//first fit

#include<stdio.h>

#define max 10

void main()

{ int block[max],process[max],j,m,i,n;

int allocate[max]={0};

int visited[max]={0};

printf("enter no of blocks and processes\n");

scanf("%d%d",&n,&m);

printf("Enter blocks\n");

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

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

printf("Enter process\n");

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

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

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

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

{ if(visited[j] == 0 && block[j] >= process[i])

{

allocate[j] = i;

visited[j] = 1;

break;

}

}

}printf("\nblock no.\tBlock\tProcess no\tprocess\n");

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

{ printf("\n%d\t\t%d",i+1,block[i]);

if(visited[i]==1)

printf("\t%d\t\t%d",allocate[i]+1,process[allocate[i]]);

else

printf("Not allocated");

}

}

//bestfit

#include<stdio.h>

void main()

{

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

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

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]);

}

//worstfit

#include<stdio.h>

void main()

{

int fragment[20],b[20],p[20],i,j,nb,np,temp,highest=0;

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

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(highest<temp)

{

parray[i]=j;

highest=temp;

}

}

}

fragment[i]=highest;

barray[parray[i]]=1;

highest=0;

}

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]);

}

//fifo

#include<stdio.h>

void main()

{ int i,j,n,a[50],frame[10],no,k,avail,count=0;

printf("\n ENTER THE NUMBER OF PAGES:\n");

scanf("%d",&n);

printf("\n ENTER THE PAGE NUMBER :\n");

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

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

printf("\n ENTER THE NUMBER OF FRAMES :");

scanf("%d",&no);

for(i=0;i<no;i++)

frame[i]= -1;

j=0;

printf("\tref string\t page frames\n");

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

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

avail=0;

for(k=0;k<no;k++)

if(frame[k]==a[i])

avail=1;

if (avail==0)

{ frame[j]=a[i];

j=(j+1)%no;

count++;

for(k=0;k<no;k++)

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

}

printf("\n");

}

printf("Page Fault Is %d",count);

}

//lru

#include<stdio.h>

#include<stdlib.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 nof, nop, frames[10], pages[30], counter=0, time[10], f1, f2, i, j, pos, faults=0;

printf("Enter number of frames: ");

scanf("%d", &nof);

printf("Enter number of pages: ");

scanf("%d", &nop);

printf("Enter reference string: ");

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

{

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

}

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

{

frames[i] = -1;

}

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

{

f1 = f2 = 0;

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

{

if(frames[j] == pages[i])

{

counter++;

time[j] = counter;

f1 = f2 = 1;

break;

}

}

if(f1 == 0)

{

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

{

if(frames[j] == -1)

{

counter++;

faults++;

frames[j] = pages[i];

time[j] = counter;

f2 = 1;

break;

}

}

}

if(f2 == 0)

{

pos = findLRU(time, nof);

counter++;

faults++;

frames[pos] = pages[i];

time[pos] = counter;

}

printf("\n");

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

{

printf("%d\t", frames[j]);

}

}

printf("\n\nTotal Page Faults = %d", faults);

return 0;

}

//fcfs

#include<stdio.h>

void main()

{

int n,a[10],b[10],t[10],w[10],g[10],i;

float att=0,awt=0;

for(i=0;i<10;i++)

{

a[i]=0; b[i]=0; w[i]=0; g[i]=0;

}

printf("enter the number of process");

scanf("%d",&n);

printf("enter the burst times");

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

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

printf("\nenter the arrival times");

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

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

g[0]=0;

for(i=0;i<10;i++)

g[i+1]=g[i]+b[i];

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

{ w[i]=g[i]-a[i];

t[i]=g[i+1]-a[i];

awt=awt+w[i];

att=att+t[i];

}

awt =awt/n;

att=att/n;

printf("\n\tprocess\tarrival time\tburst time\tcompletion time\tturn arround time\twaiting time\n");

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

{ printf("\tp%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",i,a[i],b[i],g[i+1],t[i],w[i]);

}

printf("the average waiting time is %f\n",awt);

printf("the average turn around time is %f\n",att);

}

//sjf

#include<stdio.h>

void main()

{

int n,a[10],b[10],t[10],w[10],g[10],time[10],i,j,temp;

float att=0,awt=0;

for(i=0;i<10;i++)

{

a[i]=0; b[i]=0; w[i]=0; g[i]=0;

}

printf("enter the number of process");

scanf("%d",&n);

printf("enter the burst times");

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

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

time[i]=b[i];

a[i]=0;

}

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

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

{ if(b[i]>b[i+1])

{ temp=b[i];

b[i]=b[i+1];

b[i+1]=temp;

}

}

}

g[0]=0;

for(i=0;i<10;i++)

g[i+1]=g[i]+b[i];

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

{ w[i]=g[i]-a[i];

t[i]=g[i+1]-a[i];

awt=awt+w[i];

att=att+t[i];

}

awt =awt/n;

att=att/n;

printf("\n\tprocess\tarrival time\tburst time\tcompletion time\tturn arround time\twaiting time\n");

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

{ printf("\tp%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",i,a[i],b[i+1],g[i+2],t[i+1],w[i+1]);

}

printf("the average waiting time is %f\n",awt);

printf("the average turn around time is %f\n",att);

}

//round robin

**#include<stdio.h>**

**#include<stdlib.h>**

**int main()**

**{**

**int count,j,n,time,remain,flag=0,time\_quantum;**

**int wait\_time=0,turnaround\_time=0,at[10],bt[10],rt[10];**

**printf("Enter Total Process:\t ");**

**scanf("%d",&n);**

**remain=n;**

**for(count=0;count<n;count++)**

**{**

**printf("Enter Arrival Time & Burst Time Process Number %d :",count+1);**

**scanf("%d",&at[count]);**

**scanf("%d",&bt[count]);**

**rt[count]=bt[count];**

**}**

**printf("Enter Time Quantum:\t");**

**scanf("%d",&time\_quantum);**

**printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");**

**for(time=0,count=0;remain!=0;)**

**{**

**if(rt[count]<=time\_quantum && rt[count]>0)**

**{**

**time+=rt[count];**

**rt[count]=0;**

**flag=1;**

**}**

**else if(rt[count]>0)**

**{**

**rt[count]-=time\_quantum;**

**time+=time\_quantum;**

**}**

**if(rt[count]==0 && flag==1)**

**{**

**remain--;**

**printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);**

**wait\_time+=time-at[count]-bt[count];**

**turnaround\_time+=time-at[count];**

**flag=0;**

**}**

**if(count==n-1)**

**count=0;**

**else if(at[count+1]<=time)**

**count++;**

**else**

**count=0;**

**}**

**printf("\nAverage Waiting Time= %f\n",wait\_time\*1.0/n);**

**printf("Avg Turnaround Time = %f",turnaround\_time\*1.0/n);**

**return 0;**

**}**

//bankers

#bankers

allo=[]

maxm=[]

need=[]

ava=[]

r=int(input("Enter no of resources"))

p=int(input("Enter no of processes"))

comp=[False]\*p

#allocation

print("For allocation \n")

for i in range(p):

process=[]

print("For process ",i+1)

for j in range(r):

process.append(int(input(f"Enter inst for resource {j+1} ")))

allo.append(process)

print("For maximum \n")

for i in range(p):

process=[]

print("For process ",i+1)

for j in range(r):

process.append(int(input(f"Enter max inst for resource {j+1} ")))

maxm.append(process)

for i in range(p):

process=[]

for j in range(r):

process.append(maxm[i][j]-allo[i][j])

need.append(process)

for i in range(r):

ava.append(int(input(f"Enter available inst for resource {j+1} ")))

print(need)

#safety

flag=0

while(not flag):

for i in range(p):

check=0

flag=1

if(not comp[i]):

flag=0

for j in range(r):

if (need[i][j]<=ava[j]):

check=True

else:

check=False

break

if(check):

comp[i]=True

for j in range(r):

ava[j]=ava[j]+allo[i][j]

print(ava)

print(f"Process {i+1}")

/////////////////////////////////////////////////////////////////////////////////////////////////

**#QUESTION BANK PROBLEM 3 :**

**CHECK AND DELETE DIRECTORY**

**$vi qbprg3**

**clear**

**echo "ENTER GIVEN FILE PATH"**

**read path**

**if [[ -d $path ]]**

**then**

**echo "$path IS A DIRECTORY"**

**rmdir $path**

**else**

**echo"$path IS NOT A DIRECTORY"**

**fi**

**#QUESTION BANK PROBLEM 14 :**

**ODDEVEN AND PALINDROME**

**$vi qbprg14**

**clear**

**echo "ENTER THE NUMBER"**

**read no**

**z=$(($no % 2))**

**if [ $z -eq 0 ]**

**then**

**echo "EVEN NUMBER"**

**exit**

**else**

**echo "ODD NUMBER"**

**fi**

**clear**

**echo "ENTER THE STRING"**

**read string**

**echo**

**gnirts=`echo $string | rev`**

**if [ "$string" == "$gnirts" ]**

**then**

**echo "PALINDROME CHE"**

**else**

**echo "PALINDROME NATHI"**

**fi**

**#QUESTION BANK PROBLEM 18 :**

**CALCULATOR**

**$vi prg18**

**clear**

**a=$1**

**b=$3**

**ch=$2**

**if [ "$#" -lt 3 -o "$#" -gt 3 ]**

**then**

**echo "INVALID NUMBER OF ARGUMENTS"**

**exit**

**else**

**case $ch in**

**+) result=`echo $a + $b | bc` ;;**

**-) result=`echo $a - $b | bc` ;;**

**\\*) result=`echo $a \\* $b | bc` ;;**

**/) result=`echo $a / $b | bc` ;;**

**\*)echo "INVALID INPUT"**

**exit**

**;;**

**esac**

**fi**

**echo "RESULT = $result"**

**# QUESTION BANK QUESTION 19 :**

**SUM AND NEED ARGUEMENT**

**$vi qbprg19**

**clear**

**if [ $# -ne 2 ]**

**then**

**echo "INSUFFICIENT ARGUMENTS PASSED"**

**else**

**echo "SUM = `expr $1 + $2`"**

**fi**

**#QUESTION BANK PROGRAM 20 :**

**REVERSE NUMBER**

**$vi qbprg20**

**clear**

**echo "ENTER THE NUMBER"**

**read num**

**mun=`echo $num | rev`**

**echo $mun**

**#QUESTION BANK PROBLEM 23 :**

**PRIME NUMBER OR NOT**

**$vi qbprg 23**

**clear**

**echo "ENTER THE NUMBER"**

**read num**

**i=2**

**while test $i -lt $num**

**do**

**k=`expr $num / $i`**

**if test $k -eq 0**

**then**

**echo "THE NUMBER IS COMPOSITE"**

**exit**

**fi**

**i=`expr $i + 1`**

**done**

**echo "THE NUMBER IS PRIME"**

**# QUESTION BANK PROBLEM 24 :**

**GREATEST OF 3**

**$vi qbprg24**

**clear**

**if [ $# -ne 3 ]**

**then**

**echo "INVALID NUMBER OF ARGUMENTS"**

**elif [ $1 -gt $2 -a $1 -gt $3 ]**

**then**

**echo "$1 is GREATEST"**

**elif [ $2 -gt $1 -a $2 -gt $3 ]**

**then**

**echo "$2 is GREATEST"**

**else**

**echo "$3 is GREATEST"**

**fi**

**#QUESTION BANK 28 :**

**USER DRIVEN PROGRAM FOR FILE ALTERING**

**clear**

**echo "ENTER THE FILE NAME"**

**read file**

**choice=1**

**while [ choice!=7 ]**

**do**

**echo " 1. DISPLAY OWNER AND GROUP INFORMATION"**

**echo " 2. NUMBER OF LINES IN THE FILE"**

**echo " 3. NUMBER OF WORDS IN THE FILE"**

**echo " 4. OCCURANCE OF WORD IN FILE"**

**echo " 5. REPLACE WORD IN FILE"**

**echo " 6. DISPLAY FILE"**

**echo " 7. EXIT"**

**echo "ENTER THE CHOICE"**

**read choice**

**case $choice in**

**1)**

**stat -c "%U %G" $file ;;**

**2)**

**echo "NUMBER OF LINES IN THE FILE"**

**wc -l < $file**

**;;**

**3)**

**echo "NUMBER OF WORDS IN THE FILE"**

**wc -w < $file**

**;;**

**4)**

**echo "ENTER THE WORD"**

**read word**

**grep -o $word $file | wc -w**

**;;**

**5)**

**cat $file**

**echo "ENTER WORD TO REPLACE"**

**read word1**

**echo "ENTER WORD TO REPLACE WITH"**

**read word2**

**sed -i "s~$word1~$word2~g" $file**

**echo "WORD HAS BEEN REPLACED"**

**cat $file**

**;;**

**6)**

**cat $file;;**

**7)**

**exit;;**

**\*)**

**echo "INVALID INPUT";;**

**esac**

**done**

**OPERATING SYSTEM CODES : PART 2 : AWK CODES**

**#QUESTION BANK PROBLEM 6+7  
 RESULT OF STUDENTS**

**PROGRAM TO CREATE :**

**command.awk**

**BEGIN {**

**FS=",";**

**}**

**{**

**id[NR]=$1;**

**name[NR]=$2;**

**total[NR]= ($3+$4+$5+$6+$7)**

**if(($3 && $4 >= 30) && ($5 && $6 && $7 >=40))**

**result[NR]= "PASS"**

**else**

**result[NR]= "FAIL"**

**}**

**END {**

**print "SHOWING RESULTS FOR STUDENTS\n";**

**i=2;**

**while(i<=FNR)**

**{**

**printf("%d %-10s %.2f %s\n" ,id[i] ,name[i], total[i], result[i++]);**

**}**

**}**

**Docment to Create :**

**marks.txt**

**1,A,70,45,56,67,78**

**2,B,34,45,65,76,54**

**3,C,33,22,11,44,33**

**4,D,55,66,76,78,97**

**5,E,45,34,56,76,76**

**CODE TO RUN**

**awk -f command.awk marks.txt**

**#QUESTION BANK PROGRAM 25+26 :**

**PRINTING AVERAGE OF MARKS**

**PROGRAM TO RUN :**

**command.awk**

**BEGIN {**

**FS =","**

**}**

**{**

**name[NR] =$1;**

**id [NR] =$2;**

**avg[NR] =($3+$4+$5+$6+$7)/5**

**}**

**END {**

**printf("REPORT OF AVERAGE OF STDUENTS\n");**

**i=1;**

**while(i<=FNR)**

**{**

**printf("%-10s %.2f\n" name[i+1], id[i], avg[i++])**

**}**

**}**

**TEXT DOCUMENT**

**average.txt**

**A,1111,70,45,56,67,78**

**B,2222,34,45,65,76,54**

**C,3333,33,22,11,44,33**

**D,4444,55,66,76,78,97**

**E,5555,45,34,56,76,76**

**COMMAND TO RUN**

**awk -f command.aawk average.txt**

**#QUESTION BANK PROGRAM 16 :**

**BOOKS SALE**

**PROGRAM TO RUN**

**command.awk**

**BEGIN {**

**FS=",";**

**}**

**{**

**id[NR]=$1;**

**name[NR]=$2;**

**rate[NR]=$4;**

**sold[NR]= $3 \* $4;**

**finalbill = finalbill + sold[NR];**

**}**

**END {**

**printf("Sales Report for Books\n");**

**i=1;**

**while(i<=FNR)**

**{**

**printf("%d %-20s %d %d \n", id[i], name[i], rate[i], sold[i++]);**

**}**

**printf("\n");**

**printf("TOTAL SALE = %d\n", finalbill);**

**}**

**DOCUMENT TO RUN**

**books.txt**

**1,Linux Programming,2,450**

**2,Advanced Linux,3,300**

**3,Networking,4,400**

**4,OOAD&MUL,3,450**

**5,JAVA2,5,200**

**CODE TO RUN**

**awk -f command.awk books.txt**