7a

#include<stdio.h>

int lfu();

#define SIZE 3

int full=0;//To check whether all frames are filled

int a[21],n;//To take the input

int frame[SIZE];

int ctr[SIZE]={0};

static int f;

int repptr;

int count=0;

int display()

{int i;

printf("\nThe elements in the frame are\n");

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

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

}

int Longestopt()

{int i,max;

max=0;//The increment of counter value here is same as that for of LFU

for(i=0;i<SIZE;i++)//The page with maximum frequency is selected

if(ctr[max]<ctr[i])

max=i;

repptr=max;

return repptr;

}

int Pagerep(int ele)

{

int temp;

repptr=Longestopt();

temp=frame[repptr];

frame[repptr]=ele;

ctr[repptr]=1;

return temp;

}

int Pagefault(int ele)

{if(full!=SIZE)

{ctr[full]++;

frame[full++]=ele;

}

else

printf("The page replaced is %d",Pagerep(ele));

}

int Search(int ele)

{int i,flag;

flag=0;

if(full!=0)

{

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

if(ele==frame[i])

{ flag=1;ctr[i]++;

break;

}}

return flag;

}

int mfu()

{int i;

FILE \*fp;

fp=fopen("Input.txt","r");

printf("The number of elements in the reference string are :");

fscanf(fp,"%d",&n);

printf("%d",n);

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

fscanf(fp,"%d",&a[i]);

printf("\nThe elements present in the string are\n");

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

printf("%d ",a[i]);

printf("\n\n");

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

{f=i;

if(Search(a[i])!=1)

{Pagefault(a[i]);

display();

count++;

}

}

printf("\nThe number of page faults are %d\n",count);

return 0;

}

int lfu()

{

int total\_frames, total\_pages, hit = 0;

int pages[25], frame[10], arr[25], time[25];

int m, n, page, flag, k, minimum\_time, temp;

printf("Enter Total Number of Pages:\t");

scanf("%d", &total\_pages);

printf("Enter Total Number of Frames:\t");

scanf("%d", &total\_frames);

for(m = 0; m < total\_frames; m++)

{

frame[m] = -1;

}

for(m = 0; m < 25; m++)

{

arr[m] = 0;

}

printf("Enter Values of Reference String\n");

for(m = 0; m < total\_pages; m++)

{

printf("Enter Value No.[%d]:\t", m + 1);

scanf("%d", &pages[m]);

}

printf("\n");

for(m = 0; m < total\_pages; m++)

{

arr[pages[m]]++;

time[pages[m]] = m;

flag = 1;

k = frame[0];

for(n = 0; n < total\_frames; n++)

{

if(frame[n] == -1 || frame[n] == pages[m])

{

if(frame[n] != -1)

{

hit++;

}

flag = 0;

frame[n] = pages[m];

break;

}

if(arr[k] > arr[frame[n]])

{

k = frame[n];

}

}

if(flag)

{

minimum\_time = 25;

for(n = 0; n < total\_frames; n++)

{

if(arr[frame[n]] == arr[k] && time[frame[n]] < minimum\_time)

{

temp = n;

minimum\_time = time[frame[n]];

}

}

arr[frame[temp]] = 0;

frame[temp] = pages[m];

}

for(n = 0; n < total\_frames; n++)

{

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

}

printf("\n");

}

printf("Page Hit:\t%d\n", hit);

return 0;

}

int main()

{

int ch;

do{

printf("for lfu press '1'\n");

printf("for mfu press '2'\n");

printf("enter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

lfu();

break;

case 2:

mfu();

break;

default:

printf("invalid");

break;

}

}while(1);

}

7b

#include<iostream>

#include<algorithm>

using namespace std;

struct process

{

int at,bt,pr,pno;

};

#define totalprocess 5

process proc[50];

int mat[10][6];

//int sjf();

//void swap(int \*a, int \*b);

//void arrangeArrival(int num, int mat[][6]);

//void completionTime(int num, int mat[][6]);

//int priority();

//void get\_wt\_time(int wt[]);

//bool comp(process a,process b);

//void get\_tat\_time(int tat[],int wt[]);

//void findgc();

void swap(int \*a, int \*b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void arrangeArrival(int num, int mat[][6])

{

for(int i=0; i<num; i++)

{

for(int j=0; j<num-i-1; j++)

{

if(mat[j][1] > mat[j+1][1])

{

for(int k=0; k<5; k++)

{

swap(mat[j][k], mat[j+1][k]);

}

}

}

}

}

void completionTime(int num, int mat[][6])

{

int temp, val;

mat[0][3] = mat[0][1] + mat[0][2];

mat[0][5] = mat[0][3] - mat[0][1];

mat[0][4] = mat[0][5] - mat[0][2];

for(int i=1; i<num; i++)

{

temp = mat[i-1][3];

int low = mat[i][2];

for(int j=i; j<num; j++)

{

if(temp >= mat[j][1] && low >= mat[j][2])

{

low = mat[j][2];

val = j;

}

}

mat[val][3] = temp + mat[val][2];

mat[val][5] = mat[val][3] - mat[val][1];

mat[val][4] = mat[val][5] - mat[val][2];

for(int k=0; k<6; k++)

{

swap(mat[val][k], mat[i][k]);

}

}

}

int sjf()

{

int num, temp;

cout<<"Enter number of Process: ";

cin>>num;

cout<<"...Enter the process ID...\n";

for(int i=0; i<num; i++)

{

cout<<"...Process "<<i+1<<"...\n";

cout<<"Enter Process Id: ";

cin>>mat[i][0];

cout<<"Enter Arrival Time: ";

cin>>mat[i][1];

cout<<"Enter Burst Time: ";

cin>>mat[i][2];

}

cout<<"Before Arrange...\n";

cout<<"Process ID\tArrival Time\tBurst Time\n";

for(int i=0; i<num; i++)

{

cout<<mat[i][0]<<"\t\t"<<mat[i][1]<<"\t\t"<<mat[i][2]<<"\n";

}

arrangeArrival(num, mat);

completionTime(num, mat);

cout<<"Final Result...\n";

cout<<"Process ID\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n";

for(int i=0; i<num; i++)

{

cout<<mat[i][0]<<"\t\t"<<mat[i][1]<<"\t\t"<<mat[i][2]<<"\t\t"<<mat[i][4]<<"\t\t"<<mat[i][5]<<"\n";

}

}

/\*

Writing comparator function to sort according to priority if

arrival time is same

\*/

bool comp(process a,process b)

{

if(a.at == b.at)

{

return a.pr<b.pr;

}

else

{

return a.at<b.at;

}

}

// Using FCFS Algorithm to find Waiting time

void get\_wt\_time(int wt[])

{

// declaring service array that stores cumulative burst time

int service[50];

// Initilising initial elements of the arrays

service[0] = proc[0].at;

wt[0]=0;

for(int i=1;i<totalprocess;i++)

{

service[i]=proc[i-1].bt+service[i-1];

wt[i]=service[i]-proc[i].at;

// If waiting time is negative, change it into zero

if(wt[i]<0)

{

wt[i]=0;

}

}

}

void get\_tat\_time(int tat[],int wt[])

{

// Filling turnaroundtime array

for(int i=0;i<totalprocess;i++)

{

tat[i]=proc[i].bt+wt[i];

}

}

void findgc()

{

//Declare waiting time and turnaround time array

int wt[50],tat[50];

double wavg=0,tavg=0;

// Function call to find waiting time array

get\_wt\_time(wt);

//Function call to find turnaround time

get\_tat\_time(tat,wt);

int stime[50],ctime[50];

stime[0] = proc[0].at;

ctime[0]=stime[0]+tat[0];

// calculating starting and ending time

for(int i=1;i<totalprocess;i++)

{

stime[i]=ctime[i-1];

ctime[i]=stime[i]+tat[i]-wt[i];

}

cout<<"Process\_no\tStart\_time\tComplete\_time\tTurn\_Around\_Time\tWaiting\_Time"<<endl;

// display the process details

for(int i=0;i<totalprocess;i++)

{

wavg += wt[i];

tavg += tat[i];

cout<<proc[i].pno<<"\t\t"<<

stime[i]<<"\t\t"<<ctime[i]<<"\t\t"<<

tat[i]<<"\t\t\t"<<wt[i]<<endl;

}

// display the average waiting time

//and average turn around time

cout<<"Average waiting time is : ";

cout<<wavg/(float)totalprocess<<endl;

cout<<"average turnaround time : ";

cout<<tavg/(float)totalprocess<<endl;

}

int priority()

{

int arrivaltime[] = { 1, 2, 3, 4, 5 };

int bursttime[] = { 3, 5, 1, 7, 4 };

int priority[] = { 3, 4, 1, 7, 8 };

for(int i=0;i<totalprocess;i++)

{

proc[i].at=arrivaltime[i];

proc[i].bt=bursttime[i];

proc[i].pr=priority[i];

proc[i].pno=i+1;

}

//Using inbuilt sort function

sort(proc,proc+totalprocess,comp);

//Calling function findgc for finding Gantt Chart

findgc();

return 0;

}

int main()

{

int ch;

do{

cout<<"for sjf press '1'"<<endl;

cout<<"for priority scheduling press '2'"<<endl;

cout<"enter your choice: ";

cin>>ch;

switch(ch)

{

case 1:

sjf();

break;

case 2:

priority();

break;

default:

cout<<"invalid";

break;

}

}while(1);

}