1st

#include<stdio.h>

#include<unistd.h>

int main()

{

printf("Process ID: %d\n", getpid() );

printf("Parent Process ID: %d\n", getpid() );

return 0;

}

9th question

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include<unistd.h>

#include <pthread.h>

#define N 5

#define THINKING 2

#define HUNGRY 1

#define EATING 0

#define LEFT (phnum + 4) % N

#define RIGHT (phnum + 1) % N

int state[N];

int phil[N] = { 0, 1, 2, 3, 4 };

sem\_t mutex;

sem\_t S[N];

void test(int phnum)

{

if (state[phnum] == HUNGRY

&& state[LEFT] != EATING

&& state[RIGHT] != EATING) {

state[phnum] = EATING;

sleep(2);

printf("Philosopher %d takes fork %d and %d\n",

phnum + 1, LEFT + 1, phnum + 1);

printf("Philosopher %d is Eating\n", phnum + 1);

sem\_post(&S[phnum]);

}

}

void take\_fork(int phnum)

{

sem\_wait(&mutex);

state[phnum] = HUNGRY;

printf("Philosopher %d is Hungry\n", phnum + 1);

test(phnum);

sem\_post(&mutex);

sem\_wait(&S[phnum]);

sleep(1);

}

void put\_fork(int phnum)

{

sem\_wait(&mutex);

state[phnum] = THINKING;

printf("Philosopher %d putting fork %d and %d down\n",

phnum + 1, LEFT + 1, phnum + 1);

printf("Philosopher %d is thinking\n", phnum + 1);

test(LEFT);

test(RIGHT);

sem\_post(&mutex);

}

void\* philosopher(void\* num)

{

while (1) {

int\* i;

\*i=num;

sleep(1);

take\_fork(\*i);

sleep(0);

put\_fork(\*i);

}

}

int main()

{

int i;

pthread\_t thread\_id[N];

sem\_init(&mutex, 0, 1);

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

sem\_init(&S[i], 0, 0);

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

pthread\_create(&thread\_id[i], NULL,

philosopher, &phil[i]);

printf("Philosopher %d is thinking\n", i + 1);

}

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

pthread\_join(thread\_id[i], NULL);

}

10th question

#include<stdio.h>

int main()

{

int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;

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

{

flags[i] = 0;

allocation[i] = -1;

}

printf("Enter no. of blocks: ");

scanf("%d", &bno);

printf("\nEnter size of each block: ");

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

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

printf("\nEnter no. of processes: ");

scanf("%d", &pno);

printf("\nEnter size of each process: ");

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

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

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

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

if(flags[j] == 0 && bsize[j] >= psize[i])

{

allocation[j] = i;

flags[j] = 1;

break;

}

printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");

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

{

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

if(flags[i] == 1)

printf("%d\t\t\t%d",allocation[i]+1,psize[allocation[i]]);

else

printf("Not allocated");

}

}

12th question

#include<stdio.h>

#include<conio.h>

struct st

{

char dname[10];

char sdname[10][10];

char fname[10][10][10];

int ds,sds[10];

}dir[10];

int main()

{

int i,j,k,n;

printf("enter number of directories:");

scanf("%d",&n);

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

{

printf("enter directory %d names:",i+1);

scanf("%s",&dir[i].dname);

printf("enter size of directories:");

scanf("%d",&dir[i].ds);

for(j=0;j<dir[i].ds;j++)

{

printf("enter subdirectory name and size:");

scanf("%s",&dir[i].sdname[j]);

scanf("%d",&dir[i].sds[j]);

for(k=0;k<dir[i].sds[j];k++)

{

printf("enter file name:");

scanf("%s",&dir[i].fname[j][k]);

}

}

}

printf("\ndirname\t\tsize\tsubdirname\tsize\tfiles");

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

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

{

printf("%s\t\t%d",dir[i].dname,dir[i].ds);

for(j=0;j<dir[i].ds;j++)

{

printf("\t%s\t\t%d\t",dir[i].sdname[j],dir[i].sds[j]);

for(k=0;k<dir[i].sds[j];k++)

printf("%s\t",dir[i].fname[j][k]);

printf("\n\t\t");

}

printf("\n");

}

getch();

13th question

#include<stdio.h>

int main()

{

char a;

FILE \*fp;

printf("File name :");

scanf("%s",&a);

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

if(!fp)

{

printf("Error: File cannot be opened\n") ;

return 0;

}

printf("Position pointer in the beginning : %ld\n",ftell(fp));

char ch;

while(fread(&ch,sizeof(ch),1,fp)==1)

{

printf("%c",ch);

}

printf("\nSize of file in bytes is : %ld\n",ftell(fp));

fclose(fp);

return 0;

}

14th question

#include<stdio.h>

#include<conio.h>

int main()

{

int clm[7][5],req[7][5],alloc[7][5],rsrc[5],avail[5],comp[7];

int first,p,r,i,j,prc,count,t;

count=0;

for(i=1;i<=7;i++)

comp[i]=0;

printf("\t BANKERS ALGORITHM IN C \n\n");

printf("Enter the no of processes : ");

scanf("%d",&p);

printf("\n\nEnter the no of resources : ");

scanf("%d",&r);

printf("\n\nEnter the claim for each process : ");

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

{

printf("\nFor process %d : ",i);

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

{

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

}

}

printf("\n\nEnter the allocation for each process : ");

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

{

printf("\nFor process %d : ",i);

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

{

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

}

}

printf("\n\nEnter total no of each resource : ");

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

scanf("%d",&rsrc[j]);

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

{

int total=0;

avail[j]=0;

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

{total+=alloc[i][j];}

avail[j]=rsrc[j]-total;

}

do

{

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

{

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

{

req[i][j]=clm[i][j]-alloc[i][j];

}

}

printf("\n\nAvailable resources are : ");

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

{ printf(" ",avail[j]); }

printf("\nClaim matrix:\tAllocation matrix:\n");

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

{

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

{

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

}

printf("\t\t\t");

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

{

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

}

printf("\n");

}

prc=0;

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

{

if(comp[i]==0)//if not completed

{

prc=i;

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

{

if(avail[j]=0)

{

prc=0;

break;

}

}

}

if(prc!=0)

break;

}

if(prc!=0)

{

printf("\nProcess ",prc,"runs to completion!");

count++;

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

{

avail[j]+=alloc[prc][j];

alloc[prc][j]=0;

clm[prc][j]=0;

comp[prc]=1;

}

}

}

while(count!=p&&prc!=0);

if(count==p)

printf("\nThe system is in a safe state!!");

else

printf("\nThe system is in an unsafe state!!");

getch();

}

15th question

#include<stdio.h>

#include<stdlib.h>

int mutex=1,full=0,empty=3,x=0;

int main()

{

int n;

void producer();

void consumer();

int wait(int);

int signal(int);

printf("\n1.Producer\n2.Consumer\n3.Exit");

while(1)

{

printf("\nEnter your choice:");

scanf("%d",&n);

switch(n)

{

case 1: if((mutex==1)&&(empty!=0))

producer();

else

printf("Buffer is full!!");

break;

case 2: if((mutex==1)&&(full!=0))

consumer();

else

printf("Buffer is empty!!");

break;

case 3:

exit(0);

break;

}

}

return 0;

}

int wait(int s)

{

return (--s);

}

int signal(int s)

{

return(++s);

}

void producer()

{

mutex=wait(mutex);

full=signal(full);

empty=wait(empty);

x++;

printf("\nProducer produces the item %d",x);

mutex=signal(mutex);

}

void consumer()

{

mutex=wait(mutex);

full=wait(full);

empty=signal(empty);

printf("\nConsumer consumes item %d",x);

x--;

mutex=signal(mutex);

}

16th question

#include <stdio.h>

int main()

{

int referenceString[10], pageFaults = 0, m, n, s, pages, frames;

printf("\nEnter the number of Pages:\t");

scanf("%d", &pages);

printf("\nEnter reference string values:\n");

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

{

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

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

}

printf("\n What are the total number of frames:\t");

{

scanf("%d", &frames);

}

int temp[frames];

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

{

temp[m] = NULL;

}

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

{

s = 0;

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

{

if(referenceString[m] == temp[n])

{

s++;

pageFaults--;

}

}

pageFaults++;

if((pageFaults <= frames) && (s == 0))

{

temp[m] = referenceString[m];

}

else if(s == 0)

{

temp[(pageFaults - 1) % frames] = referenceString[m];

}

printf("\n");

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

{

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

}

}

printf("\nTotal Page Faults:\t%d\n", pageFaults);

return 0;

}

17th question

#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("Enter number of frames: ");

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

printf("Enter number of pages: ");

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

printf("Enter 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;

}

18th question

#include<stdio.h>

int main()

{

int no\_of\_frames, no\_of\_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j, k,

pos, max, faults = 0;

printf("Enter number of frames: ");scanf("%d", &no\_of\_frames);

printf("Enter number of pages: ");

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

printf("Enter page 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]){

flag1 = flag2 = 1;

break;

}

}

if(flag1 == 0){

for(j = 0; j < no\_of\_frames; ++j){

if(frames[j] == -1){

faults++;

frames[j] = pages[i];

flag2 = 1;break;

}

}

}

if(flag2 == 0){

flag3 =0;

for(j = 0; j < no\_of\_frames; ++j){

temp[j] = -1;

for(k = i + 1; k < no\_of\_pages; ++k){

if(frames[j] == pages[k]){

temp[j] = k;

break;

}

}

}

for(j = 0; j < no\_of\_frames; ++j){

if(temp[j] == -1){

pos = j;

flag3 = 1;

break;

}

}

if(flag3 ==0){

max = temp[0];pos = 0;

for(j = 1; j < no\_of\_frames; ++j){

if(temp[j] > max){

max = temp[j];

pos = j;

}

}

}

frames[pos] = pages[i];

faults++;

}

printf("\n");

for(j = 0; j < no\_of\_frames; ++j){

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

}

}

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

return 0;

}

19th question

#include<stdio.h>

int main()

{

char a;

FILE \*fp;

printf("File name :");

scanf("%s",&a);

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

if(!fp)

{

printf("Error: File cannot be opened\n") ;

return 0;

}

printf("Position pointer in the beginning : %ld\n",ftell(fp));

char ch;

while(fread(&ch,sizeof(ch),1,fp)==1)

{

printf("%c",ch);

}

printf("\nSize of file in bytes is : %ld\n",ftell(fp));

fclose(fp);

return 0;

}

20th question

#include<stdio.h>

#include<stdlib.h>

void main()

{

int f[50], index[50],i, n, st, len, j, c, k, ind,count=0;

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

f[i]=0;

x:printf("Enter the index block: ");

scanf("%d",&ind);

if(f[ind]!=1)

{

printf("Enter no of blocks needed and no of files for the index %d on the disk : \n", ind);

scanf("%d",&n);

}

else

{

printf("%d index is already allocated \n",ind);

goto x;

}

y: count=0;

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

{

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

if(f[index[i]]==0)

count++;

}

if(count==n)

{

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

f[index[j]]=1;

printf("Allocated\n");

printf("File Indexed\n");

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

printf("%d-------->%d : %d\n",ind,index[k],f[index[k]]);

}

else

{

printf("File in the index is already allocated \n");

printf("Enter another file indexed");

goto y;

}

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

scanf("%d", &c);

if(c==1)

goto x;

else

exit(0);

}

21st question,

#include<stdio.h>

#include<conio.h>

void main()

{

int f[50], i, st, len, j, c, k, count = 0;

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

f[i]=0;

printf("Files Allocated are : \n");

x: count=0;

printf("Enter starting block and length of files: ");

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

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

if(f[k]==0)

count++;

if(len==count)

{

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

if(f[j]==0)

{

f[j]=1;

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

}

if(j!=(st+len-1))

printf("The file is allocated to disk\n");

}

else

printf("The file is not allocated \n");

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

scanf("%d", &c);

if(c==1)

goto x;

else

getch();

}

22nd question

#include<stdio.h>

#include<stdlib.h>

int main()

{

int RQ[100],i,n,TotalHeadMoment=0,initial;

printf("Enter the number of Requests\n");

scanf("%d",&n);

printf("Enter the Requests sequence\n");

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

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

printf("Enter initial head position\n");

scanf("%d",&initial);

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

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

initial=RQ[i];

}

printf("Total head moment is %d",TotalHeadMoment);

return 0;

}

23rd question

#include <stdio.h>

#include <math.h>

int main()

{

int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20],queue2[20], temp1 = 0, temp2 = 0;

float avg;

printf("Enter the max range of disk\n");

scanf("%d", &max);

printf("Enter the initial head position\n");

scanf("%d", &head);

printf("Enter the size of queue request\n");

scanf("%d", &n);

printf("Enter the queue of disk positions to be read\n");

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

{

scanf("%d", &temp);

if (temp >= head)

{

queue1[temp1] = temp;

temp1++;

}

else

{

queue2[temp2] = temp;

temp2++;

}

}

for (i = 0; i < temp1 - 1; i++)

{

for (j = i + 1; j < temp1; j++)

{

if (queue1[i] > queue1[j])

{

temp = queue1[i];

queue1[i] = queue1[j];

queue1[j] = temp;

}

}

}

for (i = 0; i < temp2 - 1; i++)

{

for (j = i + 1; j < temp2; j++)

{

if (queue2[i] < queue2[j])

{

temp = queue2[i];

queue2[i] = queue2[j];

queue2[j] = temp;

}

}

}

for (i = 1, j = 0; j < temp1; i++, j++)

queue[i] = queue1[j];

queue[i] = max;

for (i = temp1 + 2, j = 0; j < temp2; i++, j++)

queue[i] = queue2[j];

queue[i] = 0;

queue[0] = head;

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

{

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

seek += diff;

printf("Disk head moves from %d to %d with seek %d\n", queue[j],

queue[j + 1], diff);

}

printf("Total seek time is %d\n", seek);

avg = seek / (float)n;

printf("Average seek time is %f\n", avg);

return 0;

}

24th question

#include<stdio.h>

#include<stdlib.h>

int main()

{

int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;

printf("Enter the number of Requests\n");

scanf("%d",&n);

printf("Enter the Requests sequence\n");

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

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

printf("Enter initial head position\n");

scanf("%d",&initial);

printf("Enter total disk size\n");

scanf("%d",&size);

printf("Enter the head movement direction for high 1 and for low 0\n");

scanf("%d",&move);

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

{

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

{

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

{

int temp;

temp=RQ[j];

RQ[j]=RQ[j+1];

RQ[j+1]=temp;

}

}

}

int index;

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

{

if(initial<RQ[i])

{

index=i;

break;

}

}

if(move==1)

{

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

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

initial=RQ[i];

}

TotalHeadMoment=TotalHeadMoment+abs(size-RQ[i-1]-1);

TotalHeadMoment=TotalHeadMoment+abs(size-1-0);

initial=0;

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

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

initial=RQ[i];

}

}

else

{

for(i=index-1;i>=0;i--)

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

initial=RQ[i];

}

TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0);

TotalHeadMoment=TotalHeadMoment+abs(size-1-0);

initial =size-1;

for(i=n-1;i>=index;i--)

{

TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);

initial=RQ[i];

}

}

printf("Total head movement is %d",TotalHeadMoment);

return 0;

}

25th question

#include<stdio.h>

int main()

{

FILE \*fp;

fp = fopen("D:\\DEVC++\\file1.txt","r");

if(!fp)

{

printf("Error in opening file\n");

return 0;

}

printf("Position of the pointer : %ld\n",ftell(fp));

char ch;

while(fread(&ch,sizeof(ch),1,fp)==1)

{

printf("%c",ch);

}

printf("\nPosition of the pointer : %ld\n",ftell(fp));

rewind(fp);

printf("\n USING REWIND Position of the pointer : %ld\n",ftell(fp));

printf("\nUSING FSEEK.....");

fseek(fp, 6, 0);

while(fread(&ch,sizeof(ch),1,fp)==1)

{

printf("%c",ch);

}

fclose(fp);

return 0;

}