Name: Md Habibur Rony
Student ID: 984582
Weekday: Week 2- Day 10

Answer to the Q. No. R-3.19:

```
Algorithm removeElement(x)
    Input: key x for remove item
    Output: item deletion with the key
    p<-first position top list
    p<-find(x,p)
    if p != null
        while p != NO_SUCH_KEY do
            if p.before = MINUS_INF ^ p.after = PLUS_INF
                p.before.below.above <- null
                p.after.below.above <- null
            a<-p.below
            tmp <- p.before
            tmp.after = p.after
            p.after.before = tmp
            p<-a

Algorithm find(x,p)
    Input:key x,postion p
    Output: postion of the key x
    y<-key(p.after)
    if x = y
        return p.after
    else if x > y then
        return find(x,p.after)
    else
        return find(x,p.below)
    return null
```

Answer to the Q. No. C-4.16:

Algorithm isExistTwoEqualElement(S) Input: Sequence S with n elements Output: true if exist otherwie false	
Dic<-new Dictionary(HashTable)	O(1)
for i<--0 to S do D.insertItem(i, S[i])	O(n) O(n)

for Each item in S do
if Dic.findElement(item.value)!= null then
return true
return false
	O(n)
	O(n)
	O(1)
	O(1)

	T(n) = O(n)

Answer to the Q. No. C-4.18:

```
void inPlacePartition (int s, int nLo, int nHi) {
   if (nHi <= nLo) return ;
   int nLt = nLo, nGt = nHi;
   int nPivot = s atRank(nLo);
   int i = nLo;
   while (i <= nGt) {
      if (s.atRank(i) == nPivot)
         ++i;
      else if (s.atRank(i) > nPivot)
         swap(s.atRank(i), s.atRank(nGt--));
      else {
         swap(s.atRank(i++), s. atRank(nLt++));
      }
   inPlacePartition (s, nLo, nLt - 1);
   inPlacePartition (s, nGt + 1, nHi);
```

Answer to the Q. No. C-4.19:

```
Algorithm merge(A, B, C)
   Input: sequences A and B with n/2 elements each, comparator C
   Output: count of number of inversion
   count<-0
   S <- empty sequence
   while !A.isEmpty() ^ !B.isEmpty() do
```

```
        if C.isLessThan( B.first().element(), A.first().element() ) then
            S.insertLast(B.remove(B.first()))
            count <- count + 1
        else
            S.insertLast(A.remove(A.first()))
        while !A.isEmpty() do
            S.insertLast(A.remove(A.first()))
        while !B.isEmpty() do
            S.insertLast(B.remove(B.first()))
    return count,S

Algorithm countInversion(S, C)
    Input : sequence S with total order n elements, comparator C
    Output: number of Inversion

    if S.size() > 1 then
        (S1, S2)<-partition(S, n/2)
        countInversion(S1, C)
        countInversion(S2, C)
        (S,cnt) <-merge(S1, S2, C)
        count <- count + cnt
    return count
```

Answer to the Q. No. C-4.19:

```
Algorithm matchNutsBolts(A,B)
        Input: Sequence A of nuts, sequence B of bolts
        Output: Matched set of nuts and bolts
        D<-new Dictionary(HashTable)
        For all bots in B do
             D.insertItem(bolts,0)
        PQ<-new PriorityQueue(Array)
        For all nuts in A do
            PQ.insert(nuts,D.findElement(nuts))
        return PQ
```