**PROGRAM CODE: CONTIGUOUS ALOCATION**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct memory{

int block;

intnotfree;

char f[50];

};

typedefstruct memory Memory;

Memory disc[100];

intmemsize, blocksize, nb;

struct element{

int block;

int size;

char f[50];

struct element \*next;

};

typedefstruct element Element;

Element \*table = NULL;

void display(){

Element \*temp = table;

printf("File\tBlock\tSize\n");

while(temp!=NULL){

printf("%s\t%d\t%d\n",temp->f,temp->block,temp->size);

temp = temp->next;

}

}

voidinsertfile(char f[], int size, int block){

Element \*newnode = (Element\*)malloc(sizeof(Element));

newnode->next = NULL;

newnode->block = block;

newnode->size = size;

strcpy(newnode->f, f);

if(table == NULL)

table = newnode;

else{

newnode->next = table;

table = newnode;

}

}

void initialise(){

for(int i = 0; i <nb; i++){

disc[i].block = i;

disc[i].notfree = 0;

strcpy(disc[i].f, "---");

}

}

intcheckfree(int size){

for(int i = 0; i <nb; ){

if(disc[i].notfree == 0){

int j = i;

while(disc[j].notfree == 0 && j <nb){

if(j-i+1 == size)

return i;

j++;

}

i += (j+1);

}

else i++;

}

return -1;

}

int check(intind, int size){

if(disc[ind].notfree == 0){

for(int i = ind; i <ind+size&& i <nb; i++)

if(disc[i].notfree == 1)

return 0;

}

else

return 0;

if(ind + size - 1 <nb)

return 1;

else

return 0;

}

int allocate(int size, char name[]){

int flag = 0;

int b = size/blocksize;

if(size\*1.0/blocksize != b)

b++;

size = b;

if(checkfree(size) >= 0){

while(1){

intind = random()%nb;

if(check(ind,size) == 1){

flag = 1;

int i = ind;

do{

disc[i].notfree = 1;

strcpy(disc[i].f,name);

i++;

}while(i <b+ind);

insertfile(name,b,ind);

break;

}

}

}

else

printf("Not enough memory available!\n");

return (flag == 0) ? 0 : 1;

}

int main(){

intnf, size, block;

char f[50];

printf("Enter size of memory in KB: ");

scanf("%d", &memsize);

printf("Enter size of block in KB: ");

scanf("%d", &blocksize);

nb = memsize/blocksize;

initialise();

printf("Enter the no. of files: ");

scanf("%d", &nf);

for(int i = 0; i <nf; i++){

printf("Enter the file name: ");

scanf("%s",f);

printf("Enter the size of the file: ");

scanf("%d",&size);

allocate(size, f);

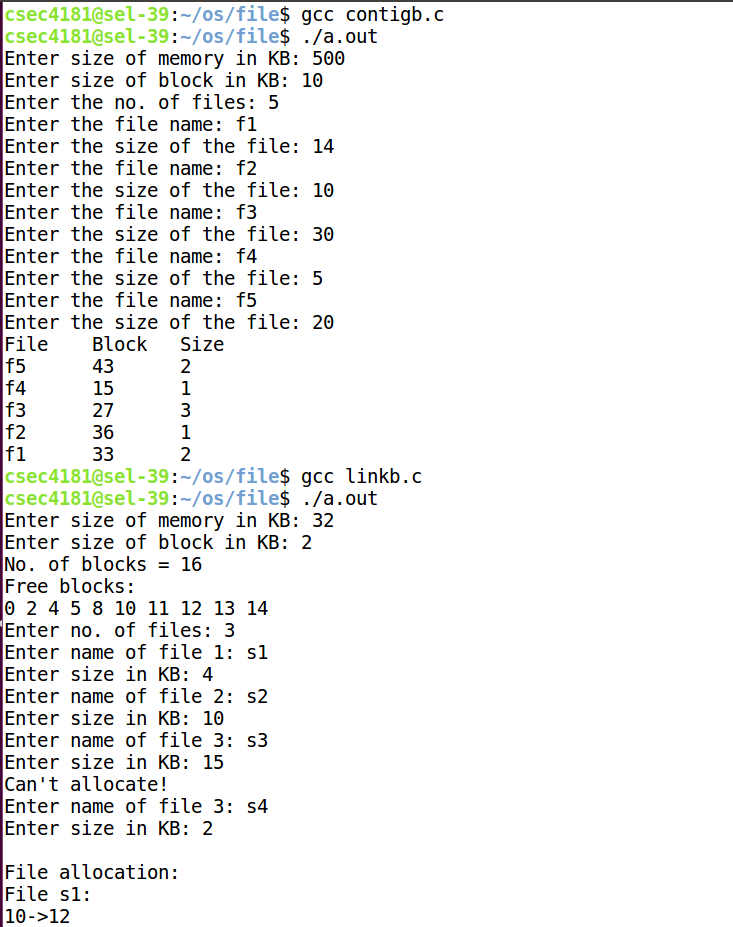
}

display();

return 0;

}

**OUTPUT:**

****

**PROGRAM CODE: LINKED ALLOCATION**

#include<stdio.h>

#include<stdlib.h>

struct list{

int block;

struct list \*next;

};

typedefstruct list List;

struct file{

char n[10];

int s, b;

List\* head;

};

typedefstruct file File;

File files[100];

intfreeb[100], mem, bsize, n, nf, c, r, f;

List\* newnode(int n){

List\* p = (List\*)malloc(sizeof(List));

p->block = n;

p->next = NULL;

return p;

}

void print(List\* head){

for(List\* t = head; t != NULL; t = t->next){

printf("%d",t->block);

if(t->next == NULL)

printf("\n");

else

printf("->");

}

}

int main(){

printf("Enter size of memory in KB: ");

scanf("%d", &mem);

printf("Enter size of block in KB: ");

scanf("%d", &bsize);

printf("No. of blocks = %d\n", mem/bsize);

n = mem/bsize;

nf = n;

for(int i = 0; i <= n/3; i++){

r = random()%n;

if(freeb[r] == 1)

i--;

else{

freeb[r] = 1;

nf--;

}

}

printf("Free blocks:\n");

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

if(freeb[i] == 0)

printf("%d ", i);

}

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

scanf("%d", &f);

for(int i = 0; i < f; i++){

printf("Enter name of file %d: ", i+1);

scanf("%s", (files[c].n));

printf("Enter size in KB: ");

scanf("%d", &files[c].s);

files[c].b = files[c].s/bsize;

if(files[c].s\*1.0/bsize> files[c].b)

(files[c].b)++;

if(files[c].b >nf){

printf("Can't allocate!\n");

i--;

}

else{

List \*t, \*p;

for(int j = 0; j < files[c].b; j++){

r = random()%n;

if(freeb[r] == 0){

freeb[r] = 1;

nf--;

t = newnode(r);

if(j == 0){

files[c].head = t;

p = files[c].head;

}

else{

p->next = t;

p = t;

}

}

else

j--;

}

c++;

}

if(nf == 0){

printf("Memory over!\n");

f = c;

break;

}

}

printf("\nFile allocation:\n");

for(int i = 0; i < f; i++){

printf("File %s:\n",files[i].n);

print(files[i].head);

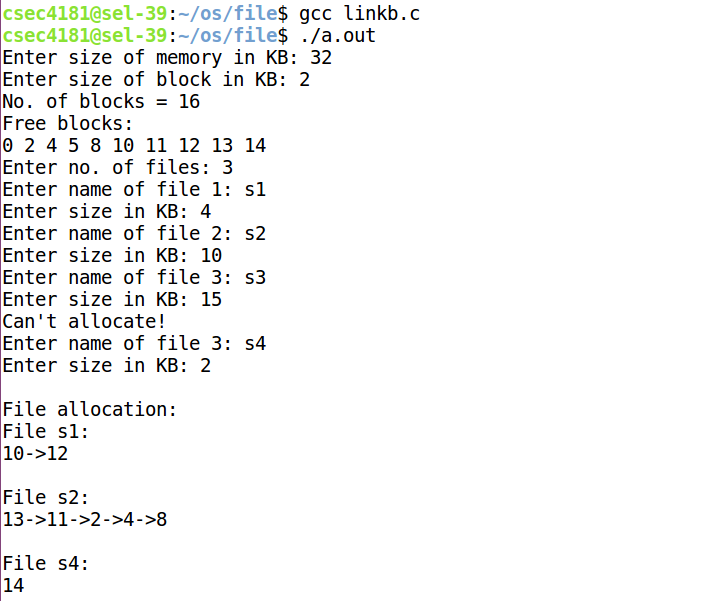
printf("\n");

}

return 0;

}

**OUTPUT:**

****

**PROGRAM CODE: INDEXED ALLOCATION**

#include<stdio.h>

#include<stdlib.h>

typedefstruct file{

char n[10];

int s, b;

int i;

}File;

typedefstructindexblock{

intblockid;

int blocks[100];

}IndexBlock;

IndexBlockindex[100];

File files[100];

intfreeb[100], mem, bsize, n, nf, c, r, f;

void print(int i){

printf("Index block no.: %d\nBlock table: ", index[i].blockid);

for(int j = 0; j < files[i].b; j++)

printf("%d ", index[i].blocks[j]);

}

int main(){

printf("Enter the size of memory in KB: ");

scanf("%d", &mem);

printf("Enter size of the block in KB: ");

scanf("%d", &bsize);

printf("No. of blocks = %d\n", mem/bsize);

n = mem/bsize;

nf = n;

for(int i = 0; i <= n/3; i++){

r = random()%n;

if(freeb[r] == 1)

i--;

else{

freeb[r] = 1;

nf--;

}

}

printf("Free blocks:\n");

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

if(freeb[i] == 0)

printf("%d ", i);

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

scanf("%d", &f);

for(int i = 0; i < f; i++){

printf("Enter name of file %d: ", i+1);

scanf("%s", files[c].n);

printf("Enter size in KB: ");

scanf("%d", &files[c].s);

files[c].b = files[c].s/bsize;

if(files[c].s\*1.0/bsize> files[c].b)

(files[c].b)++;

if(files[c].b + 1 >nf){

printf("Can't allocate!\n");

i--;

}

else{

do{

r = random()%n;

}while(freeb[r] == 1);

index[c].blockid = r;

files[c].i = r;

freeb[r] = 1;

nf--;

for(int j = 0; j < files[c].b; j++){

r = random()%n;

if(freeb[r] == 0){

freeb[r] = 1;

nf--;

index[c].blocks[j] = r;

}

else

j--;

}

c++;

}

if(nf == 0){

printf("Memory over!\n");

f = c;

break;

}

}

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

for(int i = 0; i < f; i++){

printf("File %s:\n",files[i].n);

print(i);

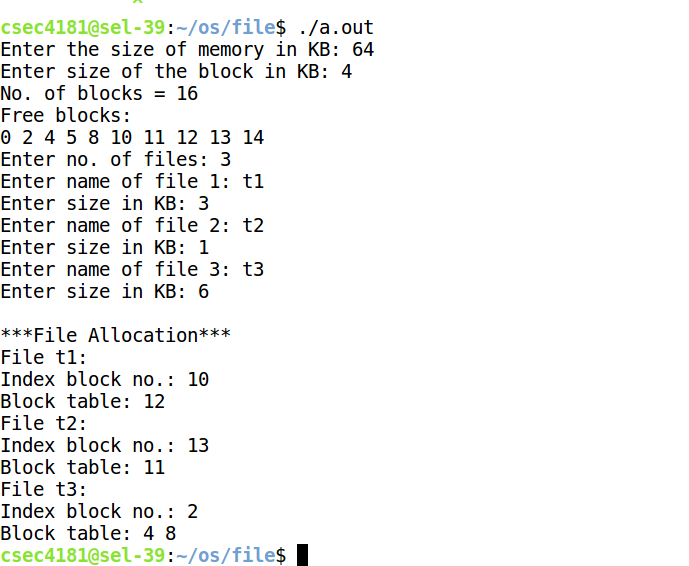
printf("\n");

}

return 0;

}

**OUTPUT:**

****

**PROGRAM CODE: SINGLE LEVEL**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct files{

charfname[10];

};

typedefstruct files File;

File\* root[50];

intfcount = 0;

File\* new\_file(char s[]){

File\* n = (File\*)malloc(sizeof(File));

strcpy(n->fname, s);

return n;

}

intsearch\_file(char s[]){

int flag = 0;

for(int i = 0; i <fcount; i++){

if(root[i] != NULL){

if(strcmp(root[i]->fname,s) == 0){

flag = 1;

break;

}

}

}

return flag;

}

voidinsert\_file(char s[]){

if(search\_file(s) == 1){

printf("File %s already exists!\n", s);

return;

}

root[fcount] = new\_file(s);

fcount++;

printf("Created!\n");

}

void display(File\* d[]){

printf("Contents of root:\n");

if(fcount == 0){

printf("Empty!\n");

return;

}

for(int i = 0; i <fcount; i++){

if(root[i] != NULL)

printf("%s\t", root[i]->fname);

}

printf("\n");

}

void main(){

int c;

while(1){

printf("1. New File\n");

printf("2. Display all files\n");

printf("3. Exit\n");

printf("Enter choice: ");

scanf("%d",&c);

if(c==1){

char s[50];

printf("Enter file name: ");

scanf("%s",s);

insert\_file(s);

}

else if(c==2)

display(root);

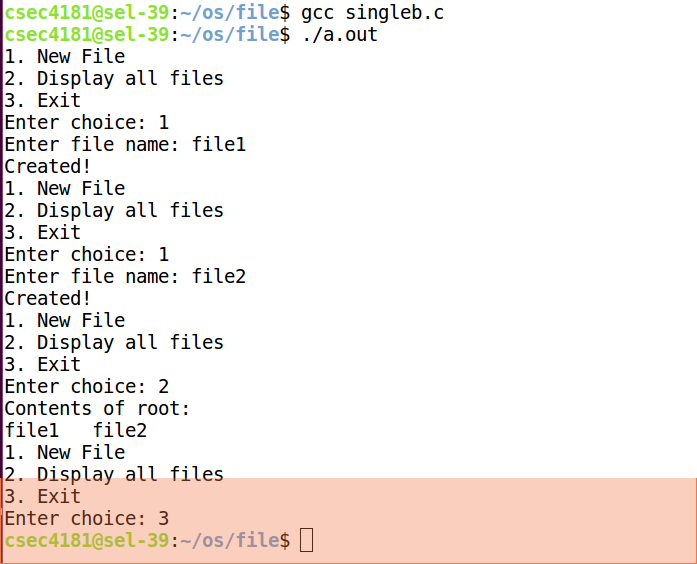
else

break;

}

}

**OUTPUT:**

****

**PROGRAM CODE: TWO LEVEL**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedefstruct files{

charfname[10];

}File;

typedefstruct directory{

charfname[10];

int c;

File\* l[5];

}Directory;

typedefstruct unit{

int d;

void \*p;

}Unit;

Unit root[50];

int count = 0;

File\* new\_file(char s[]){

File\* n = (File\*)malloc(sizeof(File));

strcpy(n->fname, s);

return n;

}

Directory\* new\_dir(char s[]){

Directory\* n = (Directory\*)malloc(sizeof(Directory));

strcpy(n->fname, s);

n->c = 0;

for(int i = 0; i < 5; i++)

n->l[i] = NULL;

return n;

}

intsearch\_file(char s[]){

int flag = 0;

for(int i = 0; i < count; i++){

if(root[i].p != NULL){

if(strcmp(((File\*)(root[i].p))->fname,s) == 0 && root[i].d == 0){

flag = 1;

break;

}

}

}

return flag;

}

Directory\* search\_dir(char s[]){

Directory\* flag = NULL;

for(int i = 0; i < count; i++){

if(root[i].p != NULL){

if(strcmp(((Directory\*)(root[i].p))->fname,s) == 0 && root[i].d == 1){

flag = ((Directory\*)(root[i].p));

break;

}

}

}

return flag;

}

voidinsert\_file(char s[]){

if(count >= 50){

printf("Full!\n");

return;

}

if(search\_file(s) == 1){

printf("File %s already exists!\n", s);

return;

}

if(search\_dir(s) != NULL){

printf("Directory named %s already exists!\n", s);

return;

}

root[count].p = new\_file(s);

root[count].d = 0;

count++;

printf("Created!\n");

}

voidinsert\_file\_dir(Directory\* d, char s[]){

int i, pos;

if(d->c >= 5){

printf("Directory full!\n");

return;

}

for(i = 0; i < 5; i++){

if(d->l[i] != NULL){

if(strcmp(d->l[i]->fname, s)==0){

printf("File already exists!\n");

return;

}

}

else{

pos = i;

i = 5;

}

}

d->l[pos] = new\_file(s);

d->c = d->c + 1;

printf("Created!\n");

}

voidinsert\_dir(char s[]){

if(count >= 50){

printf("Full!\n");

return;

}

if(search\_dir(s) != NULL){

printf("Directory %s already exists!\n", s);

return;

}

if(search\_file(s) == 1){

printf("File named %s already exists!\n", s);

return;

}

root[count].p = new\_dir(s);

root[count].d = 1;

count++;

printf("Created!\n");

}

void display(Unit d[]){

printf("Contents of root:\n");

if(count == 0){

printf("Empty!\n");

return;

}

int ch = 0;

printf("Files:\n");

for(int i = 0; i < count; i++){

if(root[i].p != NULL){

if(root[i].d == 0){

printf("%s ",((File\*)(root[i].p))->fname);

ch++;

}

}

}

if(ch == 0)

printf("None!");

printf("\nDirectories:\n");

ch = 0;

int dc = 0;

for(int i = 0; i < count; i++){

if(root[i].p != NULL){

if(root[i].d == 1){

ch++;

printf("%s ",((Directory\*)(root[i].p))->fname);

}

}

}

if(ch == 0)

printf("None!");

printf("\n");

ch = 0;

for(int i = 0; i < count; i++){

if(root[i].p != NULL){

if(root[i].d == 1){

ch++;

printf("Contents of %s:\n",((Directory\*)(root[i].p))->fname);

dc = 0;

for(int j = 0; j < 5; j++)

if(((Directory\*)(root[i].p))->l[j] != NULL){

printf("%s ", ((Directory\*)(root[i].p))->l[j]->fname);

dc++;

}

if(dc == 0)

printf("None!");

printf("\n");

}

}

}

printf("\n");

}

int main(){

int c;

while(1){

printf("1.Create new File\n");

printf("2.Create new Directory\n");

printf("3. Display all files\n");

printf("4. Exit\n");

printf("Enter choice: ");

scanf("%d",&c);

if(c==1){

char d[10], s[10];

printf("Enter root to create file in the root directory.\nEnter root/directory to create file in the sub-directory.\nEnter directory: ");

scanf("%s",d);

printf("Enter file name: ");

scanf("%s", s);

if(strcmp(d,"root")!=0){

char\* n = strtok(d, "/");

n = strtok(NULL, "/");

Directory\* dir = search\_dir(n);

if(dir != NULL){

insert\_file\_dir(dir, s);

}

else

printf("No such directory!\n");

}

else if(strcmp(d,"root")==0){

insert\_file(s);

}

}

else if(c==2){

char d[10];

printf("Enter directory name: ");

scanf("%s", d);

insert\_dir(d);

}

else if(c==3)

display(root);

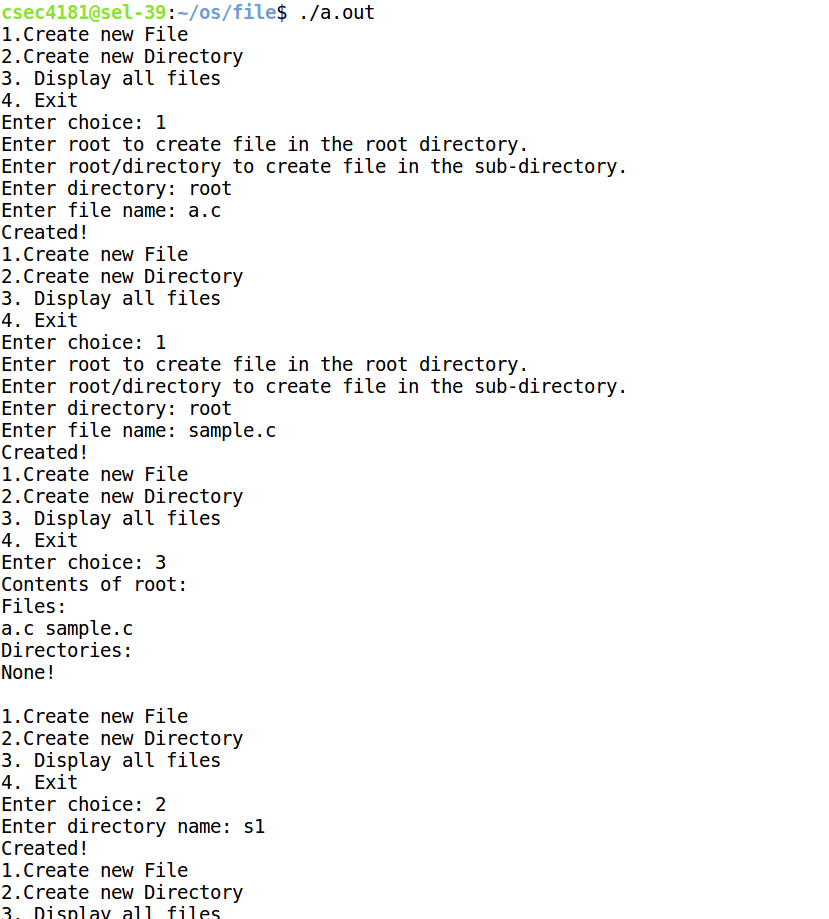
else

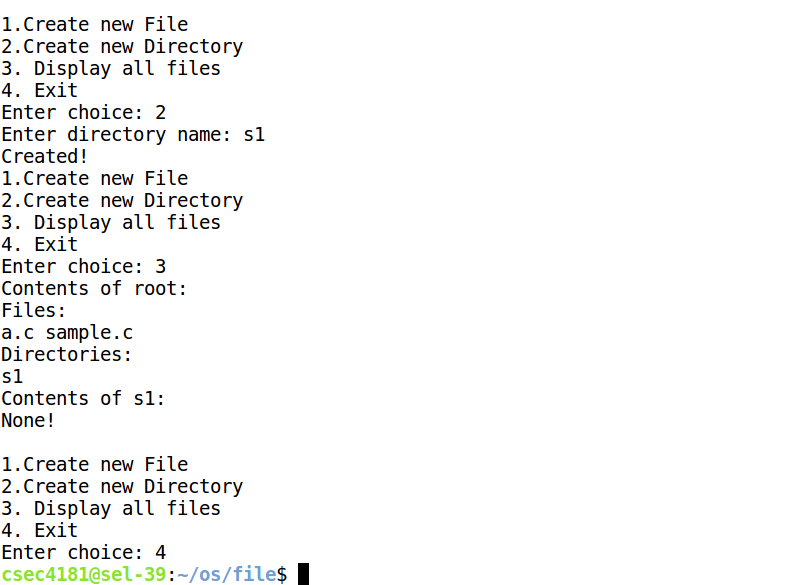
break;

}

}

**OUTPUT:**

****

****

**PROGRAM CODE: TREE**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedefstruct files {

charfname[10];

}File;

typedefstruct directory {

chardname[10];

struct directory \*d1,\*d2,\*d3;

File \*f1,\*f2;

}Directory;

Directory \*root = NULL;

voidinsert\_directory(char s[])

{

Directory\* temp=root;

char \*t = strtok(s,"/");

t = strtok(NULL,"/");

while(t != NULL) {

if(temp->d1 != NULL &&strcmp(t,temp->d1->dname)==0)

{

temp = temp->d1;

}

else if(temp->d2 != NULL &&strcmp(t,temp->d2->dname)==0)

{

temp=temp->d2;

}

else if(temp->d3 != NULL &&strcmp(t,temp->d3->dname)==0)

{

temp=temp->d3;

}

t = strtok(NULL,"/");

}

if(t==NULL)

{

if(temp->d1==NULL||temp->d2==NULL||temp->d3==NULL)

{

char d[10];

printf("Enter the directory name: ");

scanf("%s",d);

Directory\* newdir = (Directory\*)malloc(sizeof(Directory));

strcpy(newdir->dname,d);

newdir->d1=NULL;

newdir->d2=NULL;

newdir->d3=NULL;

newdir->f1=NULL;

newdir->f2=NULL;if(temp->d1 == NULL)

{

temp->d1 = newdir;

}

else if(temp->d2 == NULL &&strcmp(d,temp->d1->dname)!=0)

{

temp->d2 = newdir;

}

else if(strcmp(d,temp->d1->dname) != 0 &&strcmp(d,temp->d2->dname)!=0)

{

temp->d3 = newdir;

}

else if(strcmp(d,temp->d1->dname) == 0 || strcmp(d,temp->d2->dname)==0)

printf("Duplicate directories not allowed!\n");

}

elseprintf("Directory limit exceeded!\n");

}

}

voidinsert\_file(char s[])

{

Directory\* temp=root;

char temp1[100];

strcpy(temp1,s);

char \*t = strtok(s,"/");

t = strtok(NULL,"/");

while(t != NULL) {

if(temp->d1 != NULL &&strcmp(t,temp->d1->dname)==0)

{

temp = temp->d1;

}

else if(temp->d2 != NULL &&strcmp(t,temp->d2->dname)==0)

{

temp = temp->d2;

}

else if(temp->d3 != NULL &&strcmp(t,temp->d3->dname)==0)

{

temp = temp->d3;

}

t = strtok(NULL,"/");

}

if(t == NULL)

{

if(temp->f1 == NULL || temp->f2 == NULL)

{

char d[10];

printf("Enter the file name: ");

scanf("%s",d);

File \*newfile = (File\*)malloc(sizeof(File));

strcpy(newfile->fname,d);

if(temp->f1 == NULL)

{

temp->f1=newfile;

}

else if(temp->f2 == NULL)

{

temp->f2=newfile;

}

}

else

printf("File limit exceeded!");

}

}File\* get\_file\_pointer(char s[])

{

char \*t = strtok(s,"/");

char \*g;

Directory \*temp = root;

while(t != NULL) {

if(temp->d1 != NULL &&strcmp(t,temp->d1->dname)==0)

{

temp = temp->d1;

}

else if(temp->d2 != NULL &&strcmp(t,temp->d2->dname)==0)

{

temp = temp->d2;

}

else if(temp->d3 != NULL &&strcmp(t,temp->d3->dname)==0)

{

temp=temp->d3;

}

g = t;

t = strtok(NULL,"/");

if(t==NULL)

{

if(strcmp(temp->f1->fname,g)==0)

return temp->f1;

else if(strcmp(temp->f2->fname,g)==0)

return temp->f2;

else

{

printf("No such file!\n");

return NULL;

}

}

}

return NULL;

}

Directory\* get\_directory\_pointer(char s[])

{

char \*t = strtok(s,"/");

char \*g;

Directory \*temp = root;

while(t != NULL){

if(temp->d1 != NULL &&strcmp(t,temp->d1->dname)==0)

{

temp = temp->d1;

}

else if(temp->d2 != NULL &&strcmp(t,temp->d2->dname)==0)

{

temp = temp->d2;

}

else if(temp->d3 != NULL &&strcmp(t,temp->d3->dname)==0)

{

temp = temp->d3;

}

g = t;

t = strtok(NULL,"/");

if(t == NULL)

{

return temp;

}

}

return NULL;

}

voiddisplay\_file(File\* f, char s[])

{printf("%s\t\t%s\n",f->fname,s);

}

void display(Directory\* r, char s[])

{

if(r!=NULL)

{

strcat(s,r->dname);

strcat(s,"/");

if(r->f1 != NULL)

{

display\_file(r->f1,s);

}

if(r->f2!=NULL)

{

display\_file(r->f2,s);

}

if(r->d1 != NULL) {

char s1[50];

strcpy(s1, s);

display(r->d1,s1);

}

if(r->d2 != NULL) {

char s1[50];

strcpy(s1, s);

display(r->d2,s1);

}

if(r->d3 != NULL) {

char s1[50];

strcpy(s1, s);

display(r->d3,s1);

}

}

}

int main()

{

root = (Directory\*)malloc(sizeof(Directory));

strcpy(root->dname,"root");

root->d1=NULL;

root->d2=NULL;

root->d3=NULL;

root->f1=NULL;

root->f2=NULL;

int c;

while(1)

{

printf("1. Insert a Directory\n");

printf("2. Insert a File\n");

printf("3. Display all files\n");

printf("4. Exit\n");

printf("Enter choice: ");

scanf("%d",&c);

if(c==1)

{

char s[50];

printf("Path format:\n");

printf("root/ (or) root - to insert in root\n");

printf("root/directory/ - to insert into directory in root\n");

printf("Enter the path: ");

scanf("%s",s);

insert\_directory(s);

}

else if(c==2)

{

char s[50];printf("Path format:\n");

printf("root/ (or) root - to insert file in root\n");

printf("root/directory/ - to insert file into directory in root\n");

printf("Enter the path: ");

scanf("%s",s);

insert\_file(s);

}

else if(c==3)

{

char s[400];

strcpy(s,"");

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

display(root,s);

}

else {

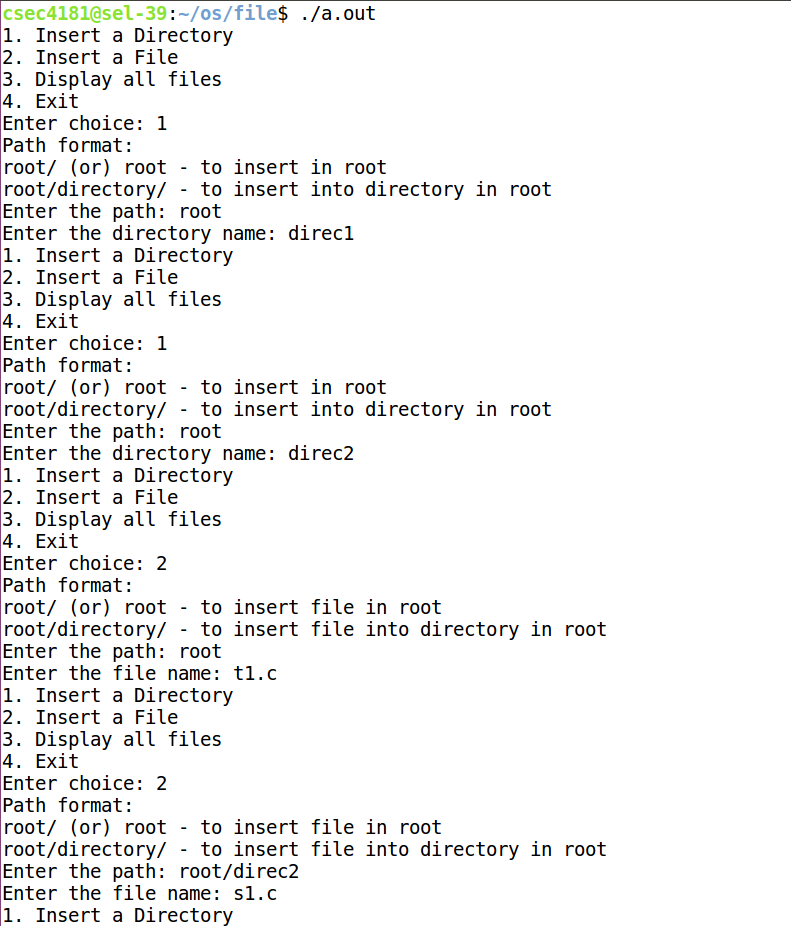
break;

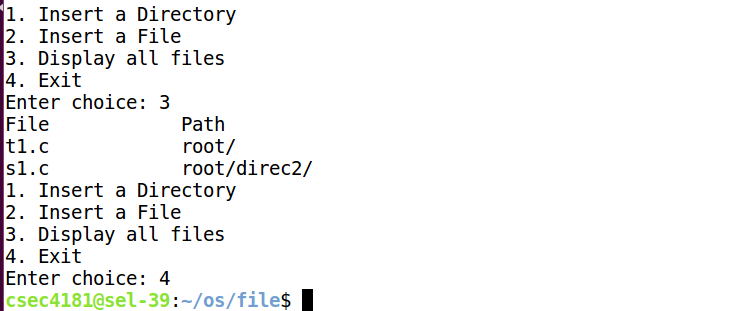
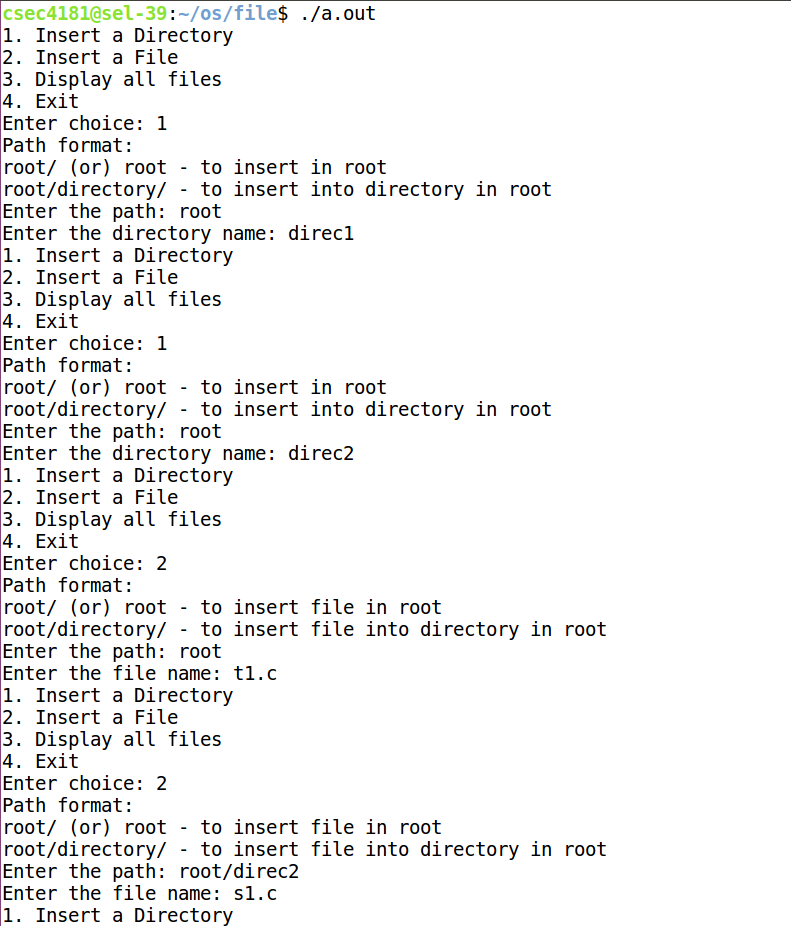
}

}

}

**OUTPUT:**

****

****

**PROGRAM CODE: DAG**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedefstruct files {

charfname[10];

}File;

typedefstruct directory {

chardname[10];

struct directory \*d1,\*d2,\*d3;

File \*f1,\*f2;

}Directory;

Directory \*root = NULL;

voidinsert\_directory(char s[])

{

Directory\* temp=root;

char \*t = strtok(s,"/");

t = strtok(NULL,"/");

while(t != NULL) {

if(temp->d1 != NULL &&strcmp(t,temp->d1->dname)==0)

{

temp = temp->d1;

}

else if(temp->d2 != NULL &&strcmp(t,temp->d2->dname)==0)

{

temp=temp->d2;

}

else if(temp->d3 != NULL &&strcmp(t,temp->d3->dname)==0)

{

temp=temp->d3;

}

t = strtok(NULL,"/");

}

if(t==NULL)

{

if(temp->d1==NULL||temp->d2==NULL||temp->d3==NULL)

{

char d[10];

printf("Enter the directory name: ");

scanf("%s",d);

Directory\* newdir = (Directory\*)malloc(sizeof(Directory));

strcpy(newdir->dname,d);

newdir->d1=NULL;

newdir->d2=NULL;

newdir->d3=NULL;

newdir->f1=NULL;

newdir->f2=NULL;

if(temp->d1 == NULL)

{

temp->d1 = newdir;

}

else if(temp->d2 == NULL &&strcmp(d,temp->d1->dname)!=0)

{

temp->d2 = newdir;

}

else if(strcmp(d,temp->d1->dname) != 0 &&strcmp(d,temp->d2->dname)!=0)

{

temp->d3 = newdir;

}

else if(strcmp(d,temp->d1->dname) == 0 || strcmp(d,temp->d2->dname)==0)

printf("Duplicate directories not allowed!\n");

}}

elseprintf("Directory limit exceeded!\n");

}

voidinsert\_file(char s[])

{

Directory\* temp=root;

char temp1[100];

strcpy(temp1,s);

char \*t = strtok(s,"/");

t = strtok(NULL,"/");

while(t != NULL) {

if(temp->d1 != NULL &&strcmp(t,temp->d1->dname)==0)

{

temp = temp->d1;

}

else if(temp->d2 != NULL &&strcmp(t,temp->d2->dname)==0)

{

temp = temp->d2;

}

else if(temp->d3 != NULL &&strcmp(t,temp->d3->dname)==0)

{

temp = temp->d3;

}

t = strtok(NULL,"/");

}

if(t == NULL)

{

if(temp->f1 == NULL || temp->f2 == NULL)

{

char d[10];

printf("Enter the file name: ");

scanf("%s",d);

File \*newfile = (File\*)malloc(sizeof(File));

strcpy(newfile->fname,d);

if(temp->f1 == NULL)

{

temp->f1=newfile;

}

else if(temp->f2 == NULL)

{

temp->f2=newfile;

}

}

else

printf("File limit exceeded!");

}

}

File\* get\_file\_pointer(char s[])

{

char \*t = strtok(s,"/");

char \*g;

Directory \*temp = root;

while(t != NULL) {

if(temp->d1 != NULL &&strcmp(t,temp->d1->dname)==0)

{

temp = temp->d1;

}

else if(temp->d2 != NULL &&strcmp(t,temp->d2->dname)==0)

{

temp = temp->d2;

}

else if(temp->d3 != NULL &&strcmp(t,temp->d3->dname)==0)

{

temp=temp->d3;}

g = t;

t = strtok(NULL,"/");

if(t==NULL)

{

if(strcmp(temp->f1->fname,g)==0)

return temp->f1;

else if(strcmp(temp->f2->fname,g)==0)

return temp->f2;

else

{

printf("No such file!\n");

return NULL;

}

}

}

return NULL;

}

Directory\* get\_directory\_pointer(char s[])

{

char \*t = strtok(s,"/");

char \*g;

Directory \*temp = root;

while(t != NULL) {

if(temp->d1 != NULL &&strcmp(t,temp->d1->dname)==0)

{

temp = temp->d1;

}

else if(temp->d2 != NULL &&strcmp(t,temp->d2->dname)==0)

{

temp = temp->d2;

}

else if(temp->d3 != NULL &&strcmp(t,temp->d3->dname)==0)

{

temp = temp->d3;

}

g = t;

t = strtok(NULL,"/");

if(t == NULL)

{

return temp;

}

}

return NULL;

}

voidcreate\_link(char s1[], char s2[])

{

File\* f1 = get\_file\_pointer(s1);

char a[300];

Directory\* d2 = get\_directory\_pointer(s2);

if(f1 != NULL) {

if(d2->f1 == NULL)

{

d2->f1 = f1;

}

else if(d2->f2 == NULL)

{

d2->f2 = f1;

}

else

{

printf("Not enough space to make the link!\n");

}

}}

voiddisplay\_file(File\* f, char s[])

{

printf("%s\t\t%s\n",f->fname,s);

}

void display(Directory\* r, char s[])

{

if(r!=NULL)

{

strcat(s,r->dname);

strcat(s,"/");

if(r->f1 != NULL)

{

display\_file(r->f1,s);

}

if(r->f2!=NULL)

{

display\_file(r->f2,s);

}

if(r->d1 != NULL) {

char s1[50];

strcpy(s1, s);

display(r->d1,s1);

}

if(r->d2 != NULL) {

char s1[50];

strcpy(s1, s);

display(r->d2,s1);

}

if(r->d3 != NULL) {

char s1[50];

strcpy(s1, s);

display(r->d3,s1);

}

}

}

int main()

{

root = (Directory\*)malloc(sizeof(Directory));

strcpy(root->dname,"root");

root->d1=NULL;

root->d2=NULL;

root->d3=NULL;

root->f1=NULL;

root->f2=NULL;

int c;

while(1)

{

printf("1. Insert a Directory\n");

printf("2. Insert a File\n");

printf("3. Create a link to a file\n");

printf("4. Display all files\n");

printf("5. Exit\n");

printf("Enter choice: ");

scanf("%d",&c);

if(c==1)

{

char s[50];

printf("Path format:\n");

printf("root/ (or) root - to insert in root\n");

printf("root/directory/ - to insert into directory in root\n");

printf("Enter the path: ");

scanf("%s",s);

insert\_directory(s);}

else if(c==2)

{

char s[50];

printf("Path format:\n");

printf("root/ (or) root - to insert file in root\n");

printf("root/directory/ - to insert file into directory in root\n");

printf("Enter the path: ");

scanf("%s",s);

insert\_file(s);

}

else if(c==3)

{

char s1[50];

char s2[50];

printf("Enter path of file (including file name): ");

scanf("%s",s1);

printf("Enter path of directory to create link in: ");

scanf("%s",s2);

create\_link(s1,s2);

}

else if(c==4)

{

char s[400];

strcpy(s,"");

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

display(root,s);

}

else {

break;

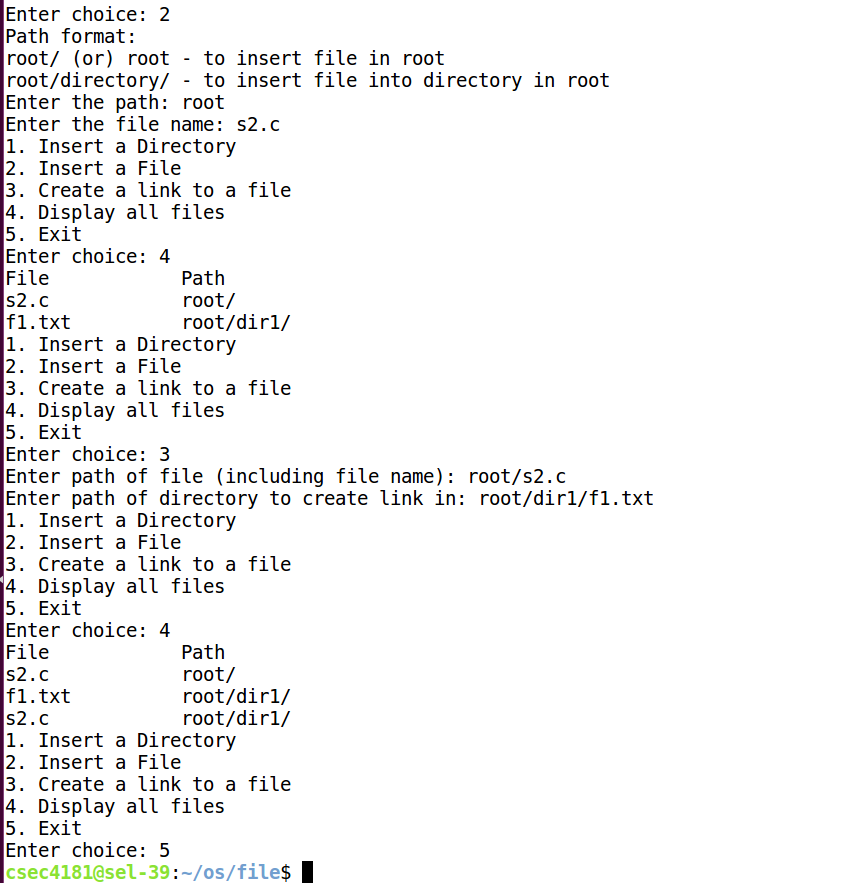
}

}

}

**OUTPUT:**

****

****