**Slip no 2**

q.1

#include<stdio.h>

#include<stdlib.h>

void main()

{

int f[50],p,i,st,len,j,c,k,a;

for(i=0;i<50;i++)

f[i]=0;

printf("Enter how many blocks already allocated:");

scanf("%d",&p);

printf("Enter blocks already allocated:");

for(i=0;i<p;i++)

{

scanf("%d",&a);

f[a]=1;

}

x:printf("Enter index starting block and length:");

scanf("%d%d",&st,&len);

k=len;

if(f[st]==0)

{

for(j=st;j<(st+k);j++)

{

if(f[j]==0)

{

f[j]=1;

printf("%d------>%d\n",j,f[j]);

}

else

{

printf("%dBlock is already allocated\n",j);

k++;

}

}

}

else

printf("%d starting block is already allocated\n",st);

printf("Do you want to enter more file(Yes-1/No-0)");

scanf("%d",&c);

if(c==1)

goto x;

else

exit(0);

getch();

}

**q.2**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int queue[100], n, head, i, j, seek=0;

float avg;

printf("Enter the number of requests: ");

scanf("%d",&n);

printf("Enter the request queue:\n");

for(i=1;i<=n;i++)

scanf("%d",&queue[i]);

printf("Enter the initial head position: ");

scanf("%d",&head);

queue[0]=head;

printf("\nHead movement:\n");

for(j=0;j<=n-1;j++)

{

printf("%d --> ",queue[j]);

seek+=abs(queue[j+1]-queue[j]);

}

printf("%d\n",queue[n]);

avg=seek/(float)n;

printf("\nAverage seek time = %0.2f\n",avg);

return 0;

}

**Q2**

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h>

#define ARRAY\_SIZE 1000

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

int rank, size;

int array[ARRAY\_SIZE];

int sum = 0, local\_sum = 0;

int i;

MPI\_Init(&argc, &argv);

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

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

srand(rank);

for (i = 0; i < ARRAY\_SIZE; i++) {

array[i] = rand() % 100;

}

int start = rank \* (ARRAY\_SIZE / size);

int end = (rank + 1) \* (ARRAY\_SIZE / size);

if (rank == size - 1) {

end = ARRAY\_SIZE;

}

for (i = start; i < end; i++) {

local\_sum += array[i];

}

MPI\_Reduce(&local\_sum, &sum, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (rank == 0) {

printf("Sum of %d random numbers: %d\n", ARRAY\_SIZE, sum);

}

MPI\_Finalize();

return 0;

}