Slip 28

q.1

#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);

// logic for C-look disk scheduling

/\*logic for sort the request array \*/

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 movement is towards high value

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];

}

}

// if movement is towards low value

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;

}

Q.2

#include <mpi.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define n 10

int a[] = {1, 3, 5, 10, 12, 20, 4, 50, 100, 1000};

int a2[1000];

int main(int argc, char\* argv[])

{

int pid, np, elements\_per\_process, n\_elements\_recieved;

MPI\_Status status;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &pid);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &np);

if (pid == 0) {

int index, i;

elements\_per\_process = n / np;

if (np > 1) {

for (i = 1; i < np - 1; i++) {

index = i \* elements\_per\_process;

MPI\_Send(&elements\_per\_process, 1, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

MPI\_Send(&a[index],elements\_per\_process, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

}

index = i \* elements\_per\_process;

int elements\_left = n - index;

MPI\_Send(&elements\_left, 1, MPI\_INT,i , 0, MPI\_COMM\_WORLD);

MPI\_Send(&a[index], elements\_left, MPI\_INT, i, 0,MPI\_COMM\_WORLD);

}

int sum = 0;

for (i = 0; i < elements\_per\_process; i++)

sum += a[i];

int tmp;

for (i = 1; i < np; i++) {

MPI\_Recv(&tmp, 1, MPI\_INT, MPI\_ANY\_SOURCE, 0,MPI \_COMM\_WORLD, &status);

int sender = status.MPI\_SOURCE;

sum += tmp;

}

printf("Sum of array is : %d\n", sum);

}

else {

MPI\_Recv(&n\_elements\_recieved, 1, MPI\_INT, 0, 0,MPI\_COMM\_WORLD, &status);

MPI\_Recv(&a2, n\_elements\_recieved, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, &status);

int partial\_sum = 0;

for (int i = 0; i < n\_elements\_recieved; i++)

partial\_sum += a2[i];

MPI\_Send(&partial\_sum, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

MPI\_Finalize();

return 0;

}