Maryam Aly
UNI: mya2114
Data Structures Hw2
Professor Blaer
Date: Oct. 3rd 2016

Data Structures Hw2

1)

```
public void printLots(Collection<Integer> L, Collection<Integer> P){

                    //iterating through list P
                     for(int e : P){
                          //creating iterator for list L
                           if(e<L.size && e>0){

                           Iterator<Integer> itrerator = L.Iterator();

                           //iterating through the list l
                         for(int i =0 ; i<e; i++){
                               itrerator.next();

                         }
                         //printing the results
                         System.out.println(itrerator.next());
                          }
                     }
```

2)

create a method called listIntersection and it takes in List1, List2

create a resultList which will store the intersection between L1 and L2

create two variables that keep track of the position and initialize them 0

Ex:

```
int posList1 = 0;
int posList2 = 0 ;
while( posList1< L1.size() & posList2 < L2.size())
        if (L1.get(posList1)).compareTo(L2.get(posList2)) < 0
        print posList1 of L1 into the resultList
        then increment posList1 by 1

        else if (L1.get(posList1)).compareTo(L2.get(posList2)) > 0
        then print posList2 of L2 into the resultList
        then increment posList2 by 1
        else
        (L1.get(posList1)).compareTo(L2.get(posList2)) = 0
        then print either the value of (posList2 of L2) or the value of (posList1 of L1) to the resultList
        increment both posList1 and posList2 by 1
return resultList
```

3)

```
n=sum of stack1 size and stack2 size ;
array[n]
stack1 top = -1
stack2 top = n

stack1 push(x){
        check if (stack1top< stack2top-1)
        stack1top ++
        array[stack1top] = x

        else print ("array is overflowed")
}

stack2 push(x){
        check if (stack1top< stack2top-1)
        stack2top --
        array[stack2top] = x

        else print ("array is overflowed")
}



stack1pop() {
            if (stack1top >= 0)
                x = array[stack1top]
                stack1 top --
```

```
                else print ("stack1 is empty")
}

stack2pop() {
        if (stack2top < n)
            x = array[stack2top]
            stack2top ++

        else print ("stack2 is empty")
}
```

4)

a)

1) Put 4 in S2

2) Put 3 in S2

3) Put car#1 in Output

4) Put car#8 in S1

5) Put car#2 in Output

6) Take car#3 from S2 and put it in Output

7) Take car#4 from S2 and put it in Output

8) Put car#7 in S2

9) Put car#6 in S2

10) Put car#9 to S3

11) Put car#5 to Output

12) Put car#6 from S2 to Output

13) Put car#7 from S2 to Output

14) Put car#8 from S1 to Output

15) Put car#9 from S3 to Output

b)

(4, 5, 1, 8, 2, 7, 9, 3)

That is:

- car#4 being the front car

and

- car#3 being the back car