Q1: SORT ARRAY WHICH CONTAIN ONLY 0 AND 1

ANS:

```
static void sortZeroandOne(int []arr){
    int n=arr.length;
    int zeroes=0;
    for (int i = 0; i <n ; i++) {
        if(arr[i]==0){
            zeroes++;
        }
    }
    for (int i = 0; i <n ; i++) {
        if(i<zeroes){
            arr[i]=0;
        }
        else{
            arr[i]=1;
        }
    }
}
```

Q2: SORT ARRAY IN SUCH WAY THAT EVEN NUMBER COMES FIRST THEN ODD NUMBER

ANS:

```
static void swap(int []arr,int i,int j){
    int temp=arr[i];
    arr[i]=arr[j];
    arr[j]=temp;
}
static  void sortOddEven(int[]arr2 ){
    int n=arr2.length;
    int left=0;
    int right=n-1;
  while (left<right){
```

```
        if(arr2[left]%2==1 && arr2[right]%2==0){
            swap(arr2,left,right);
            left++;
            right--;
        }
        if (arr2[left]%2==0){
            left++;
        }
        if(arr2[right]%2==1){
            right--;
        }
    }
}
```

Q3:SORT ACCORDING TO SQUARE OF ELEMENTS

ANS:

```
static int[] sortSquare(int[]arr3){
 int n=arr3.length;
 int left=0;
 int right=n-1;
 int [] ans=new int[n];
 int k=0;
 while (left<=right){
     if (Math.abs(arr3[left])>Math.abs(arr3[right])){
         ans[k++]=arr3[left]*arr3[left];
         left++;
     }
     else{
         ans[k++]=arr3[right]* arr3[right];
         right--;
     }
 }
  return ans;
}
```

Q4: Given an array arr[] of size n, find the first repeating element. The element should occur more than once and the index of its first occurrence should be the smallest. If no repeating element exists, print -1.

ANS:

```
int m=8;
        int [] myarr={1,2,3,2,3,5,5,7};
        int rep=0;
    for (int i = 0; i <m ; i++) {
        for (int j = i+1; j <m ; j++) {
            if (myarr[i]==myarr[j]){
                rep++;
            }
        }
    }
    System.out.println(rep);
    }
}
```

5. Given an array of positive and negative numbers, arrange them in an alternate fashion such that every positive number is followed by a negative and vice-versa maintaining the order of appearance.
The number of positive and negative numbers need not be equal. Begin with a negative number.
If there are more positive numbers, they appear at the end of the array. If there are more negative numbers, they too appear at the end of the array.

SOL:

```
public static void alternateArrange(int[] arr) {
int[] positive = new int[arr.length];
 int[] negative = new int[arr.length];
int posCount = 0;
int negCount = 0;
for (int num : arr)
 {
if (num < 0)
{
 negative[negCount] = num; negCount++;
 }
 else {
 positive[posCount] = num; posCount++;
}
 }
 int[] result = new int[arr.length];
int i = 0, j = 0, k = 0;
 while (i < posCount && j < negCount) {
 result[k++] = negative[j++]; result[k++] =
positive[i++]; }
         while (i < posCount) { result[k++] =
positive[i++]; }
         while (j < negCount) { result[k++] =
negative[j++]; }
                 System.arraycopy(result, 0, arr, 0,
arr.length); }
```