Name: Yu Zheng
UNI: yz2978

1.

```
public static <AnyType> void printLots(List<AnyType> L,List<Integer> P){
        int indexl=0;
        Iterator<Integer> IteratorP = P.iterator();
        Iterator<AnyType> IteratorL = L.iterator();
        int a=IteratorP.next();
        AnyType b=IteratorL.next();
        while (IteratorL.hasNext()){
                if (IteratorP.hasNext()){
                        if (a==indexl){
                                System.out.println(b);
                                a=IteratorP.next();
                        }
                }
                else{
                        break;
                }
                b=IteratorL.next();
                indexl++;
        }
        if(a>indexl){
                System.out.println("the rest values in P exceed the limit of L");
        }

}
```

2.

```
//assume the two lists are ascending sorted
L3=new List
ite1=L1.iterator
ite2=L2.iterator
temp1=ite1.next
temp2=ite2.next
while (1){
        comp= temp1.compareTo(temp2)
        if (comp==0){
                L3.add(temp1)
                if (!ite1.hasNext | !ite2.hasNext ) break
                else{
                        temp1=ite1.next
                        temp2=ite2.next
                }
        }
        else if(comp>0){
                if(!ite2.hasNext) break
                else temp2=ite2.next
        }
        else{
                if((!ite1.hasNext) break
                else temp1=ite1.next
        }

}
```

3.(using java code for this question)

```java
import java.util.EmptyStackException;

public class TwoStacks<AnyType> {
	private AnyType[] data;
	private int length;
	private int leftindex;
	private int rightindex;
	private static final int length0=40;
	@SuppressWarnings("unchecked")
	public TwoStacks(int length){
		data = (AnyType[]) new Object[length];
		this.length=length;
		rightindex= length-1;
	}
	public TwoStacks(){
		this(length0);
	}
	public void pushleft(AnyType item){
		if (isFull()){
			throw new StackOverflowError();
		}
		data[leftindex]=item;
		leftindex++;
	}
	public void pushright(AnyType item){
		if (isFull()){
			throw new StackOverflowError();
		}
		data[rightindex]=item;
		rightindex--;
	}
	public AnyType popleft(){
		if(leftindex<=0){
			throw new EmptyStackException();
		}
		AnyType temp=data[leftindex-1];
		data[leftindex]=null;
		return temp;
	}
	public AnyType popright(){
		if(rightindex>=length-1){
			throw new EmptyStackException();
		}
		AnyType temp=data[rightindex+1];
		data[rightindex]=null;
		return temp;
	}
	public int size(){
		return length+leftindex-rightindex-1;
	}
	public boolean isFull(){
		return size()==length;
	}

}
```

4. (a)
move 4 to S_1; (S_1:4;S_2:empty;S_3:empty;output:empty)
move 3 to S_1; (S_1:3,4;S_2:empty;S_3:empty;output:empty)
move 1 to output; (S_1:3,4;S_2:empty;S_3:empty;output:1)
move 8 to S_2; (S_1:3,4;S_2:8;S_3:empty;output:1)
move 2 to output; (S_1:3,4;S_2:8;S_3:empty;output:1,2)
move 7 to S_2; (S_1:3,4;S_2:7,8;S_3:empty;output:1,2)
move 6 to S_2; (S_1:3,4;S_2:6,7,8;S_3:empty;output:1,2)
move 9 to S_3; (S_1:3,4;S_2:6,7,8;S_3:9;output:1,2)
move 3 then 4 to output; (S_1:empty;S_2:6,7,8;S_3:9;output:1,2,3,4)
move 5 to output; (S_1:empty;S_2:6,7,8;S_3:9;output:1,2,3,4,5)
move 6 then 7 then 8 to output; (S_1:empty;S_2:empty;S_3:9;output:1,2,3,4,5,6,7,8)
move 9 to output; (S_1:empty;S_2:empty;S_3:empty;output:1,2,3,4,5,6,7,8,9)

(4)assume that we have a train with the order of [1,9,8,7,6,5,4,3,2]
we have to first move 2 to a stack, then 3、4 to the other two(they cannot share a stack because the value is increasing). Now 5 is not able to go to any of the stacks because its value is larger, and we can not output 2 or 3 or 4 because 1 has not shown up.