***Aim***:

To implement various Disk Scheduling algorithms like FIFO, SCAN, SSTF,CSCAN

***Algorithm***:

1.**FIFO**:

1.Add seek times from the (disk arm position-arr[i]) FIFO Sequence

2.Print Seek time

2.***SSTF***:

1.For SSTF create a duplicate array and copy contents of original array

2.add another element to dup which is the diskpos ptr

3.Sort the new array

4.Add up seek times+=diskpos-arr[i]

5.Print the seek time

3.***SCAN***:

1. For SCAN create a duplicate array and copy contents of original array

2. add another element to dup which is the diskpos ptr

3.Sort the new array along with lb,ub,diskptr.

4. iterate from diskpos to left and add up seek times+=(disk arm position-arr[i] till lb

5.jump to arr[ub-1] that is the max req seq and iterate from max to disk pos and add seek time

6.Print the Seek time

4.***CSCAN***:

1. For SCAN create a duplicate array and copy contents of original array

2. add another element to dup which is the diskpos ptr

3.Sort the new array along with lb,ub,diskptr.

4. iterate from diskpos to left and add up seek times+=(disk arm position-arr[i] till min seq and we do not index lb or ub in this case.

5.jump to arr[ub-1] that is the max req seq and iterate from max to disk pos and add seek time

6.Print the Seek time

***Code***:

```
#include <stdio.h>
#include <stdlib.h>
void swap(int *a1,int*a2){
        int tmp=*a1;
        *a2=*a1;
        *a2=tmp;

}
int seqlen;
int FIFO(int arr[seqlen],int diskarmpos){
        int seek_time=0;
        for(int i=0;i<seqlen;i++){
                seek_time+=abs(diskarmpos-arr[i]);
                diskarmpos=arr[i];
        }
        return seek_time;

}
void sort(int *arr,int seqlen)
{
        int tmp;
        for(int i=0;i<seqlen;i++){
                for(int j=0;j<seqlen;j++){
                        if(arr[i]<arr[j]){
                                tmp=arr[i];
                                arr[i]=arr[j];
                                arr[j]=tmp;
                        }
                }
        }
```

```
}
int CSCAN(int arr[seqlen],int lb,int ub,int diskarmpos){
int dup[seqlen+3],i,seek_time=0;
        for( i=0;i<seqlen;i++){
                dup[i]=arr[i];
        }
        int diskidx;
        dup[i]=diskarmpos;
        dup[i+1]=lb;
        dup[i+2]=ub;
        sort(dup,seqlen+3);
        for(int j=0;j<seqlen+1;j++){
                if(dup[j]==diskarmpos){
                        diskidx=j;
                        break;
                }
        }
        int cpdi=diskidx;
        diskidx--;


        for(int i=diskidx;i>=0;i--){
                seek_time+=abs(diskarmpos-dup[i]);
                diskarmpos=dup[i];
        }
        seek_time+=abs(diskarmpos-dup[seqlen+2]);
        diskarmpos=dup[seqlen+2];
        for(int j=seqlen+1;j>cpdi;j--){

                seek_time+=abs(diskarmpos-dup[j]);
                diskarmpos=dup[j];
```

```
	}


	return seek_time;


}
int SCAN(int arr[seqlen],int lb,int ub,int diskarmpos ){
int dup[seqlen+3],i,seek_time=0;
	for( i=0;i<seqlen;i++){
		dup[i]=arr[i];
	}
	int diskidx;
	dup[i]=diskarmpos;
	dup[i+1]=lb;
	dup[i+2]=ub;
	sort(dup,seqlen+3);
	for(int j=0;j<seqlen+1;j++){
		if(dup[j]==diskarmpos){
			diskidx=j;
			break;
		}
	}
	int cpdi=diskidx;
	diskidx--;



	for(int i=diskidx;i>=0;i--){
		seek_time+=abs(diskarmpos-dup[i]);
		diskarmpos=dup[i];
	}
	for(int j=cpdi+1;j<seqlen+2;j++){
		seek_time+=abs(diskarmpos-dup[j]);
```

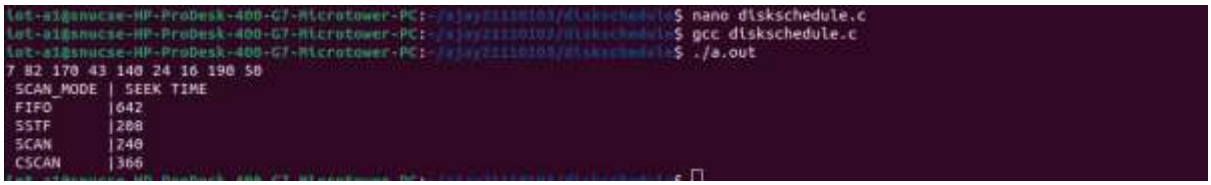
```
                diskarmpos=dup[j];


        }
        return seek_time;
}
int SSTF(int arr[seqlen],int diskarmpos){
        int dup[seqlen+1],i,seek_time=0;
        for( i=0;i<seqlen;i++){
                dup[i]=arr[i];
        }
        int diskidx;
        dup[i]=diskarmpos;
        sort(dup,seqlen+1);
        for(int j=0;j<seqlen+1;j++){
                if(dup[j]==diskarmpos){
                        diskidx=j;
                        break;
                }
        }
        int cpdi=diskidx;
        diskidx--;


        for(int i=diskidx;i>=0;i--){
                seek_time+=abs(diskarmpos-dup[i]);
                diskarmpos=dup[i];
        }
        for(int j=cpdi+1;j<seqlen+1;j++){
                seek_time+=abs(diskarmpos-dup[j]);
                diskarmpos=dup[j];
        }
```

```
        return seek_time;
}
int main(){
int diskarmpos,tmp;
scanf("%d",&seqlen);
int arr[seqlen];
for(int i=0;i<seqlen;i++){
        scanf("%d",&tmp);
        arr[i]=tmp;
}
scanf("%d",&diskarmpos);
printf(" SCAN_MODE | SEEK TIME\n" );
printf(" FIFO      |%d\n",FIFO(arr,diskarmpos));
printf(" SSTF      |%d\n",SSTF(arr,diskarmpos));
printf(" SCAN      |%d\n",SCAN(arr,0,199,diskarmpos));
printf(" CSCAN     |%d\n",CSCAN(arr,0,199,diskarmpos));
}
```

***Output***:




***Result***:

Thus Out of many disk scheduling algorithms 4 of them were implemented and their Seek times were studied