```
//-->HARSH MISHRA<--


void selectionSort(int A[], int n) {
    for(int i = 0;i < n-1; i++) {
        int iMin = i;
        for(int j = i+1; j < n;j++)
            if(A[j] < A[iMin])  iMin = j;
        swap(A[i],A[iMin]);
    }
}


void bubbleSort(int A[], int n) {
    for(int i = 0; i < n-1; i++) {
        for(int j = 0; j <n-1-i;j++)
            if(A[j] > A[j+1])   swap(A[j],A[j+1]);
    }
}


void insertionSort(int A[], int n) {
    for(int i = 1;i < n;i++) {
        int val = A[i];
        int hole = i;
        while(hole > 0 && A[hole-1] > val) {
            A[hole] = A[hole-1];
            hole = hole-1;
        }
        A[hole] = val;
    }
}

int partition(int A[], int start, int end) {
    int pivot = A[end];
    int partitionIndex = start;
    for(int i = start;i < end; i++) {
        if(A[i] <= pivot) {
            swap(A[i],A[partitionIndex]);
            partitionIndex++;
        }
    }
    swap(A[partitionIndex],A[end]);
    return partitionIndex;
}

```

```

void quickSort(int A[], int start, int end) {
    if(start < end) {
        int partitionIndex = partition(A,start,end);
        quickSort(A,start,partitionIndex-1);
        quickSort(A,partitionIndex+1,end);
    }
}

void merge(int A[], int start, int mid, int end) {
    int p = start, q = mid+1;
    int Arr[end-start+1], k = 0;
    for(int i = start; i <= end;i++) {
        if(p > mid) Arr[k++] = A[q++];
        else if(q > end) Arr[k++] = A[p++];
        else if(A[p] < A[q]) Arr[k++] = A[p++];
        else Arr[k++] = A[q++];
    }
    for(int p = 0;p < k;p++) A[start++] = Arr[p];
}

void mergeSort(int A[], int start, int end) {

    if(start < end) {
        int mid = (start+end)/2;
        mergeSort(A,start,mid);
        mergeSort(A,mid+1,end);
        merge(A,start,mid,end);
    }
}

```