6 7	3 2 1
5 9		 8	6 7	4 3 2 1
5	 9	 8	6 7	4 3 2 1
	5 9	 8	6 7	4 3 2 1
			6	5 4 3 2 1

	9	8	7	
	9	8	7	6 5 4 3 2 1
	9	8		7 6 5 4 3 2 1
	9			8 7 6 5 4 3 2 1
				9 8 7 6 5 4 3 2 1

b) 1 2 3 9 8 7 6 5 4

Input	Holdings			Output
1 2 3 9 8 7 6 5 4				
1 2 3 9 8 7 6 5	4			
1 2 3 9 8 7 6	4	5		
1 2 3 9 8 7	4	5	6	

No more moves ☹