

[Date]

Submitted to:

Name: Suman Saha

Assistant Professor

Dept. of CSE

at Bangladesh University of Business and Technology.

Submitted by:

Name: Syeda Nowshin Ibnat

ID: 17183103020

Intake: 39

Section: 01

Program:B.Sc. in CSE

Semester: Fall 20-21

Lab Final(part-2)

Course Code: CSE 310

Course Title: Operating Systems

Date of Submission: 31/03/2021

**1 no**

**Solution:**

**Source code:**

#include<stdio.h>

int main() {

int fragment[20],b[20],p[20],i,j,nb,np,temp,lowest=9999;

static int barray[20],parray[20];

printf("\n\t\t\tMemory Management Scheme - Best Fit");

printf("\nEnter the number of blocks:");

scanf("%d",&nb);

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

scanf("%d",&np);

printf("\nEnter the size of the blocks:-\n");

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

printf("Block no-%d:",i);

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

printf("\nEnter the size of the processes :-\n");

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

printf("Process no-%d:",i);

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

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

for(j=1; j<=nb; j++) {

if(barray[j]!=1) {

temp=b[j]-p[i];

if(temp>=0)if(lowest>temp) {

parray[i]=j;

lowest=temp; } } }

fragment[i]=lowest;

barray[parray[i]]=1;

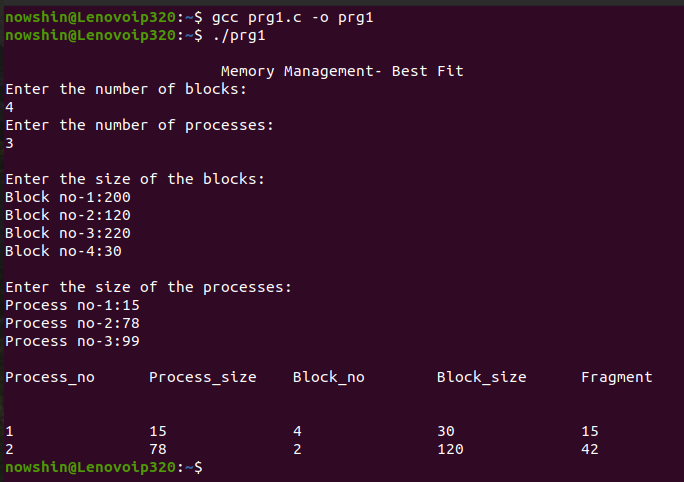
lowest=10000; }

printf("\nProcess\_no\tProcess\_size\tBlock\_no\tBlock\_size\tFragment");

for(i=1; i<=np && parray[i]!=0; i++)

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,p[i],parray[i],b[parray[i]],fragment[i]); }

**Output:**



**2 no**

**Solution:**

**Source Code:**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main() {

int alloc[10][10];

int request[10][10],avail[10],r[10],w[10];

static int mark[20];

int i,j,np,nr;

FILE\* file = fopen ("in.txt", "r");

fscanf (file, "%d", &np);

fscanf (file, "%d", &nr);

for(i=0; i<nr; i++) {

fscanf (file, "%d", &r[i]); }

// read the allocation matrix

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

for(j=0; j<nr; j++)

fscanf (file, "%d", &alloc[i][j]);

// read the request matrix

for(i=0; i<np; i++) for(j=0; j<nr; j++)

fscanf (file, "%d", &request[i][j]);

// solution starts here

printf("\n");

//marking processes with zero allocation

for(i=0; i<np; i++) {

int count=0;

for(j=0; j<nr; j++) {

if(alloc[i][j]==0)

count++;

else

break; }

if(count==nr)

mark[i]=1; }

for(j=0; j<nr; j++)

w[j]=avail[j];

//mark processes with request less than or equal to W

for(i=0; i<np; i++) {

int canbeprocessed=0;

if(mark[i]!=1) {

for(j=0; j<nr; j++) {

if(request[i][j]<=w[j])

canbeprocessed=1;

else {

canbeprocessed=0;

break; } }

if(canbeprocessed) {

mark[i]=1;

for(j=0; j<nr; j++) w[j]+=alloc[i][j]; } }}

printf("printing the contents read from file: \n");

printf("\n\nNumer of processes: %d",np);

printf("\n\nNumer of resources: %d",nr);

printf("\n\nAvailbaility vector\n");

for(i=0; i<nr; i++) {

printf("%d ",r[i]); }

printf("\n");

printf("\nAllocation Matrix\n");

for(i=0; i<np; i++) {

for(j=0; j<nr; j++)

printf("%d ", alloc[i][j]);

printf("\n"); }

printf("\n");

printf("\nRequest Matrix\n");

for(i=0; i<np; i++) {

for(j=0; j<nr; j++)

printf("%d ", request[i][j]);

printf("\n");}

//checking for unmarked processes

int deadlock=0;

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

if(mark[i]!=1)

deadlock=1;

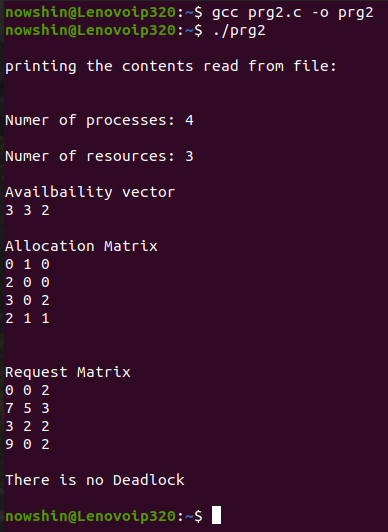
if(deadlock)

printf("\nThere exist a Deadlock\n\n");

else printf("\nThere is no Deadlock\n\n");

fclose (file); }

**Output:**



**3 no**

**Solution:**

**Source Code:**

#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("number of frames: ");

scanf("%d", &no\_of\_frames);

printf("number of pages: ");

scanf("%d", &no\_of\_pages);

printf("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; }

**Output:**

