Set D2 - Part 1

allot_seat(A, n, t)

```
//find position of last element in the array A
    1. max = -1
    2. last = -1
    3. for i ← 0 to n-1 do
            if A[i] > max
            then last ← i
                 max = A[i]
//calculate next position to insert the new element
    4. next ← (last + 1) % n
       //insert t in next position
    5. A[next] ← t
    6. print next in new line
```

Evaluation criteria : **[4 marks]**
Division: Finding the next position to insert - 2 marks
Calculating next position in circular manner - 2 Marks

process_transaction(A, n)

```
//find position of smallest element in the array A
    1. first ← 0
    2. for i ← 1 to n-1 do
            if A[i] < A[first]
            then first ← i
//delete the element at position first
    3. A[first] ← -1
    4. print first in new line
```

Evaluation criteria : **[3 marks]**
Division: Finding the position to delete - 2 marks
Deletion - 1 Mark

Set D2 - Part 2

```
allot_seat(A, B, n, t, p)
//find the first empty position
    1. for i ← 0 to n-1 do
           if A[i]  = -1
               then break
//insert t and p at position i
    2. A[i] ← t
    3. B[i] ← p
    4. print i in new line

process_transaction(A, B, n)
//find position of highest preference in the array B
    1. high ← 0
    2. for i ← 1 to n-1 do
               if B[i] > B[high]
                  then high ← i
               //if preferences are same, select the position of smaller value in A
               else if B[i] = B[high]
                       then if A[i] < A[high]
                               then high ← i
//delete the element at position i
    3. A[i] ← -1
    4. B[i] ← -1
    5. print newline; print i;

update_preference(A, B, t, p)
//find the position of t in A
    1. for i ← 0 to n-1 do
           if A[i] = t
               then break
//update the preference in B at position i
    2. B[i] ← p
```

Array_Empty(A)

```
1. for i ← 0 to n-1 do
     if A[i] ≠ -1
          then return 0
2. return 1
```

sort_customers()

//repeatedly process_transaction(A, B, n) until array is empty

```
1. while Array_Empty(A) ≠ 1
      do process_transaction(A, B, n)
```

print_customers(A, B, n)

```
1. for i ← 0 to n-1 do
       if A[i] = -1
          then print newline; print -1;
       else print newline; print A[i]; print ' ' ; print B[i];
```

Evaluation criteria : **[3 marks]**

Division: ***sort_customers()*** function using ***process_transaction()*** - 1 mark

Other four functions - 2 Marks (0.5 each)