EXP 1

```
#! /bin/bash
op=1
while [ "$op" -lt 6 ]
do
        echo enter the option
        echo "1 for create"
        echo "2 for add"
        echo "3 for display"
        echo "4 for search"
        echo "5 for delete"
        echo "6 for modify"
        echo "7 for exit"
        echo "enter u r choice"
        read op
        word="$op"

        case "$word" in
                1)
                if [ "$op" == "1" ]
                then
                echo "Enter the name for the database"
                read db touch "$db"
                fi
                ;;
                2)
                if [ "$op" == "2" ]
                then
                echo "in which database u want to add records" read db
                echo "enter the no. of records"
                read n
                while [ $it -lt $n ]
                do
                        echo "enter id:" read id1
                        echo "enter name:"
                        read nm pa1="^[A-Za-z]"
                while [[ ! $add =~ $pa ]]
                do
                        echo "enter valid address:"
                        read add
                        done
                        echo "enter address:"
                        read add
                pa="^[A-Za-z0-9]"
                while [[ ! $add =~ $pa ]]
                do
                        echo "enter valid address:"
                        read add
                done
```

```
#echo $add
echo "enter phone no.:"
read ph
pat="^[0-9]{10}$"
while [[ ! $ph =~ $pat ]]
do
        echo "please enter phone number as XXXXXXXXXX:"
        read ph
done
#echo $ph
echo "eter email:"
read em
patem="^[a-z0-9._%-+]+@[a-z]+\.[a-z]{2,4}$"
while [[ ! $em =~ $patem ]]
do
        echo "please enter valid email address"
        read em
done
#echo $em
echo "$id1,$nm,$add,$ph,$em" >> "$db" it=`expr $it + 1`
echo "$it record entered"
done
fi
;;
3)
if [ "$op" == "3" ]
then
echo "enter name of database from where data to be display:"
read db
cat $db
fi
;;
4)
if [ "$op" == "4" ]
then
echo "enter name of database from where to search:"
read db
echo "enter email to be search:"
read em1
grep $em1 $db
echo "record found" else
echo "not found"
fi
;;
5)
if [ "$op" == "5" ]
then
echo "enter name of database:"
read db
echo "enter id:"
```

```
read id1
echo "enter line no. u want to delete:"
read linenumber
for line in `grep -n "$id1" $db`
do
        number=`echo "$line" | cut -c1`
        echo $number
        if [ $number == $linenumber ]
        then
                lineRemove="${linenumber}d"
                sed -i -e "$lineRemove"
                $db echo "record removed"
        fi
        echo cat $db
done
fi
;;
6)
if [ "$op" == "6" ]
then
echo "enter name of database:"
read db

echo "enter id:"
read id1
echo "enter line u want to modify:"
read linenumber
for line in `grep -n "$id1" "$db"`
do
number=`echo "$line" | cut -c1`
if [ "$number" == "$linenumber" ]
then
echo "what would u like to change"
echo "\"id,name,address,mobile,email\""
read edit
linechange="${linenumber}s"
sed -i -e "$linechange/.*/$edit/" $db
echo record edited
fi
done
fi
;;
7)
echo "bye"
;;
*)
echo invalid input
esac
done
```

EXP 2 FORK WAIT

```
# include<stdio.h>
# include <stdlib.h>
# include<sys/types.h>
# include<unistd.h>
# include<sys/wait.h>
int split ( int[], int , int );
void quickSort(int* ,int, int);
void mergeSort(int arr[],int low,int mid,int high)
{
int i,j,k,l,b[20];
l=low;
i=low;
j=mid+1;
while((l<=mid)&&(j<=high))
{
if(arr[l]<=arr[j])
{
b[i]=arr[l];
l++;
}
else {
b[i]=arr[j];
j++;
}
i++;
}
if(l>mid)
{
for(k=j;k<=high;k++)
{ b[i]=arr[k];
i++;
}
}
else {
for(k=l;k<=mid;k++)
{
b[i]=arr[k];
i++;
}
}
for(k=low;k<=high;k++)
{
arr[k]=b[k];
}
}
void partition(int arr[],int low,int high)
{
int mid;
```

```
if(low<high)
{
double temp;
mid=(low+high)/2;
partition(arr,low,mid);
partition(arr,mid+1,high);
mergeSort(arr,low,mid,high);
}
}
void display(int a[],int size)
{
int i;
for(i=0;i<size;i++)
{
printf("%d\t\t",a[i]);
}
printf("\n");
}
int main()
{
int pid, child_pid;
int size,i,status;
printf("Enter the number of Integers to Sort::::\t");
scanf("%d",&size);
int a[size];
int pArr[size];
int cArr[size];
for(i=0;i<size;i++){
printf("Enter number %d:",(i+1));
scanf("%d",&a[i]);
pArr[i]=a[i];
cArr[i]=a[i];
}
printf("Your Entered Integers for Sorting\n"); display(a,size);
pid=getpid();
printf("Current Process ID is : %d\n",pid);
printf("[ Forking Child Process ... ] \n");
child_pid=fork();
if( child_pid < 0){
printf("\nChild Process Creation Failed!!!!\n");
exit(-1);
}
else if( child_pid==0) {
printf("\nThe Child Process\n");
printf("\nchild process is %d",getpid());
printf("\nparent of child process is %d",getppid());
printf("Child is sorting the list of Integers by QUICK SORT::\n");
quickSort(cArr,0,size-1);
printf("The sorted List by Child::\n");
display(cArr,size);
```

```
printf("Child Process Completed ...\n");
sleep(10);
printf("\nparent of child process is %d",getppid());
}
else {
printf("parent process %d started\n",getpid());
printf("Parent of parent is %d\n",getppid());
sleep(30);
printf("The Parent Process\n");
printf("Parent %d is sorting the list of Integers by MERGE SORT\n",pid);
partition(pArr,0,size-1);
printf("The sorted List by Parent::\n");
display(pArr,size);
wait(&status);
printf("Parent Process Completed ...\n");
}
return 0;
}
int split ( int a[ ], int lower, int upper )
{
int i, p, q, t ;
p = lower + 1 ;
q = upper ;
i = a[lower] ;
while ( q >= p )
{
while ( a[p] < i ) p++ ;
while ( a[q] > i )
q-- ;
if ( q > p )
{
t = a[p] ;
a[p] = a[q] ;
a[q] = t ;
}
}
t = a[lower] ;
a[lower] = a[q] ;
a[q] = t ;
return q ;
}
void quickSort(int a[],int lower, int upper)
{ int i ;
if ( upper > lower )
{
i = split ( a, lower, upper ) ;
quickSort ( a, lower, i - 1 ) ;
quickSort ( a, i + 1, upper ) ;
}
}
```

EXP 2 FORK EXECVE

```
#include <stdio.h> //PARENT
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
int main(int argc, char *argv[])
{
	int val[10],ele;
	pid_t pid;
	char* cval[10];
	char *newenviron[] = {NULL };
	int i,j,n,temp;
	printf("\nEnter the size for an array: ");
	scanf("%d",&n);
	printf("\nEnter %d elements: ", n);
	for(i=0;i<n;i++)
		scanf("%d",&val[i]);
	printf("\nEntered elements are: ");
	for(i=0;i<n;i++)
		printf("\t%d",val[i]);
	for(i=1;i<n;i++) {
		for(j=0;j<n-1;j++) {
			if(val[j]>val[j+1])
			{
				temp=val[j];
				val[j]=val[j+1];
				val[j+1]=temp;
			}
		}
	}
	printf("\nSorted elements are: ");
	for(i=0;i<n;i++)
	printf("\t%d",val[i]);
	printf("\nEnter element to search: ");
	scanf("%d",&ele);
	val[i] = ele;
	for (i=0; i < n+1; i++)
	{
		char a[sizeof(int)];
		snprintf(a, sizeof(int), "%d", val[i]);
		cval[i] = (char*)malloc(sizeof(a));
		strcpy(cval[i], a);
	}
	cval[i]=NULL;
	pid=fork();
	printf("\nstart process and its child process ID is :%d and parent id is: %d", getpid(),
	getppid());
	if(pid==0)
	{
	execve(argv[1], cval, newenviron);
```

```
        perror("Error in execve call...");
        }
}
```

```
#include <stdio.h> //CHILD
#include <stdlib.h>
#include <string.h>
int main(int argc, char *argv[],char *en[])
{
int i,j,c,ele;
int arr[argc];
for (j = 0; j < argc-1; j++)
{
int n=atoi(argv[j]);
arr[j]=n;
}
ele=atoi(argv[j]);
i=0;
j=argc-1;
c=(i+j)/2;
while(arr[c]!=ele && i<=j)
{
if(ele > arr[c]) i = c+1;
else
j = c-1;
c = (i+j)/2;
}
if(i<=j)
printf("\nElement Found in the given Array...!!!\n");
else
printf("\nElement Not Found in the given Array...!!!\n");
}
```

EXP 3 SJF

```
#include<stdio.h>
int main()
{
int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
float avg_wt,avg_tat;
printf("Enter number of process:");
scanf("%d",&n);
printf("\nEnter Burst Time:\n");
for(i=0;i<n;i++)
{
printf("p%d:",i+1);
scanf("%d",&bt[i]);
p[i]=i+1;
}
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(bt[j]<bt[pos])
pos=j;
}
temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;
temp=p[i];
p[i]=p[pos];
p[pos]=temp;
}
wt[0]=0;
for(i=1;i<n;i++)
{
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
total+=wt[i];
}
avg_wt=(float)total/n;
total=0;
printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];
total+=tat[i];
printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
}
avg_tat=(float)total/n;
printf("\n\nAverage Waiting Time=%f",avg_wt);
```

```
printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

EXP 3 ROUND ROBIN

```
#include<stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
int main()
{
int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];
float avg_wt, avg_tat;
printf(" Total number of process in the system: ");
scanf("%d", &NOP);
y = NOP;
for(i=0;i<NOP; i++)
{
printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
printf(" Arrival time is: \t");
scanf("%d", &at[i]);
printf(" \nBurst time is: \t");
scanf("%d", &bt[i]);
temp[i] = bt[i];
}
printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
printf("\n Process No\t\t Burst Time \t\t TAT \t\t Waiting Time ");
for(sum=0, i = 0; y!=0; )
{
if(temp[i] <= quant && temp[i] > 0)
{
sum = sum + temp[i];
temp[i] = 0;
count=1;
}
else if(temp[i] > 0)
{
temp[i] = temp[i] - quant;
sum = sum + quant;
}
if(temp[i]==0 && count==1)
{
y--;
printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum- at[i]-bt[i]);
wt = wt+sum-at[i]-bt[i];
tat = tat+sum-at[i];
count =0;
}
if(i==NOP-1)
```

```
{
i=0;
}
else if(at[i+1]<=sum)
{
i++;
}
else
{
i=0;
}
}
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\n Average Turn Around Time: \t%f", avg_wt);
printf("\n Average Waiting Time: \t%f",
avg_tat);
}
```

EXP 4 Thread synchronization and mutual exclusion

```
#include<stdio.h>
#include<pthread.h>
#include<unistd.h>
pthread_mutex_t wr, mutex;
int a=10, readcount=0;
void * reader(void *arg){
long int num;
num=(long int) arg;
pthread_mutex_lock(&mutex);
readcount++;
pthread_mutex_unlock(&mutex);
if(readcount==1)
pthread_mutex_lock(&wr);
printf("Reader %ld is in critical section", num);
printf("Reader %ld is reading data %d", num,a);
sleep(1);
pthread_mutex_lock(&mutex);
readcount--;
pthread_mutex_unlock(&mutex);
if(readcount==0)
pthread_mutex_lock(&wr);
printf("\n Reader %ld left the critical section", num);
return NULL;
}
void * writer (void *arg){
long int num;
num=(long int) arg;
pthread_mutex_lock(&wr);
```

```
printf("\n Writer %ld is in critical section", num);
printf("\n Writer %ld has written dataas %d", num,++a);
sleep(1);
pthread_mutex_unlock(&wr);
printf("\n Writer %ld is in critical section", num);
return NULL;
}
int main() {
pthread_t r[10],w[10];long int i,j;
int nor, now;
pthread_mutex_init(&wr,NULL);
pthread_mutex_init(&mutex,NULL);

printf("Enter number of reader and writter");
scanf("%d %d",&nor,&now);
for(i=0;i<nor;i++)
{
pthread_create(&r[i],NULL,reader,(void *)i); }
for(j=0;j<now;j++)
{
pthread_create(&w[j],NULL,reader,(void *)j);
}
for(i=0;i<nor;i++)
{
pthread_join(r[i],NULL);
}
for(j=0;j<now;j++)
{
pthread_join(w[j],NULL);
}
return 0;
}
```

EXP 5 Deadlock Avoidance Algorithm: Bankers Algorithm

```
#include <stdio.h>
int main()
{
int n, m, i, j, k;
n = 5; m = 3;
int alloc[5][3] = { { 0, 1, 0 },
{ 2, 0, 0 },
{ 3, 0, 2 },
{ 2, 1, 1 },
{ 0, 0, 2 } };
int max[5][3] = { { 7, 5, 3 },
{ 3, 2, 2 },
{ 9, 0, 2 },
{ 2, 2, 2 },
{ 4, 3, 3 } };
int avail[3] = { 3, 3, 2 };
int f[n], ans[n], ind = 0; for (k = 0; k < n; k++) { f[k] = 0;
}
int need[n][m];
for (i = 0; i < n; i++) { for (j = 0; j < m; j++)
need[i][j] = max[i][j] - alloc[i][j];
}
int y = 0;
for (k = 0; k < 5; k++) { for (i = 0; i < n; i++) { if (f[i] == 0) {
int flag = 0;
for (j = 0; j < m; j++) { if (need[i][j] > avail[j]){ flag = 1;
break;
}
}
if (flag == 0) { ans[ind++] = i;
for (y = 0; y < m; y++) avail[y] += alloc[i][y];
f[i] = 1;
}
}
}
}
printf("Following is the SAFE Sequence\n");
for (i = 0; i < n - 1; i++)
printf(" P%d ->", ans[i]);
printf(" P%d", ans[n - 1]);
return (0);
}
```

EXP 6 FCFS

```
#include <stdio.h>
int main() {
int referenceString[10], pageFaults = 0, m, n, s, pages, frames;
printf("\nEnter the number of Pages:\t");
scanf("%d", & pages);
printf("\nEnter reference string values:\n");
for (m = 0; m < pages; m++)
{
printf("Value No. [%d]:\t", m + 1);
scanf("%d", & referenceString[m]);
}
printf("\n What are the total number of frames:\t");
{ scanf("%d", & frames);
}
int temp[frames];
for (m = 0; m < frames; m++) {
temp[m] = -1;
}
for (m = 0; m < pages; m++)
{
s = 0;
for (n = 0; n < frames; n++)
{
if (referenceString[m] == temp[n])
{ s++; pageFaults--;
}
}
pageFaults++;
if ((pageFaults <= frames) && (s == 0))
{
temp[m] = referenceString[m];
}
else if (s == 0)
{
temp[(pageFaults - 1) % frames] = referenceString[m];
}
printf("\n");
for (n = 0; n < frames; n++)
{
 printf("%d\t", temp[n]);
}
}
printf("\nTotal Page Faults:\t%d\n", pageFaults);
return 0;
}
```

EXP 6 LRU

```
#include<stdio.h>
int findLRU(int time[], int n)
{
int i, minimum = time[0], pos = 0;
for (i = 1; i < n; ++i) {
if (time[i] < minimum)
{
 minimum = time[i];
 pos = i;
}
}
return pos;
}
int main() {
int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j,
pos,
faults = 0;
printf("Enter number of frames: ");
scanf("%d", & no_of_frames); printf("Enter number of pages: ");
scanf("%d", & no_of_pages); printf("Enter reference string: ");
for (i = 0; i < no_of_pages; ++i)
{
scanf("%d", & pages[i]);
}
for (i = 0; i < no_of_frames; ++i)
{
frames[i] = -1;
}
for (i = 0; i < no_of_pages; ++i)
{
flag1 = flag2 = 0;
for (j = 0; j < no_of_frames; ++j)
{
if (frames[j] == pages[i])
 {
 counter++;
time[j] = counter;
flag1 = flag2 = 1;
break;
}
}
if (flag1 == 0) {
for (j = 0; j < no_of_frames; ++j)
{
if (frames[j] == -1)
{
counter++; faults++;
frames[j] = pages[i];
```

```
time[j] = counter;
flag2 = 1;
break;
}
}
}
if (flag2 == 0) {
pos = findLRU (time, no_of_frames);
counter++;
faults++;
frames[pos] = pages[i];
time[pos] = counter;
}
printf("\n");
for (j = 0; j < no_of_frames; ++j)
{
printf("%d\t", frames[j]);
}
}
printf ("\n\nTotal Page Faults = %d", faults);
return 0;
}
```

EXP 6 OPTIMAL

```
#include<stdio.h>
int main() {
int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j, k,
pos, max,
faults = 0;
printf("Enter number of frames: ");
scanf("%d", & no_of_frames);
printf("Enter number of pages: ");
scanf("%d", & no_of_pages);
printf("Enter page reference string: ");
for (i = 0; i < no_of_pages; ++i)
{
scanf("%d", & pages[i]);
}
for (i = 0; i < no_of_frames; ++i)
{
frames[i] = -1;
}
for (i = 0; i < no_of_pages; ++i)
{
flag1 = flag2 = 0;
for (j = 0; j < no_of_frames; ++j)
{
 if (frames[j] == pages[i])
 {
flag1 = flag2 = 1;
break;
}
}
if (flag1 == 0)
{
for (j = 0; j < no_of_frames; ++j)
{
 if (frames[j] == -1)
 {
faults++;
frames[j] = pages[i];
flag2 = 1;
break;
}
}
}
if (flag2 == 0)
{
flag3 = 0;
for (j = 0; j < no_of_frames; ++j)
{
 temp[j] = -1;
```

```
for (k = i + 1; k < no_of_pages; ++k)
{ if (frames[j] == pages[k])
{
temp[j] = k;
break;
}
}
}
for (j = 0; j < no_of_frames; ++j)
{
 if (temp[j] == -1)
 {
pos = j; flag3 = 1;
break;
}
}
if (flag3 == 0)
{
max = temp[0];
 pos = 0;
for (j = 1; j < no_of_frames; ++j)
{
if (temp[j] > max)
{
max = temp[j];
pos = j;
}
}
}
frames[pos] = pages[i];
faults++;
}
printf("\n");
for (j = 0; j < no_of_frames; ++j)
{
printf("%d\t", frames[j]);
}
}
printf("\n\nTotal Page Faults = %d", faults);
return 0;
}
```

EXP 7 INTER-PROCESS COMMUNICATION USING SHARED MEMORY

//sharedMemory.c

```
#include <cstddef>
#include <sys/ipc.h>
#include <sys/shm.h>
#define PROJECT_ID 220
#define READ_BY_CLIENT 0
#define WRITTEN_BY_SERVER 1
#define ARRAY_LENGTH 5
typedef struct SharedMemory {
int status;
int array[ARRAY_LENGTH];
}SharedMemory;
key_t getKey() {
return ftok(".", PROJECT_ID);
}
int shm_init() {
return shmget(getKey(), sizeof(SharedMemory), IPC_CREAT | 0666);
}
SharedMemory * attach(int shm_id) {
return (SharedMemory * ) shmat(shm_id, NULL, 0);
}
int detach(SharedMemory * shm) {
return shmdt((void * ) shm);
}
```

// reader.c

```
#include <stdio.h>
#include <stdlib.h>
#include "SharedMemory.c"
int main() {
int shm_id, i;
if ((shm_id = shm_init()) == -1) {
perror("Error occured while initialising Shared Memory\n");
exit(-1);
}
SharedMemory * mSharedMemory = attach(shm_id);
if (mSharedMemory -> status == READ_BY_CLIENT) {
printf("Server hasn't written value yet\n");
exit(-1);
}
printf("Printing %d Numbers\n", ARRAY_LENGTH);
for (i = 0; i < ARRAY_LENGTH; i++) {
printf("%d\n", mSharedMemory -> array[i]);
}
mSharedMemory -> status = READ_BY_CLIENT;
if (detach(mSharedMemory) == -1) {
perror("Error occured while detaching Shared memory\n");
exit(-1);
```

```
}
}
```

//Writer.cc

```
#include <stdio.h>
#include <stdlib.h>
#include "SharedMemory.c"
int main() {
int shm_id, i;
if ((shm_id = shm_init()) == -1) {
perror("Error occured while initialising Shared Memory\n");
exit(-1);}
SharedMemory * mSharedMemory = attach(shm_id);
if (mSharedMemory -> status == WRITTEN_BY_SERVER) {
printf("Client hasn't read value yet\n");
exit(-1);
}
printf("Enter %d Numbers\n", ARRAY_LENGTH);
for (i = 0; i < ARRAY_LENGTH; i++) {
scanf("%d", & mSharedMemory -> array[i]);
}
mSharedMemory -> status = WRITTEN_BY_SERVER;
if (detach(mSharedMemory) == -1) {
perror("Error occured while detaching Shared memory\n");
exit(-1);
}
char c;
printf("Press any key to exit\n");
scanf(" %c", & c);
}
```

EXP 8 SSTF

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int RQ[100], i, n, TotalHeadMoment = 0, initial, count = 0;
printf("Enter the number of Requests\n");
scanf("%d", & n);
printf("Enter the Requests sequence\n");
for (i = 0; i < n; i++)
scanf("%d", & RQ[i]);
printf("Enter initial head position\n");
scanf("%d", & initial);
while (count != n)
{
int min = 1000, d, index;
for (i = 0; i < n; i++)
{
d = abs(RQ[i] - initial);
if (min > d)
{
min = d;
index = i;
}
}
TotalHeadMoment = TotalHeadMoment + min;
initial = RQ[index];
RQ[index] = 1000; count++;
}
printf("Total head movement is %d", TotalHeadMoment);
return 0;
}
```

EXP 8 SCAN

```
#include<stdio.h>
int main()
{
int i, j, sum = 0, n; int d[20];
int disk;
int temp, max;
int dloc;
printf("enter number of location\t");
scanf("%d", & n);
printf("enter position of head\t");
scanf("%d", & disk);
printf("enter elements of disk queue\n");
for (i = 0; i < n; i++)
{
scanf("%d", & d[i]);
}
d[n] = disk; n = n + 1;
for (i = 0; i < n; i++)
{
for (j = i; j < n; j++)
{ if (d[i] > d[j]) {
temp = d[i];
d[i] = d[j];
d[j] = temp;
}
}
}
max = d[n];
for (i = 0; i < n; i++)
{
if (disk == d[i]) {
dloc = i;
break;
}
}
for (i = dloc; i >= 0; i--)
{
printf("%d -->", d[i]);
}
printf("0 -->");
for (i = dloc + 1; i < n; i++)
{
printf("%d-->", d[i]);
}
sum = disk + max;
printf("\nmovement of total cylinders %d", sum);
return 0;
}
```

EXP 8 C-LOOK

```
#include<stdio.h>
#include<stdlib.h>
int main () {
int RQ[100], i, j, n, TotalHeadMoment = 0, initial, size, move;
printf("Enter the number of Requests\n");
scanf("%d", & n);
printf("Enter the Requests sequence\n");
for (i = 0; i < n; i++)
scanf("%d", & RQ[i]);
printf("Enter initial head position\n");
scanf("%d", & initial);
printf("Enter total disk size\n");
scanf("%d", & size);
printf("Enter the head movement direction for high 1 and for low 0 \n");
scanf("%d", & move);
for (i = 0; i < n; i++)
{
for (j = 0; j < n - i - 1; j++)
{
if (RQ[j] > RQ[j + 1])
{
int temp;
temp = RQ[j];
RQ[j] = RQ[j + 1];
RQ[j + 1] = temp;
}
}
}
int index;
for (i = 0; i < n; i++)
{ if (initial < RQ[i])
{
index = i;
break;
}
}
if (move == 1)
{
for (i = index; i < n; i++)
{
TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);
initial = RQ[i];
}
for (i = 0; i < index; i++)
{
TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);
initial = RQ[i];
}
```

```
}
else {
for (i = index - 1; i >= 0; i--)
{
TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);
initial = RQ[i];
}
for (i = n - 1; i >= index; i--)
{
TotalHeadMoment = TotalHeadMoment + abs(RQ[i] - initial);
initial = RQ[i];
}
}
printf("Total head movement is %d", TotalHeadMoment);
return 0;
}
```