CSE 220: Data Structures

Quiz 1

Question 01[CO1]

Given an array containing integer elements, find the minimum value in the array and shift right in such a way that the minimum element is in the last valid position.
[Note: You are NOT allowed to use any additional array or built-in function]

Sample Input	Sample Output
[5,6,3,5,1,2]	[0,5,6,3,5,1]
[20,10,-2,5,3,-1]	[0,0,0,20,10,-2]

```
def shiftRightMinLast(source):
      #TO DO
```

OR

```
public static int[]shiftRightMinLast(int [] source) {
      //TO DO

}
```

```
def shiftRightMinLast(source):
      #finding minimum value
      min = source[0]
      minInd = 0
      for i in range(1,len(source)):
            if(source[i]<min):
                  min = source[i]
                  minInd = i

      #shifting times
      k = len(source) - minInd - 1

      #right shifting k times
      for i in range(k):
            for j in range(len(source)-1,0,-1):
                  source[j]=source[j-1]
            source[0]=0

      return source
```

Question 02[CO1]

Given circular array:

Index	0	1	2	3	4	5	6	7	8
Value	NULL	NULL	54	45	10	14	75	45	1

Start = 2; size = 7

Draw the resultant circular array after each of the below steps.
Step1: Insert 78 at the position 4 with the help of right-shifting.
Step2: Insert 15 at the position 6 with the help of left-shifting.
Step3: Insert -25 at the position 3 with the help of right-shifting.

N.B.: Position is the relative distance from start; while position of start is ZERO. If the array is full, you MUST RESIZE BY LINEARINZING and the new capacity would be (previous capacity +2).

Step 1: Start=2

Index	0	1	2	3	4	5	6	7	8
Value	1	NULL	54	45	10	14	78	75	45

Step 2: Start=1

Index	0	1	2	3	4	5	6	7	8
Value	1	54	45	10	14	78	75	45	15

Step 2: Start=0

Index	0	1	2	3	4	5	6	7	8	9	10
Value	54	45	10	14	78	75	45	15	1	NULL	NULL

Index	0	1	2	3	4	5	6	7	8	9	10
Value	54	45	10	-25	14	78	75	45	15	1	NULL

CSE 220: Data Structures

Quiz 2

Question 01[CO1]

You are given the following singly-linked Node class:

```
public class Node {
      Object elem;
      Node next;
      public Node(Object v, Node n)
      {elem = v; next = n;}
}
```

```
class Node:
      def __init__(v,n):
            self.elem = v
            self.next = n
```

Given a singly linked list of integers, rearrange the elements of the list in such a way that the odd numbers will be placed at the end of the list in reverse order.

Sample Input	Sample Output
1->2->3->4->5->6->7->8	2->4->6->8->7->5->3->1
1->2->3->4->5	2->4->5->3->1

```
def oddReverse(head):

      #TO DO
OR
public static Node oddReverse (Node head){

      //TO DO
}
```

```
def oddReverse(head):
  #creating an odd numbers linked list in reverse order
  oddListHead = None
  evenListHead = None #creating an even numbers linked list
  evenListTail = None
  n = head
  while (n!= None):
    newNode = Node(n.elem,None)

    if(n.elem%2==0):  #even
      if(evenListHead==None):
        evenListHead = newNode
        evenListTail = newNode
      else:
        evenListTail.next = newNode
        evenListTail = newNode
    else:
      newNode.next = oddListHead
      oddListHead = newNode
    n = n.next

  evenListTail.next = oddListHead
  return evenListHead
```