Q1

```
public class Bsort {

   void printArray(int arr[]){
    int n = arr.length;
   for(int i=0;i<n;i++)
   {
    System.out.print(arr[i]+" ");
    }System.out.println();
   }
   void bsort(int arr[]) {
    for(int i =0;i<arr.length-1;i++){
     for (int j =0;j<arr.length-i-1;j++){
      if(arr[j]>arr[j+1]){
       int temp =arr[j];
       arr[j]=arr[j+1];
       arr[j+1]=temp;
        }
       }
      }
   }
  public static void main(String[] args){
   Bsort s1 =new Bsort();
   int arr[]={7,8,3,1,2};
   s1.printArray(arr);
   s1.bsort(arr);
   System.out.println();
   s1.printArray(arr);

   }
  }
```

Q2.

```
public class Qsort {
  private static void quicksort(int[] arr, int low, int high){
   if(low<high)
   {
    int pivot = partition(arr, low, high); //will return pivot element
    quicksort(arr,low, pivot-1);
    quicksort(arr, pivot+1, high);
   }
     }

  private static int partition(int[] arr, int low, int high)
  {
   int pivot = arr[high]; //you can take either low or high(quick sort algorithm diagram where 13 is pivot)
   int i = (low-1);
   for(int j=low; j<=high-1; j++)
   {
    if(arr[j] < pivot)
    {
     i++;
     swap(arr,i,j);
```

```
    }
   }
   swap(arr,i+1,high);
   return (i+1);  //left and right array separate hoga yeh point pe
  }

 private static void swap(int[] arr, int i , int j)
 {
  int temp = arr[i];
  arr[i] = arr[j];
  arr[j] = temp;
 }

     public static void main(String args[]){
         int arr[] = {23,123,43,12,4,312,22};
         quicksort(arr, 0, arr.length-1);


   for(int i : arr)
            System.out.print(i + " ");
     }
}
```

Q3.

```
public class Ssort {
  void printArray(int arr[]){
   int n = arr.length;
  for(int i=0;i<n;i++)
  {
   System.out.print(arr[i]+" ");
   }System.out.println();
  }
  void ssort(int arr[])
  {
   int n = arr.length;
   for(int i=0;i<n-1;i++)
   {
    int min = i;
    for(int j=i+1;j<n;j++)
     if(arr[j] < arr[min])
      min = j;

    int temp = arr[min];
    arr[min] = arr[i];
    arr[i] = temp;

   }
  }
  public static void main(String[] args){
   Ssort s1 =new Ssort();
   int arr[]={7,8,3,1,2,4,5};
```

```
   s1.printArray(arr);
   s1.ssort(arr);
   System.out.println();
   s1.printArray(arr);

   }
}
```

Q4.

```
public class Isort {
 void isort(int arr[])
 {
  int n = arr.length;
  for(int i = 1;i<n;i++)
  {
   int key = arr[i];
   int j=i-1;

   while(j>=0 && arr[j]>key)
   {
    arr[j+1] = arr[j];
    j=j-1;
   }
   arr[j+1] = key;

  }
 } void printArray(int arr[]){
   int n = arr.length;
  for(int i=0;i<n;i++)
  {
   System.out.print(arr[i]+" ");
   }System.out.println();
  }


  public static void main(String[] args) {
   Isort s1 =new Isort();
   int arr[]={7,8,3,1,2,4};
   s1.printArray(arr);
   s1.isort(arr);
   System.out.println();
   s1.printArray(arr);

  }
}
```

Q5.

```
public class mergeSort {
 static void Msort(int arr[], int l,int r) {
  if(l<r) {
   int mid=(l+(r-l)/2);
   Msort(arr,l,mid);//left part
```

```
  Msort(arr,mid+1, r);//right part
  merge(arr,l,mid,r);
 }
}
 static void merge(int arr[],int l,int mid,int r) {
  {
   int n1 = mid-l+1;
   int n2=r-mid;

   int L[]=new int[n1];
   int R[]=new int[n2];

   for(int i=0;i<n1;i++)
    L[i]=arr[l+i];

   for(int j=0;j<n2;j++)
    R[j]=arr[mid+1+j];

   int i=0,j=0;
   int k=l;
   while(i<n1 && j<n2)
   {
    if(L[i]<=R[j]) {
     arr[k]=L[i];
     i++;
    }else {
     arr[k]=R[j];
     j++;
    }
    k++;
   }
   while(i<n1) {
    arr[k]=L[i];
    i++;
    k++;
   }while(j<n2) {
    arr[k]=R[j];
    j++;
    k++;
   }
  }

 }
 void display(int arr[])
 {
  int n =arr.length;
  for(int i=0;i<n;i++)
  {
   System.out.print(arr[i]+ " ");
  }
 }

public static void main(String[] args) {
mergeSort h1 = new mergeSort();
int arr[]= {99,89,34,11,55,33,88,44,22};
```

```
int n =arr.length;

h1.display(arr);
Msort(arr, 0, n-1);
System.out.println();
h1.display(arr);
 }
}
```

Q6.

```
public class Qsort {
  private static void quicksort(int[] arr, int low, int high){
   if(low<high)
   {
    int pivot = partition(arr, low, high); //will return pivot element
    quicksort(arr,low, pivot-1);
    quicksort(arr, pivot+1, high);
   }
     }

  private static int partition(int[] arr, int low, int high)
  {
   int pivot = arr[high]; //you can take either low or high(quick sort algorithm diagram where 13 is pivot)
   int i = (low-1);
   for(int j=low; j<=high-1; j++)
   {
    if(arr[j] < pivot)
    {
     i++;
     swap(arr,i,j);
    }
   }
   swap(arr,i+1,high);
   return (i+1);  //left and right array separate hoga yeh point pe
  }

 private static void swap(int[] arr, int i , int j)
 {
  int temp = arr[i];
  arr[i] = arr[j];
  arr[j] = temp;
 }

     public static void main(String args[]){
         int arr[] = {23,123,43,12,4,312,22};
         quicksort(arr, 0, arr.length-1);


   for(int i : arr)
            System.out.print(i + " ");
     }
}
```

Q7.

```
public class sString {
 void bsort(String[] arr) {
   for(int i =0;i<arr.length-1;i++){
    for (int j =0;j<arr.length-i-1;j++){
     if(arr[j].compareTo(arr[j+1])>0){
      String temp =arr[j];
      arr[j]=arr[j+1];
      arr[j+1]=temp;
       }
      }
     }
  }
 public static void main(String[] args) {
  sString s1=new sString();
  String [] str= {"banana","apple", "orange", "grape", "kiwi"};

  System.out.println("Original :"+"===>");

  for(String s: str) {
   System.out.print(s+" ");
  }System.out.println();
  s1.bsort(str);

  System.out.println("Sorted"+"====>");
  for (String s:str) {
   System.out.print(s+" ");
  }
 }
}
```

Q9.

```
public class bsortLL {
 Node head;
 class Node{
  int data;
  Node next;

  Node(int d){
   data=d;
   next=null;
  }
 }
 void append(int new_data)
 {
  Node new_node = new Node(new_data);

  if(head == null)
   {
    head = new_node;
    return;
   }
```

```
 new_node.next = null;
 Node last = head;
 while(last.next != null)
 {
  last = last.next;
 }
 last.next = new_node;
 return;
}
void display()
{
Node n = head;
while(n!= null)
{
 System.out.print(n.data+ "---> ");
 n=n.next;
}
}

void bubbleSort() {
      if (head == null || head.next == null)
          return;

      boolean swapped;
      Node ptr1;
      Node lptr = null;

      do {
          swapped = false;
          ptr1 = head;

          while (ptr1.next != lptr) {
              if (ptr1.data > ptr1.next.data) {
                  // Swap the nodes
                  int temp = ptr1.data;
                  ptr1.data = ptr1.next.data;
                  ptr1.next.data = temp;
                  swapped = true;
              }
              ptr1 = ptr1.next;
          }
          lptr = ptr1;
      } while (swapped);
   }
public static void main(String[] args) {
 bsortLL b1= new bsortLL();
 b1.append(5);
 b1.append(3);
 b1.append(8);
 b1.append(1);
 b1.append(2);
 System.out.println("LL Before Sorting :");
 b1.display();
 b1.bubbleSort();
 System.out.println();
```

```
  System.out.println("LL after sorting :");
  b1.display();
 }

}
```

Q10.

```
public class bsortDLL {
 Node head;

 static class Node{
  int data;
  Node next,prev;

  Node(int d)
  {
   data = d;
   next = null;
   prev = null;
  }
 }void append(int new_data)
 {
  Node new_node = new Node(new_data);
  Node last = head;
  new_node.next = null;
  if( head == null)
  {
   new_node.prev=null;
   head = new_node;
   return;
  }
  while(last.next != null)
  {
   last=last.next;
  }
  last.next = new_node;
  new_node.prev = last;
 }public void bubbleSort() {
        if (head == null)
         return;

        boolean swapped;
        Node last = null;
        do {
            swapped = false;
            Node current = head;
            while (current.next != last) {
                if (current.data > current.next.data) {
                    swap(current, current.next);
                    swapped = true;
                }
                current = current.next;
            }
            last = current;
```

```
        } while (swapped);
    }
    private void swap(Node node1, Node node2) {
        int temp = node1.data;
        node1.data = node2.data;
        node2.data = temp;
    }void display(Node n)
 {
  Node p = null;
  while(n != null)
  {
   System.out.print(n.data+"---> ");
   p=n;
   n=n.next;
  }
  System.out.println();
 }
 public static void main(String[] args) {
  bsortDLL b1=new bsortDLL();
  b1.append(5);
  b1.append(3);
  b1.append(8);
  b1.append(1);
  b1.append(2);
  b1.append(6);

  System.out.println("Original List :");
  b1.display(b1.head);

  b1.bubbleSort();
  System.out.println("Sorted List :");
  b1.display(b1.head);
 }
}
```