Aim: Write C program to simulate the Single level directory organization technique.

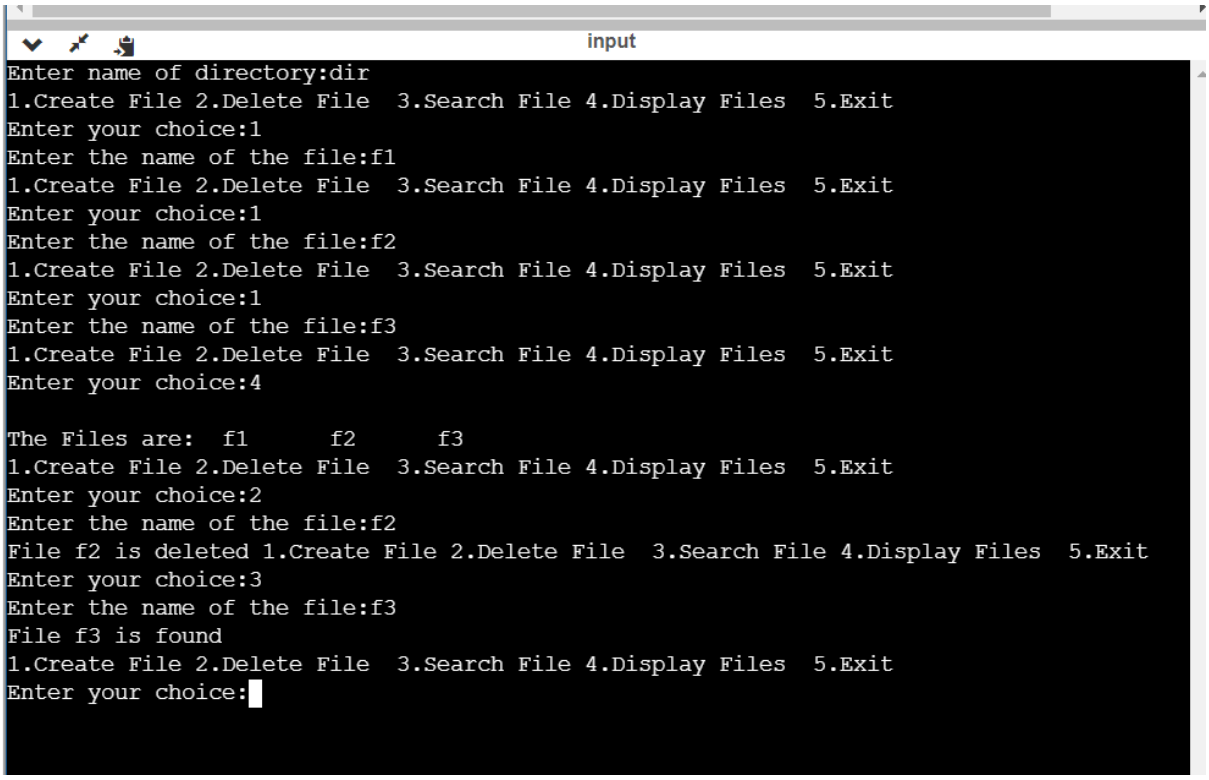
Program:

```
#include<stdlib.h>
#include<string.h>
#include<stdio.h>
struct{
char dname[10],fname[10][10];
int fcnt;
}dir;
void main(){
   int i,ch;
   char f[30];
   dir.fcnt = 0;
   printf("Enter name of directory:");
   scanf("%s", dir.dname);
   while(1)
   {
      printf("1. Create File\t2. Delete File\t3. Search File \n 4. Display Files\t5. Exit\nEnter your choice:");
      scanf("%d",&ch);
      switch(ch)       {
      case 1: printf("Enter the name of the file:");
            scanf("%s",dir.fname[dir.fcnt]);
            dir.fcnt++;
            break;
      case 2: printf("Enter the name of the file:");
```

```
        scanf("%s",f);
        for(i=0;i<dir.fcnt;i++)             {
        if(strcmp(f, dir.fname[i])==0)              {
        printf("File %s is deleted ",f);
        strcpy(dir.fname[i],dir.fname[dir.fcnt-1]); break; } }
        if(i==dir.fcnt)
        printf("File %s not found",f);
        else
        dir.fcnt--;
        break;
case 3: printf("Enter the name of the file:");
        scanf("%s",f);
        for(i=0;i<dir.fcnt;i++)             {
        if(strcmp(f, dir.fname[i])==0)              {
        printf("File %s is found ", f);
        break;
        }
        }
        if(i==dir.fcnt)
        printf("File %s not found",f);
        break;
case 4: if(dir.fcnt==0)
           printf("\nDirectory Empty");
        else              {
        printf("\nThe Files are:");
        for(i=0;i<dir.fcnt;i++)
```

```
            printf("\t%s",dir.fname[i]);
          }
          break;
      default: exit(0);
      }
   }
}
```

Output:



Aim: write a c program to simulate two level directory organization technique

Program:

```
#include<string.h>
#include<stdio.h>
struct{
char dname[10],fname[10][10];
int fcnt;
}dir[10];
void main(){
int i,ch,dcnt,k;
char f[30], d[30];
dcnt=0;
while(1){
printf("\n\n1. Create Directory\t2. Create File\t3. Delete File");
printf("\n4. Search File\t\t5. Display\t6. Exit\t Enter your choice --");
scanf("%d",&ch);
switch(ch){
case 1: printf("\nEnter name of directory -- ");
scanf("%s", dir[dcnt].dname);
dir[dcnt].fcnt=0;
dcnt++;
printf("Directory created");
break;
case 2: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++)
if(strcmp(d,dir[i].dname)==0){
printf("Enter name of the file -- ");
scanf("%s",dir[i].fname[dir[i].fcnt]);
dir[i].fcnt++;
printf("File created");
break;
}
```

```
if(i==dcnt)
printf("Directory %s not found",d);
break;
case 3: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++){
if(strcmp(d,dir[i].dname)==0){
printf("Enter name of the file -- ");
scanf("%s",f);
for(k=0;k<dir[i].fcnt;k++){
if(strcmp(f, dir[i].fname[k])==0){
printf("File %s is deleted ",f);
dir[i].fcnt--;
strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);
goto jmp;
}}
printf("File %s not found",f);
goto jmp;
}}
printf("Directory %s not found",d);
jmp : break;
case 4: printf("\nEnter name of the directory -- ");
scanf("%s",d);
for(i=0;i<dcnt;i++){
if(strcmp(d,dir[i].dname)==0){
printf("Enter the name of the file -- ");
scanf("%s",f);
for(k=0;k<dir[i].fcnt;k++){
if(strcmp(f, dir[i].fname[k])==0){
printf("File %s is found ",f);
goto jmp1;
}
}
```

```
printf("File %s not found",f);
goto jmp1;
}
}
printf("Directory %s not found",d);
jmp1: break;
case 5: if(dcnt==0)
printf("\nNo Directory's ");
else{
printf("\nDirectory\tFiles");
for(i=0;i<dcnt;i++)
{
printf("\n%s\t\t",dir[i].dname);
for(k=0;k<dir[i].fcnt;k++)
printf("\t%s",dir[i].fname[k]);
}
}
break;
}
}
}
```

Output:

```
input


1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --1

Enter name of directory -- dir1
Directory created

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --2

Enter name of the directory -- dir1
Enter name of the file -- f1
File created

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --2

Enter name of the directory -- dir1
Enter name of the file -- f2
File created

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --1

Enter name of directory -- dir2
Directory created

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --2

Enter name of the directory -- dir2
Enter name of the file -- f3
File created
```

```
Enter name of the file -- f3
File created

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --5

Directory       Files
dir1                    f1      f2
dir2                    f3

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --4

Enter name of the directory -- dir1
Enter the name of the file -- f2
File f2 is found

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --3

Enter name of the directory -- dir1
Enter name of the file -- f2
File f2 is deleted

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --5

Directory       Files
dir1                    f1
dir2                    f3

1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit  Enter your choice --^C

...Program finished with exit code 0
Press ENTER to exit console.
```

Aim: write a c program to simulate contiguous file allocation method

Program:

```
#include<stdio.h>
#include<string.h>
struct fileTable{
char name[20];
int sb, nob;
int end;
}ft[30];
int isExists(char f[],int n){
int i;
for(i=0;i<n;i++){
if(strcmp(f, ft[i].name)==0){
return 1;
}
}
return 0;
}
int isValid(int base,int end_address,int n){
int i,flag=0;
for(i=0;i<n;i++){
if(ft[i].sb <= base && ft[i].end > base)
return 0;
if(ft[i].sb <= end_address && ft[i].end >= end_address)
return 0;
if(base < ft[i].sb && end_address > ft[i].end)
return 0;
if(base < ft[i].sb && end_address > ft[i].sb && end_address < ft[i].end)
return 0;
}
return 1;
}
```

```
void main(){
int i, j, n,nblocks,flag,check;
char s[20];
printf("Enter no of files : ");
scanf("%d",&n);
printf("\n No of blocks : ");
scanf("%d",&nblocks);
for(i=0;i<n;i++){
printf("\nEnter file name %d:",i+1);
scanf("%s",ft[i].name);
/* check whether name already exists */
check = isExists(ft[i].name,i);
while(check == 1){
printf("\nFile name already exists");
printf("\nEnter the name of the file -- ");
scanf("%s",ft[i].name);
check = isExists(ft[i].name,i);
}
printf("Enter starting block of file %d:",i+1);
scanf("%d",&ft[i].sb);
printf("Enter no of blocks in file %d :",i+1);
scanf("%d",&ft[i].nob);
/* check whether it in block range */
while(ft[i].sb + ft[i].nob > nblocks)
{
printf("\n Block size has exceeded\n");
printf("Enter starting block of file %d:",i+1);
scanf("%d",&ft[i].sb);
printf("Enter no of blocks in file %d :",i+1);
scanf("%d",&ft[i].nob);
}
ft[i].end = ft[i].sb + ft[i].nob;
/* checking whether another file exists at that location */
```

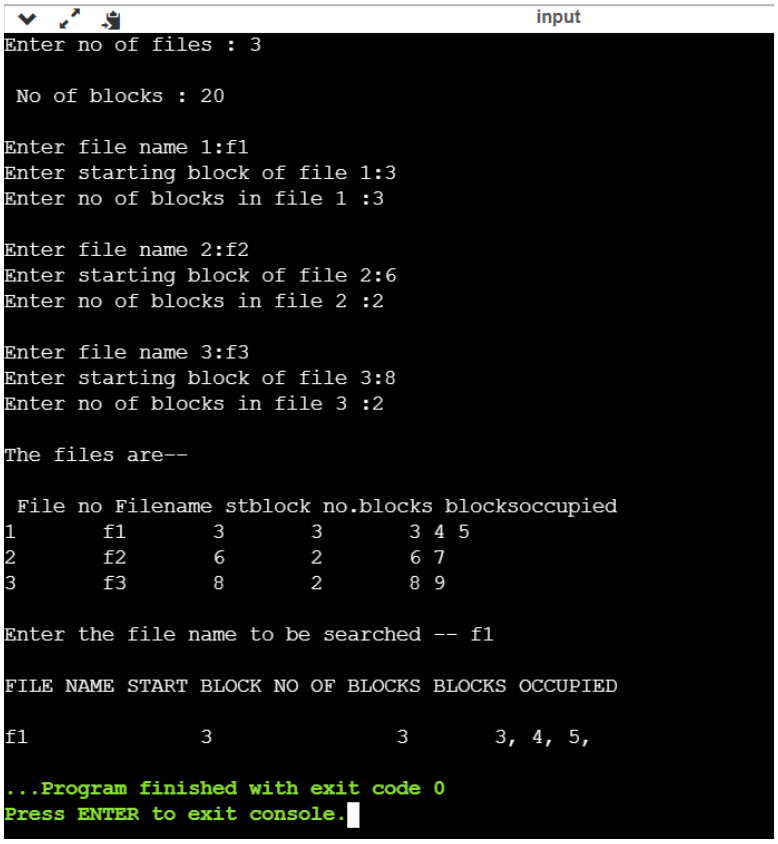
```
flag = isValid(ft[i].sb,ft[i].end,i);
while(flag == 0){
printf("\nAnother file exists at that location\n");
printf("Enter starting block of file %d:",i+1);
scanf("%d",&ft[i].sb);
printf("Enter no of blocks in file %d :",i+1);
scanf("%d",&ft[i].nob);
if(ft[i].sb + ft[i].nob > nblocks)
printf("\n Block size has exceeded");
ft[i].end = ft[i].sb + ft[i].nob;
flag = isValid(ft[i].sb,ft[i].end,i);
}
}
printf("\nThe files are--\n");
printf("\n File no Filename stblock no.blocks blocksoccupied\n");
for(i=0;i<n;i++){
printf("%d\t%s\t %d\t %d\t",i+1,ft[i].name,ft[i].sb,ft[i].nob);
for(j=0;j<ft[i].nob;j++){
printf(" %d",ft[i].sb+j);
}
printf("\n");
}
printf("\nEnter the file name to be searched -- ");
scanf("%s",s);
for(i=0;i<n;i++)
if(strcmp(s,ft[i].name)==0)
break;
if(i==n)
printf("\nFile Not Found");
else{
printf("\nFILE NAME START BLOCK NO OF BLOCKS BLOCKS OCCUPIED\n");
printf("\n%s\t\t%d\t\t%d\t",ft[i].name,ft[i].sb,ft[i].nob);
for(j=0;j<ft[i].nob;j++)
```

```
printf("%d, ",ft[i].sb+j);

}

}
```

Output:



Aim: Write a c program to simulate linked file allocation method

Program:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
struct fileTable{
   char name[20];
   int nob;
   struct block *sb;
}ft[30];
struct block{
   int bno;
   struct block *next;
};
int blockNum[100],top=0;
int isExists(int blno){
   int i;
   for(i=0;i<top;i++)    {
      if(blockNum[i] == blno)
         return 1;    }
   return 0;}
void main(){
   int i, j, n,nblocks,check,available;
   char s[20];
   struct block *temp;
   printf("Enter no of files :");
   scanf("%d",&n);
   printf("No of blocks : ");
   scanf("%d",&nblocks);
   available = nblocks;
   for(i=0;i<n;i++)    {
      printf("Enter file name %d:",i+1);
      scanf("%s",ft[i].name);
```

```
printf("Enter no of blocks in file %d :",i+1);
scanf("%d",&ft[i].nob);
/* availability of blocks */
while(ft[i].nob > available)       {
   printf("\nSufficient blocks are not available\n");
   printf("Enter no of blocks in file %d :",i+1);
   scanf("%d",&ft[i].nob);       }
available -= ft[i].nob;
/* whether the block is in range */
while(ft[i].nob > nblocks)       {
   printf("\n Block size has exceeded\n");
   printf("Enter no of blocks in file %d :",i+1);
   scanf("%d",&ft[i].nob);       }
ft[i].sb=(struct block*)malloc(sizeof(struct block));
temp = ft[i].sb;
printf("Enter block number :");
scanf("%d",&temp->bno);
while(temp->bno > nblocks)       {
   printf("Block size has exceeded\n");
   printf("Enter block number : ");
   scanf("%d",&temp->bno);
}
/* exists or not */
check = isExists(temp->bno);
while(check == 1)       {
   printf("It is already occupied");
   printf("Enter block number : ");
   scanf("%d",&temp->bno);
   check = isExists(temp->bno);       }
blockNum[top] = temp->bno;
top++;
temp->next=NULL;
for(j=1;j<ft[i].nob;j++)       {
```

```
        temp->next = (struct block*)malloc(sizeof(struct block));
        temp = temp->next;
        printf("Enter block number : ");
        scanf("%d",&temp->bno);
        while(temp->bno > nblocks)          {
           printf("\n Block size has exceeded\n");
           printf("Enter block number : ");
           scanf("%d",&temp->bno);          }
        /* exists or not */
        check = isExists(temp->bno);
        while(check == 1)          {
           printf("It is already occupied");
           printf(" Enter block number : ");
           scanf("%d",&temp->bno);
           check = isExists(temp->bno);
        }
        blockNum[top] = temp->bno;
        top++;       }
      temp->next = NULL;    }
   printf("The files are\n");
   printf("\nFILE NAME    NO OF BLOCKS     BLOCKS OCCUPIED");
   for(i=0;i<n;i++)    {
   printf("\n %s\t\t%d\t",ft[i].name,ft[i].nob);
   temp=ft[i].sb;
   for(j=0;j<ft[i].nob;j++)    {
   printf("%d\t ",temp->bno);
   temp = temp->next;
   }
   printf("  ");}
printf("\nEnter the file name to be searched -- ");
scanf("%s",s);
for(i=0;i<n;i++)
if(strcmp(s,ft[i].name)==0)
```

```
break;
if(i==n)
printf("\nFile Not Found");
else{
printf("\nFILE NAME NO OF BLOCKS BLOCKS OCCUPIED");
printf("\n %s\t\t%d\t",ft[i].name,ft[i].nob);
temp=ft[i].sb;
for(j=0;j<ft[i].nob;j++){
printf("%d\t ",temp->bno);
temp = temp->next;
}}}
```

Output: