# 2A. Compare Two Strings

echo enter two strings read a

read b

if [ -z $a ] then

echo The first string is empty fi

if [ -z $b ] then

echo The second string is empty fi

if [ $a = $b ] then

echo The strings are equal else

echo The strings are not equal fi

# 2B. Extract the First and Last Character from a string

echo enter the string read a first="${a:0:1}"

second="${a: -1}" echo $first

echo $second

# 2C. Palindrome

echo "Enter the number" read n

num=0 a=$n

while [ $n -gt 0 ] do

num=$(expr $num \\* 10) k=$(expr $n % 10) num=$(expr $num + $k) n=$(expr $n / 10)

done

if [ $num -eq $a ] then

echo "it is a palindrome" else

echo "it is not a palindrome" fi

# 3A. Factorial

echo enter the number of rows read r

i=1 t=1

while [ $i -le $r ] do

j=1

while [ $j -le $i ] do

echo -n "\*"

j=$((j+1)) done echo i=$((i+1)) done

# 3B. Sum of n numbers

echo How many numbers do you need to add read n

i=1 sum=0

echo Enter the numbers while [ $i -le $n ]

do

read num sum=$((sum + num)) i=$((i + 1))

done

echo Total is $sum

## 3C. Menu Driven Program to Perform Arithmetic Operation

echo Enter two numbers read a

read b

echo MENU

echo 1.Addition 2.Subraction 3.Multiplication 4.Division echo Select from the above menu

read c case $c in

1. echo sum = $(expr $a + $b);;
2. echo Difference = $(expr $a - $b);;
3. echo Product = $(expr $a \\* $b);;
4. echo Quotient = $(expr $a / $b);;
5. echo Invalid Choice esac

# 3D. Print \* Pattern

echo enter the number of rows read r

i=1 t=1

while [ $i -le $r ] do

j=1

while [ $j -le $i ] do

echo -n "\*"

j=$((j+1)) done echo i=$((i+1)) done

### 4A. Convert the characters of file from Lowercase to Uppercase

echo "Enter the filename" read name

if [ ! -f $name ] then

echo filename $name does not exist exit 1

fi

tr "[a-z]" "[A-Z]" < $name

### 4B. Count the Number of characters, words and lines in a given text file.

echo Enter the filename read file

c=`cat $file | wc -c` w=`cat $file | wc -w` l=`grep -c "." $file`

echo Number of characters in $file is $c echo Number of Words in $file is $w echo Number of lines in $file is $l

## 4C. Check if the given file exists, if not create a new file.

echo enter the file name read file

if [ -f $file ] then

echo File exists else

echo File does not exist touch $file

echo $file has been created fi

# 5. Display Various System Information

echo SYSTEM INFORMATION

echo Current Date is $(date) echo Current Directory = $(pwd)

echo Kernel All Information = $(uname -m) echo Disk space Usage:

df

echo List of blocked devices: lsblk

# 7A. First Come First Serve

#include<stdio.h> #include<conio.h> void main()

{

int nop,wt[10],twt,tat[10],ttat,i,j,bt[10],t; float awt,atat;

clrscr(); awt=0.0; atat=0.0;

printf("Enter the no.of process:"); scanf("%d",&nop); for(i=0;i<nop;i++)

{

printf("Enter the burst time for process %d: ", i); scanf("%d",&bt[i]);

} wt[0]=0;

tat[0]=bt[0]; twt=wt[0]; ttat=tat[0]; for(i=1;i<nop;i++)

{

wt[i]=wt[i-1]+bt[i-1];

tat[i]=wt[i]+bt[i]; twt+=wt[i]; ttat+=tat[i];

}

awt=(float)twt/nop; atat=(float)ttat/nop;

printf("\nProcessid\tBurstTime\tWaitingTime\tTurnaroundTime\n"); for(i=0;i<nop;i++) printf("%d\t\t%d\t\t%d\t\t%d\n",i,bt[i],wt[i],tat[i]);

printf("\nTotal Waiting Time:%d\n",twt); printf("\nTotal Around Time:%d\n",ttat); printf("\nAverage Waiting Time:%f\n",awt); printf("\nAverage Total Around Time:%f\n",atat); getch();

}

# 7B. Shortest Job First

#include<stdio.h> #include<conio.h> void main()

{

int nop,wt[10],twt,tat[10],ttat,i,j,bt[10],t; float awt,atat;

clrscr(); awt=0.0; atat=0.0;

printf("Enter the no.of process:"); scanf("%d",&nop); for(i=0;i<nop;i++)

{

printf("Enter the burst time for process %d: ", i); scanf("%d",&bt[i]);

}

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

{

for(j=i+1;j<nop;j++)

{

if(bt[i]>=bt[j])

{

t=bt[i]; bt[i]=bt[j]; bt[j]=t;

}

}

} wt[0]=0;

tat[0]=bt[0]; twt=wt[0]; ttat=tat[0]; for(i=1;i<nop;i++)

{

wt[i]=wt[i-1]+bt[i-1];

tat[i]=wt[i]+bt[i]; twt+=wt[i]; ttat+=tat[i];

}

awt=(float)twt/nop;

atat=(float)ttat/nop; printf("\nProcessid\tBurstTime\tWaitingTime\tTurnaroundTime\n"); for(i=0;i<nop;i++) printf("%d\t\t%d\t\t%d\t\t%d\n",i,bt[i],wt[i],tat[i]);

printf("\nTotal Waiting Time:%d\n",twt); printf("\nTotal Around Time:%d\n",ttat); printf("\nAverage Waiting Time:%f\n",awt); printf("\nAverage Total Around Time:%f\n",atat); getch();

}

# 7C. Priority Scheduling

#include<stdio.h> #include<conio.h> void main()

{

int nop,t,wt[10],twt,tat[10],ttat,i,j,p[10],b[10],tmp; float awt, atat;

clrscr(); awt=0.0; atat=0.0;0

printf("Enter the number of process:"); scanf("%d",&nop);

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

{

printf("Enter the burst time of Process %d:",i); scanf("%d",&b[i]);

}

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

printf("Enter the priority number of each Process %d:",i); scanf("%d",&p[i]);

}

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

{

for(j=i+1;j<nop;j++)

{

if(p[i]>p[j])

{

t=p[i]; p[i]=p[j]; p[j]=t;

tmp=b[i]; b[i]=b[j]; b[j]=tmp;

}

}

} wt[0]=0;

tat[0]=b[0]; twt=wt[0]; ttat=tat[0]; for(i=1;i<nop;i++)

{

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

tat[i]=wt[i]+b[i]; twt+=wt[i]; ttat+=tat[i];

}

awt=(float)twt/nop; atat=(float)ttat/nop;

printf("Process No:\tPriority:\tBurst Time:\tWaiting Time\tTurnaround Time:\n"); for(i=0;i<nop;i++)

printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",i,p[i],b[i],wt[i],tat[i]);

printf("Total TurnAround Time:%d\n",ttat); printf("Total Waiting Time:%d\n",twt); printf("Average Waiting Time:%f\n",awt); printf("Average Turnaround Time:%f\n",atat); getch();

}

# Reader - Writer Problem

#include<stdio.h> #include<conio.h> #include<process.h> void main()

{

typedef int semaphore; semaphore sread=0, swrite=0; int ch,r=0;

clrscr();

printf("\nReader writer"); do

{

printf("\nMenu");

printf("\n\t 1.Read from file"); printf("\n \t2.Write to file"); printf("\n \t 3.Exit the reader"); printf("\n \t 4.Exit the writer"); printf("\n \t 5.Exit"); printf("\nEnter your choice:"); scanf("%d",&ch);

switch(ch)

{

case 1: if(swrite==0)

{

sread=1; r+=1;

printf("\nReader %d reads",r);

}

else

{printf("\n Not possible");

}

break;

case 2: if(sread==0 && swrite==0)

{

swrite=1;

printf("\nWriter in Progress");

}

else if(swrite==1)

{printf("\nWriter writes the files");

}

else if(sread==1)

{printf("\nCannot write while reader reads the file");

}

else

printf("\nCannot write file"); break;

case 3: if(r!=0)

{

printf("\n The reader %d closes the file",r); r-=1;

}

else if(r==0)

{

printf("\n Currently no readers access the file"); sread=0;

}

else if(r==1)

{

printf("\nOnly 1 reader file");

}

else

printf("%d reader are reading the file\n",r);

break;

case 4: if (swrite==1)

{

printf("\nWriter close the file"); swrite=0;

}

else

printf("\nThere is no writer in the file"); break;

case 5: exit(0);

}

}

while(ch<6); getch();

}

# Dining Philosophers Problem

#include<stdio.h> #include<conio.h> #define LEFT (i+4) %5 #define RIGHT (i+1) %5 #define THINKING 0

#define HUNGRY 1

#define EATING 2 int state[5];

void put\_forks(int); void test(int);

void take\_forks(int); void philosopher(int i)

{

if(state[i]==0)

{

take\_forks(i); if(state[i]==EATING)

printf("\n Eating in process. ");

put\_forks(i);

}

}

void put\_forks(int i)

{

state[i]=THINKING;

printf("\n philosopher %d completed its works",i); test(LEFT);

test(RIGHT);

}

void take\_forks(int i)

{

state[i]=HUNGRY; test(i);

}

void test(int i)

{

if(state[i]==HUNGRY && state[LEFT]!=EATING && state[RIGHT]!=EATING)

{

printf("\n philosopher %d can eat",i); state[i]=EATING;

}

}

void main()

{

int i; clrscr();

for(i=1;i<=5;i++) state[i]=0;

printf("\n\t\t\t Dining Philosopher Problem"); printf("\n\t\t. ");

for(i=1;i<=5;i++)

{

printf("\n\n the philosopher %d falls hungry\n",i); philosopher(i);

}

getch();

}

# First fit, Best Fit, Worst fit

#include<stdio.h> #include<conio.h> void main()

{

int f3[20],f2[20],r[20],r1[20],ms,bod,sb[20],nsb[20],nsb1[20],np,sp[20]; int f[20],i,j,l,k,z[20],s=0;

clrscr();

printf("enter the memory size:"); scanf("%d",&ms);

printf("\n enter the number of block of division of memory:"); scanf("%d",&bod);

printf("enter the size of each block:"); for(i=1;i<=bod;i++)

{

printf("\nBlock[%d]:",i);

scanf("%d",&sb[i]); f[i]=1;

f2[i]=1;

f3[i]=1;

r[i]=1;

r1[i]=1;

z[i]=sb[i];

}

printf("\nenter the number of process:"); scanf("%d",&np);

printf("\nenter the size of each process:"); for(i=1;i<=np;i++)

{

printf("\nprocess[%d]:",i);

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

}

printf("\n FIRST FIT "); printf("\n \*\*\*\*\*\*\*\*\* "); for(i=1;i<=np;i++)

{

for(j=1;j<=bod;j++)

{

if((sb[j]>=sb[i]) && (f[j]!=0))

{

printf("\n Process p[%d] is allocated to Block[%d]",i,j); f[j]=0;

z[j]=sb[j]-sp[i]; s++;

goto l1;

}

}

printf("\n process p[%d] cannot be allocated",i); l1:

printf(" ");

}

printf("\n\n Remaining space left in each block \n"); printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n"); for(i=1;i<=bod;i++)

{

printf("\n Block[%d]: free space =%d",i,z[i]);

}

printf("\n\nUnallocated Blocks"); printf(" \n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); for(i=1;i<=bod;i++)

{

if(f[i]!=0)

{

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

}

}

if(s==bod)

printf("\n No Block is left unallocated"); getch();

clrscr(); s=0;

getch();

printf("\n\n BEST FIT "); printf("\n \*\*\*\*\*\*\*\* "); for(i=2;i<=bod;i++)

{

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

{

if(sb[i]>=sb[j]) r[i]++;

else r[j]++;

}

}

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

{

nsb[r[i]]=sb[i];

z[r[i]]=sb[i];

}

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

{

for(j=1;j<=bod;j++)

{

if((nsb[j]>=sp[i]) && (f2[j]!=0))

{

for(k=1;k<=bod;k++)

{

if(r[k]==j) l=k;

}

printf("\nProcess p[%d] is allocated to Block[%d]",i,l); f2[j]=0;

z[j]=nsb[j]-sp[i]; s++;

goto l2;

}

}

printf("\n process p[%d] cannot be allocated",i); l2:

printf(" ");

}

printf("\n free space in each block \n"); printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n"); for(i=1;i<=bod;i++)

printf("\nBlock [%d]: free space =%d",i,z[r[i]]);

printf("\n\nUnallocated Blocks"); printf(" \n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); for(i=1;i<=bod;i++)

{

if(f2[r[i]]!=0)

{

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

}

}

if(s==bod)

printf("\n No Block is left unallocated"); getch();

clrscr(); s=0;

getch();

printf("\n\n WORST FIT "); printf("\n \*\*\*\*\*\*\*\*\* "); for(i=2;i<=bod;i++)

{

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

{

if(sb[i]<=sb[j]) r1[i]++;

else r1[j]++;

}

}

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

{

nsb1[r1[i]]=sb[i];

z[r1[i]]=sb[i];

}

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

{

for(j=1;j<=bod;j++)

{

if((nsb1[j]>=sp[i]) && (f3[j]!=0))

{

for(k=1;k<=bod;k++)

{

if(r1[k]==j) l=k;

}

printf("\nProcess p[%d] is allocated to Block[%d]",i,l); f3[j]=0;

z[j]=nsb1[j]-sp[i]; s++;

goto l3;

}

}

printf("\n process p[%d] cannot be allocated",i); l3:

printf(" ");

}

printf("\n free space in each block \n"); printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n"); for(i=1;i<=bod;i++)

printf("\nBlock [%d]: free space =%d",i,z[r1[i]]);

printf("\n\nUnallocated Blocks"); printf(" \n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); for(i=1;i<=bod;i++)

{

if(f3[r1[i]]!=0)

{

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

}

if(s==bod)

printf("\n No Block is left unallocated"); getch();

printf("\n");

}

}

# Bankers Algorithm

#include<stdio.h> #include<conio.h>

int np,nr,r[10],safe[10],ava[10],aval[10],re[10],f[10],i,j,flag,z,index,pid; int m[10][10],need[10][10],all[10][10];

void resourse()

{

printf("\nEnter the no. of resourses: "); scanf("%d",&nr);

printf("\nEnter the resources instances \n"); for(i=0;i<nr;i++)

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

}

void alloc()

{

printf("\nEnter the no of process: "); scanf("%d",&np);

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

{ f[i]=0;

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

{

printf("\n Resourse %d for %d ",j+1,i+1); scanf("%d",&all[i][j]);

}}}

void maxreq()

{

printf("\nEnter the maximum request for each process \n"); for(i=0;i<np;i++)

for(j=0;j<nr;j++) scanf("%d",&m[i][j]); printf("\nThe Available Matrix\n"); printf(" \n");

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

{ z=0;

for(j=0;j<np;j++) z+=all[j][i];

ava[i]=r[i]-z;

printf("%d\t",ava[i]); aval[i]=ava[i];

}}

void needcal()

{

printf("\n");

printf("\nThe Need Matrix \n"); printf(" \n");

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

{

printf("\n"); for(j=0;j<nr;j++)

{

need[i][j]=m[i][j]-all[i][j];

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

}}

printf("\n\n");

}

void request()

{

flag=0;index=0;

printf("\nEnter the requesting process id:"); scanf("%d",&pid);

printf("\nEnter the resourseintance required \n"); for(i=0;i<nr;i++)

{

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

if(re[i]>m[pid][i]); flag=1;

}

if(flag==0)

{

for(i=0;i<nr;i++) need[pid][i]=re[i]; for(i=0;i<np;i++)

{

printf("\n"); for(j=0;j<nr;j++) printf("%d \t",need[i][j]);

}}

else

{

printf("\n request exceeds maximum request"); exit(0);

}.

}

void out()

{

printf("The safe sequensce is\n"); for(i=0;i<np;i++) printf("p[%d]\t",safe[i]); printf("\n\n");

}

void safety()

{

flag=0;i=0;j=0;z=0;index=0; while(1)

{

if(z++ > 2\*np)

{

printf("\n no safe sequence"); exit(0);

}

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

{

if(need[i][j] <=ava[j]&&f[i] !=1)

{

flag=0;

}

else

{

flag=1; break;

}

}

if(flag==0)

{ f[i]=1;

safe[index]=i; for(j=0;j<nr;j++) ava[j]+= all[i][j]; index++; if(index >=np) return;

} i++;

if(i>=np) i=0;

}

}

void main()

{

clrscr(); resourse(); alloc();

maxreq(); needcal(); safety();

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

{f[i]=0;safe[i]=0;} request(); for(j=0;j<nr;j++) ava[j]=aval[j]; safety();

out();

}

# Producer Consumer Problem

#include<stdio.h> #include<conio.h> int main()

{

int s,n,b=0,p=0,c=0; clrscr();

printf("\n producer and consumer problem"); do

{

printf("\n menu");

printf("\n 1.producer an item"); printf("\n 2.consumer an item"); printf("\n 3.add item to the buffer"); printf("\n 4.display status"); printf("\n 5.exit");

printf("\n enter the choice"); scanf("%d",&s);

switch(s)

{

case 1:

p=p+1;

printf("\n item to be produced"); break;

case 2:

if(b!=0)

{

c=c+1; b=b-1;

printf("\n item to be consumed");

}

else

{

printf("\n the buffer is empty please wait...");

}

break; case 3:

if(b<n)

{

if(p!=0)

{

b=b+1;

printf("\n item added to buffer");

}

else

printf("\n no.of items to add...");

}

else

printf("\n buffer is full,please wait"); break;

case 4:

printf("no.of items produced :%d",p); printf("\n no.of consumed items:%d",c); printf("\n no.of buffered item:%d",b); break;

case 5:exit(0);

}

}

while(s<=5); getch(); return 0;

}