# Sorting

## 1.sort the array of zeros and ones

void swapZeros(int arr[],int n){

    for(int i=n-1;i>=0;i--){

        int j=0;

        bool flag=false;

        while(j!=i){

            if(arr[j]==0 && arr[j+1]!=0){

                swap(arr[j],arr[j+1]);

                flag=true;

            }

            j++;

        }

        if(!flag) break;

    }

}

void print\_array(int arr[],int n){

    for(int i=0;i<n;i++){

        cout<<arr[i]<<" ";

    }

}

## 2.sort array of string in lexicographical order

### Method 1

 void dictionary(char arr[][60],int n){

    bool flag;

    for(int i=0;i<n-1;i++){

        flag=false;

        for(int j=0;j<n-i-1;j++){

            if (strcmp(arr[j],arr[j+1])>0){

                swap(arr[j],arr[j+1]);

                flag=true;

            }

        }

    }

 }

### Method 2 -if no swap function

## 3.write a program to find kth smallest element using quick sort

//k-1 index par joh element hoga in the sorted array

// first sort then find O(nlogn)

// of TC O(n)

int partition(int arr[],int low,int high){

int pivot=arr[high];

    int i=(low-1);

    for(int j=low;j<=high-1;j++){

        if(arr[j]<pivot){

            i++;

            swap(arr[i],arr[j]);

        }

    }

    swap(arr[i+1],arr[high]);

    return (i+1);

}

int quickSort(int arr[],int low,int high ,int key){

    int pi=partition(arr,low ,high);

    if(pi==(key-1)) return arr[pi];

    if(pi<(key-1)) quickSort(arr,pi+1,high,key);

    else quickSort(arr,low,pi-1,key);

}

## 4.Sort array zeros, ones, twos

void swap\_0\_1\_2(int arr[],int N){

    int ptr=0,head=0,tail=N-1;

    while(ptr!=tail){

        if(arr[ptr]==2){

            //insertion ele at last using insertion method

            int temp=arr[ptr];

            int temp\_ptr=ptr;

            while(temp\_ptr!=N-1){

                arr[temp\_ptr]=arr[temp\_ptr+1];

                temp\_ptr++;

            }

            arr[N-1]=temp;

            tail--;

        }

        if(arr[ptr]==0){

            if(head==ptr) ptr++;

            else{

                int temp=arr[ptr];

                int temp\_ptr=ptr;

                while(temp\_ptr!=head){

                    arr[temp\_ptr]=arr[temp\_ptr-1];

                    temp\_ptr--;

                }

                arr[head]=temp;

            }

            head++;

        }

        if(arr[ptr]==1) ptr++;

    }

}

## 5.Sort the array of positive and negative elements

void positive\_negitive(int arr[],int N){

    int ptr=0,n\_inx=0;

    while(ptr!=N){

        if(arr[ptr]<0){

            swap(arr[ptr],arr[n\_inx]);

            n\_inx++;

        }

        ptr++;

    }

}

//by maintaining relavitve order

void segregate\_alt(int arr[],int N){

    int ptr=0,n\_inx=0;

    while(ptr!=N){

        if (arr[ptr]<0){

            int temp\_ptr=ptr;

            int temp=arr[ptr];

            while(temp\_ptr!=n\_inx){

                arr[temp\_ptr]=arr[temp\_ptr-1];

                temp\_ptr--;

            }

            arr[n\_inx]=temp;

            n\_inx++;

        }

        ptr++;

    }

}