EXPERIMENTS FROM 11 to 20

11. Construct a C program to organize the file using single level directory.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
int nf=0,i=0,j=0,ch;
char mdname[10],fname[10][10],name[10];
printf("Enter the directory name:");
scanf("%s",mdname);
printf("Enter the number of files:");
scanf("%d",&nf);
do
{
printf("Enter file name to be created:");
scanf("%s",name);
for(i=0;i<nf;i++)
{
if(!strcmp(name,fname[i]))
break;
}
if(i==nf)
{
strcpy(fname[j++],name);
nf++;
}
else
printf("There is already %s\n",name);
printf("Do you want to enter another file(yes - 1 or no - 0):");
scanf("%d",&ch);
}
while(ch==1);
printf("Directory name is:%s\n",mdname);
printf("Files names are:");
```

```
for(i=0;i<j;i++)
printf("\n%s",fname[i]);
getch();
}
```

INPUT AND OUTPUT:

```
C:\Users\dinak\OneDrive\Des
Enter the directory name:operating
Enter the number of files:3
Enter file name to be created:2
Do you want to enter another file(yes - 1 or no - 0):no
Directory name is:operating
Files names are:
2
--------------------------------
Process exited after 27.38 seconds with return value 0
Press any key to continue . . .
```

12. Design a C program to organize the file using two level directory structure

Program:

```
#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();
```

INPUT AND OUTPUT:

```
C:\Users\dinak\OneDrive\Des

enter number of directories:2
enter directory 1 names:saran
enter size of directories:1
enter subdirectory name and size:polan 1
enter file name:dhruv
enter directory 2 names:anshul
enter size of directories:1
enter subdirectory name and size:dhruv 1
enter file name:ss

dirname         size    subdirname      size    files
*****************************************************
saran           1       polan           1       dhruv

anshul          1       dhruv           1       ss
```

13. Develop a C program for implementing random access files for processing the employee details.

Program:

```
#include <stdio.h>
struct clientData
{
unsigned int acctNum;
char lastName[ 15 ];
char firstName[ 10 ];
double balance;
};
int main( void )
{
unsigned int i;
struct client Data blank Client = { 0, "", "", 0.0 };
FILE *cfPtr;
if ( ( cfPtr = fopen( "credit.dat", "wb" ) ) == NULL )
{
puts( "File could not be opened." );
}
else
{
for ( i = 1; i <= 100; ++i )
{
fwrite( &blankClient, sizeof( struct clientData ), 1, cfPtr );
}
fclose ( cfPtr );
} }
```

INPUT AND OUTPUT:

```
Resource request for Process 1 granted.
Resource request for Process 3 denied (would lead to deadlock).

--------------------------------
Process exited after 0.02142 seconds with return value 0
Press any key to continue . . .
```

14. Illustrate the deadlock avoidance concept by simulating Bankers algorithm with C.

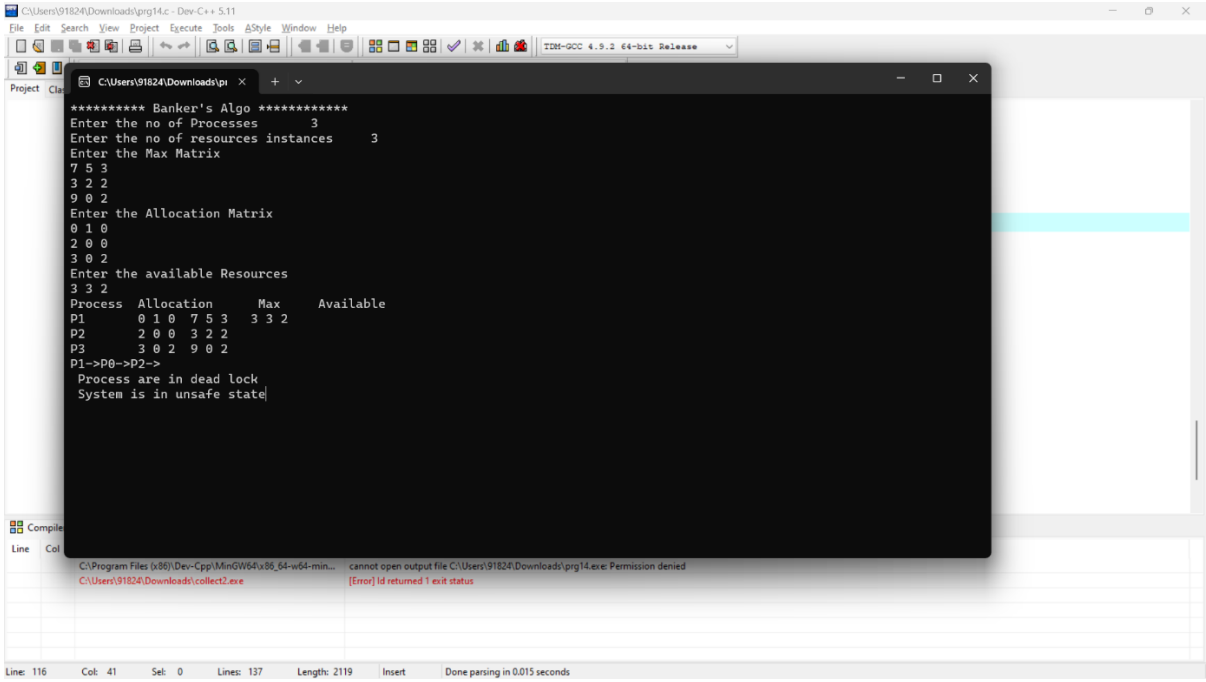
```
#include<stdio.h>

#include<conio.h>

int max[100][100];

int alloc[100][100];

int need[100][100];

int avail[100];

int n,r;

void input();

void show();

void cal();

int main()

{

 int i,j;

printf("********** Banker's Algo ************\n");

        input();

          show();

          cal();


          getch();

          return 0;

}
```

```
void input()
{
        int i,j;
          printf("Enter the no of Processes\t");
        scanf("%d",&n);
           printf("Enter the no of resources instances\t");
        scanf("%d",&r);
           printf("Enter the Max Matrix\n");
        for(i=0;i<n;i++)
                for(j=0;j<r;j++)
                        scanf("%d",&max[i][j]);
             printf("Enter the Allocation Matrix\n");
        for(i=0;i<n;i++)
                for(j=0;j<r;j++)
                        scanf("%d",&alloc[i][j]);
         printf("Enter the available Resources\n");
        for(j=0;j<r;j++)
                scanf("%d",&avail[j]);
}
 void show()
{
        int i,j;
        printf("Process\t Allocation\t Max\t Available\t");
          for(i=0;i<n;i++)
        {
           printf("\nP%d\t ",i+1);
                for(j=0;j<r;j++)
               printf("%d ",alloc[i][j]);
                 printf("\t");
                  for(j=0;j<r;j++)
```

```
                printf("%d ",max[i][j]);
                    printf("\t");
                            if(i==0)
                             for(j=0;j<r;j++)
                                        printf("%d ",avail[j]);
                        }
            }
void cal()
{
        int finish[100],temp,need[100][100],flag=1,k,c1=0;
          int safe[100];
               int i,j;
             for(i=0;i<n;i++)
                    finish[i]=0;
                  for(i=0;i<n;i++)
                    for(j=0;j<r;j++)
                              need[i][j]=max[i][j]-alloc[i][j];
            printf("\n");
          while(flag)
          {
                    flag=0;
                    for(i=0;i<n;i++)
                    {
                              int c=0;
                              for(j=0;j<r;j++)
                              {
                                        if((finish[i]==0)&&(need[i][j]<=avail[j]))
                                        {
                                                  c++;
                                                  if(c==r)
                                                  {
```

```
                                        for(k=0;k<r;k++)
                                        {
                                                avail[k]+=alloc[i][j];
                                                finish[i]=1;
                                                flag=1;
                                        }

                                        printf("P%d->",i);

                                        if(finish[i]==1)
                                                i=n;
                                }
                        }
                }
        }
        }

        for(i=0;i<n;i++)
                if(finish[i]==1)
                        c1++;
                else
                        printf("P%d->",i);

        if(c1==n)
                printf("\n The system is in safe state");
        else
        {
                printf("\n Process are in dead lock");
                  printf("\n System is in unsafe state");
        }
}
```

INPUT AND OUTPUT:



15. Construct a C program to simulate producer-consumer problem using semaphores.

Program:

```
#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);
}
```

INPUT AND OUTPUT:

```
C:\Users\Admin\OneDrive\Documents\semophores.exe

1.Producer
2.Consumer
3.Exit
Enter your choice:1

Producer produces the item 1
Enter your choice:2

Consumer consumes item 1
Enter your choice:1

Producer produces the item 1
Enter your choice:2

Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:1

Producer produces the item 1
Enter your choice:1

Producer produces the item 2
Enter your choice:2

Consumer consumes item 2
Enter your choice:2

Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:1

Producer produces the item 1
Enter your choice:1

Producer produces the item 2
Enter your choice:1

Producer produces the item 3
Enter your choice:1
Buffer is full!!
Enter your choice:3

--------------------------------
```

16. Construct a C program to simulate the First in First Out paging technique of memory management

Program:

```
#include <stdio.h>

int main()

{

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

  int pageFaults = 0;

  int frames = 3;

   int m, n, s, pages;

    pages = sizeof(incomingStream)/sizeof(incomingStream[0]);

   printf("Incoming \t Frame 1 \t Frame 2 \t Frame 3");

   int temp[frames];

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

   {

      temp[m] = -1;

   }

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

   {

      s = 0;

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

      {

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

         {

            s++;

            pageFaults--;

         }

      }

      pageFaults++;

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

      {

         temp[m] = incomingStream[m];
```

```
        }
        else if(s == 0)
        {
            temp[(pageFaults - 1) % frames] = incomingStream[m];
        }
        printf("\n");
        printf("%d\t\t\t",incomingStream[m]);
        for(n = 0; n < frames; n++)
        {
            if(temp[n] != -1)
                printf(" %d\t\t\t", temp[n]);
            else
                printf(" - \t\t\t");
        }
    }
    printf("\nTotal Page Faults:\t%d\n", pageFaults);
    return 0;
}
```

INPUT AND OUTPUT:

```
Enter number of frames: 5
Enter number of pages: 4
Enter page reference string: 3
9
5
6

3        -1       -1       -1       -1
3        9        -1       -1       -1
3        9        5        -1       -1
3        9        5        6        -1

Total Page Faults = 4
```