**PROGRAM CODE:**

#include<stdio.h>

typedef struct

{

int roll;

char name[40];

char add[80];

long mobile;

}student;

student\* input(int n)

{

static student s[50];

int i;

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

{

printf("\nEnter details of student %d\n",i+1);

printf("Enter Roll.No: ");

scanf("%d",&s[i].roll);

printf("Enter name: ");

scanf("%s",s[i].name);

getchar();

printf("Enter address: ");

gets(s[i].add);

printf("Enter mobile number: ");

scanf("%ld",&s[i].mobile);

}

return s;

}

void display(int n,student \*record)

{int i;

printf("\nROLL No NAME ADDRESS\t\t\t\t\tMOBILE NUMBER\n");

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

{

printf("%d\t",((record+i))->roll);

printf("%s ",(record+i)->name);

printf("%s\t\t",(record+i)->add);

printf("%ld\n",(record+i)->mobile);

}

}

void search(int n,student \*record)

{

char name[50];

int flag=0,i;

printf("\nEnter name whose mobile no is to be displayed: ");

scanf("%s",&name);

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

{

if(strcmp(name,(record+i)->name)==0)

{ printf("\nThe mobile no of %s is %ld\n",(record+i)->name,(record+i)->mobile);

flag=1;

break;

}

}

if (flag==0)

printf("Student data not found");

}

void main()

{

student \*record;

int n;

printf("Enter no of students: ");

scanf("%d",&n);

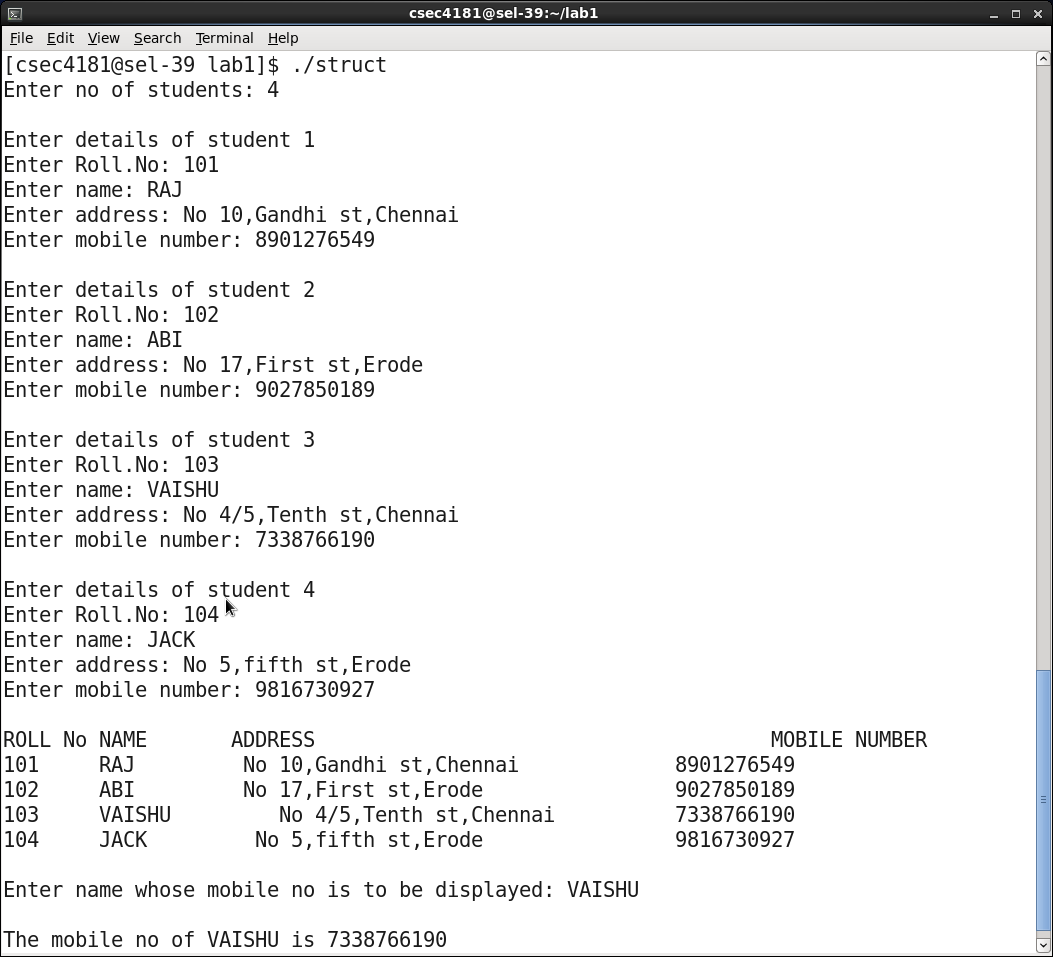
record=input(n);

display(n,record);

search(n,record);

}

**OUTPUT:**



**PROGRAM CODE:**

#include<stdio.h>

#include<string.h>

typedef struct

{

int roll;

char name[40];

char add[80];

long mobile;

}student;

student\* input(int n)

{

static student s[50];

int i;

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

{

printf("\nEnter details of student %d\n",i+1);

printf("Enter Roll.No: ");

scanf("%d",&s[i].roll);

printf("Enter name: ");

scanf("%s",s[i].name);

getchar();

printf("Enter address: ");

gets(s[i].add);

printf("Enter mobile number: ");

scanf("%ld",&s[i].mobile);

}

return s;

}

void display(int n,student \*record)

{

int i;

printf("\nROLL No NAME ADDRESS\t\t\t\tMOBILE NUMBER\n");

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

{

printf("%d\t",((record+i))->roll);

printf("%s ",(record+i)->name);

printf("%s\t\t",(record+i)->add);

printf("%ld\n",(record+i)->mobile);

}

}

student\* sort(int n,student \*record)

{

int i,j,minidx;

student temp;

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

{

minidx=i;

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

if (strcmp((record+minidx)->name,(record+j)->name)>0)

minidx=j;

temp=\*(record+i);

\*(record+i)=\*(record+minidx);

\*(record+minidx)=temp;

}

return record;

}

void search(int n,student \*record)

{

char name[50];

int start=0,end=n-1,mid,flag=0;

printf("\nEnter name whose mobile no is to be displayed: ");

scanf("%s",&name);

while(start<=end)

{

mid=(start+end)/2;

if(strcmp((record+mid)->name,name)==0)

{ printf("\nThe mobile no of %s is %ld\n",(record+mid)->name,(record+mid)->mobile);

flag=1;

break;

}

else if(strcmp((record+mid)->name,name)>0)

end=mid-1;

else if(strcmp((record+mid)->name,name)<0)

start=mid+1;

}

if (flag==0)

printf("Student data not found");

}

void main()

{

student \*record;

int n;

printf("Enter no of students: ");

scanf("%d",&n);

record=input(n);

display(n,record);

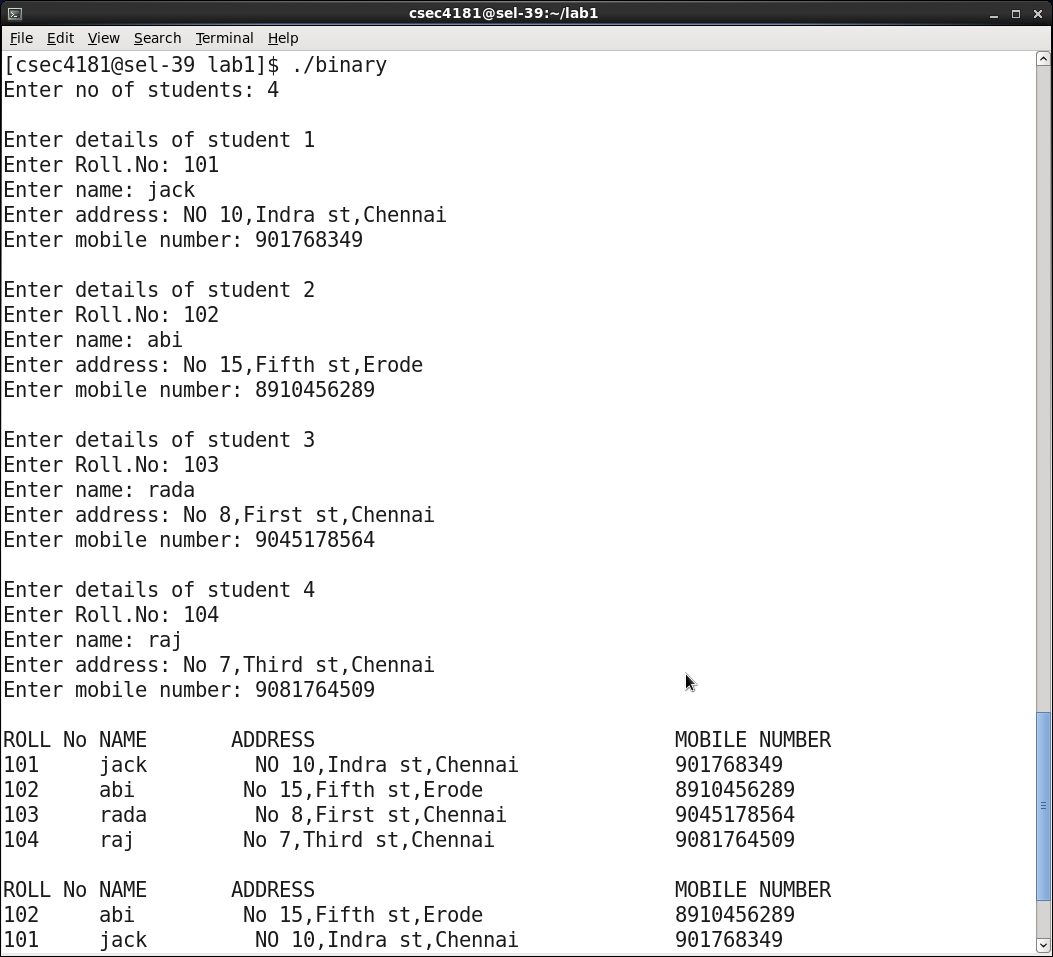
record=sort(n,record);

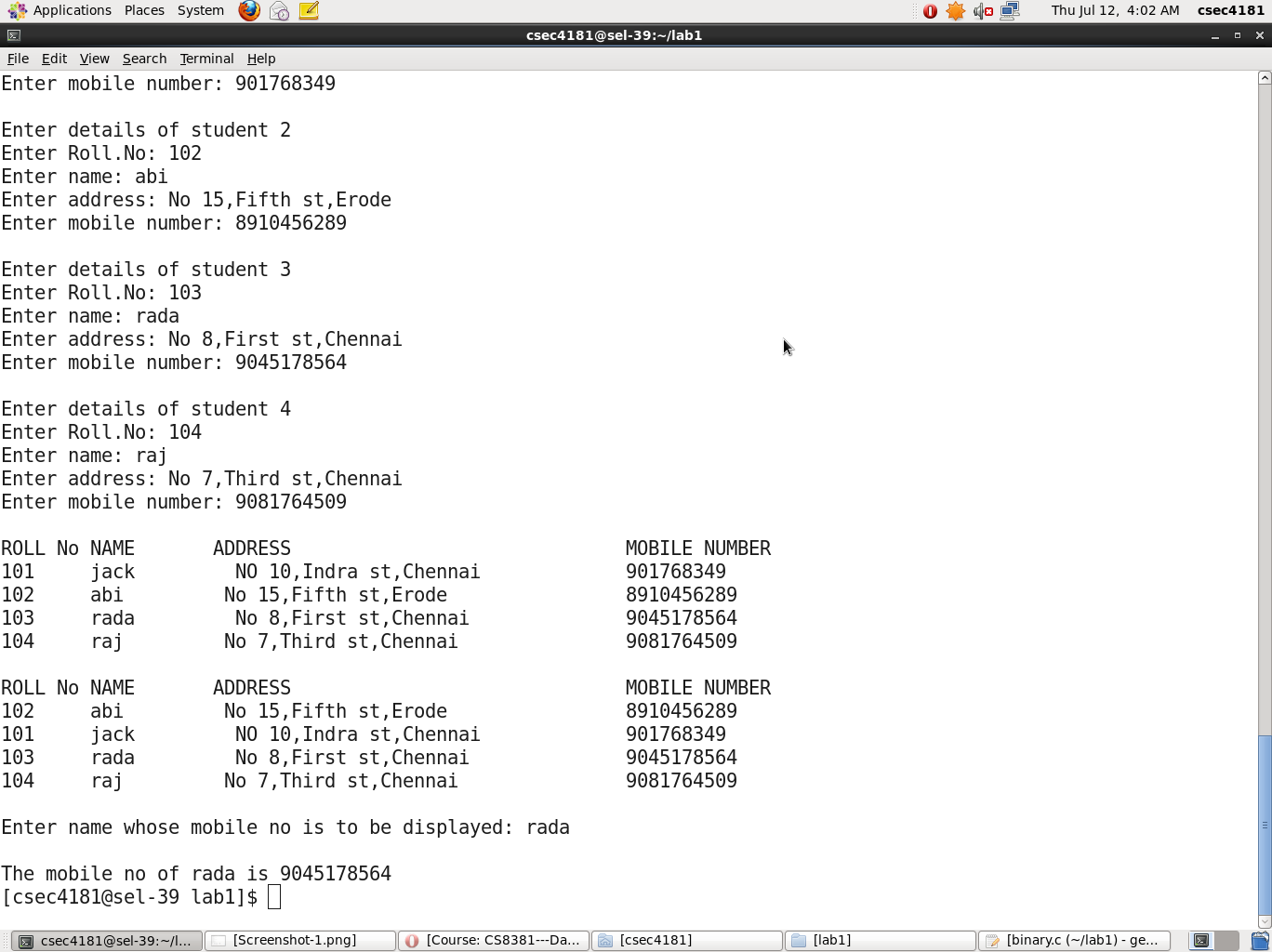
display(n,record);

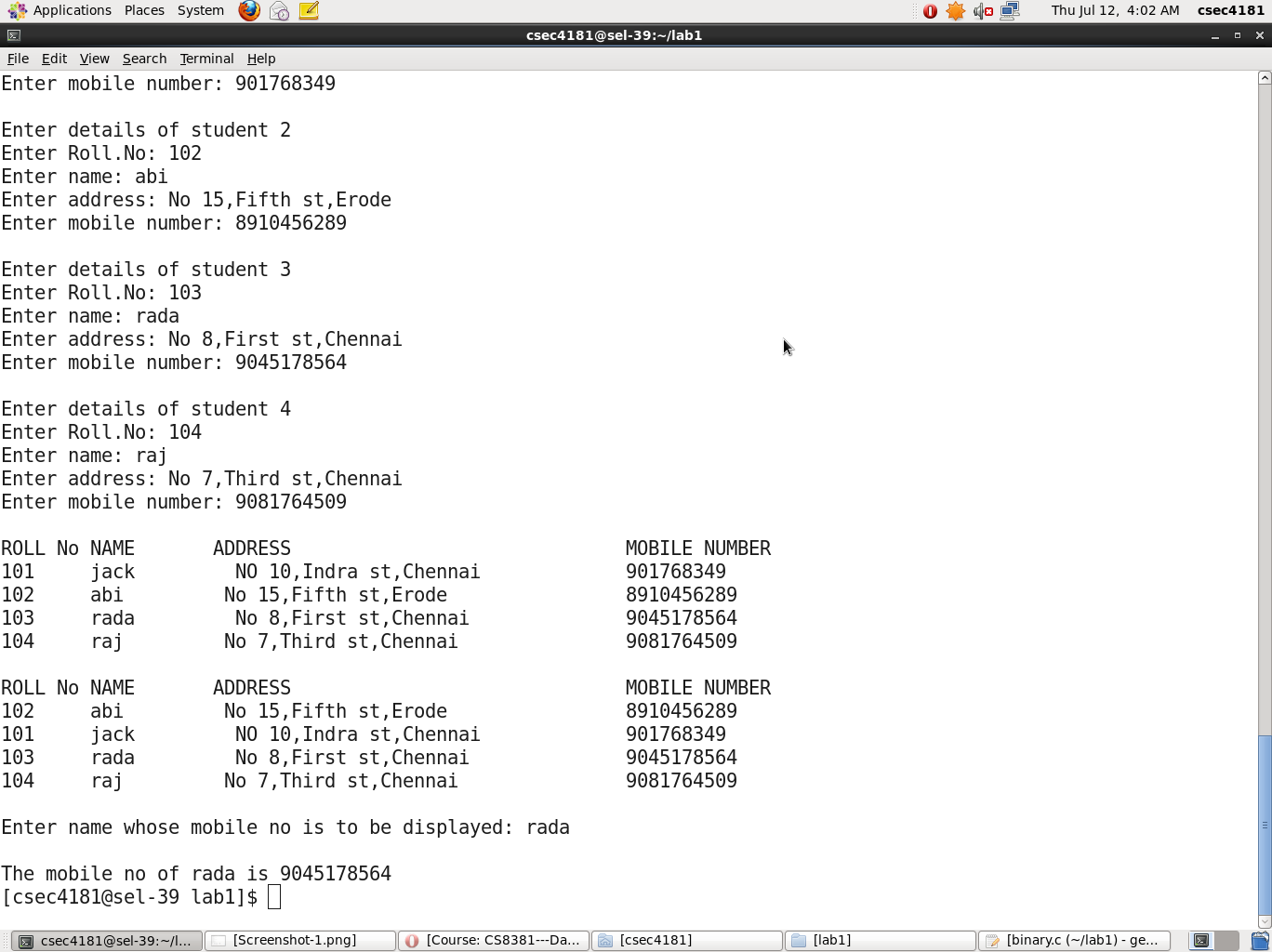
search(n,record);

}

**OUTPUT:**







**PROGRAM CODE: INSERTION SORT**

#include<stdio.h>

int\* input(int n)

{ int i;

static int a[30];

printf("Enter the values: ");

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

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

return a;

}

void display(int a[],int n)

{

int i;

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

printf("%d ",a[i]);

printf("\n");

}

int\* insertion(int a[],int n)

{

int i,j,key;

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

{

key=a[i];

j=i-1;

while((j>=0) && (a[j]>key))

{

a[j+1]=a[j];

j=j-1;

}

a[j+1]=key;

}

return a;

}

void main()

{

int n,\*arr;

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

scanf("%d",&n);

arr=input(n);

arr=insertion(arr,n);

printf("The sorted array is(using insertion sort): ");

display(arr,n);

}

**PROGRAM CODE: SHELL SORT**

#include<stdio.h>

int\* input(int n)

{ int i;

static int a[30];

printf("Enter the values: ");

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

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

return a;

}

void display(int a[],int n)

{

int i;

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

printf("%d ",a[i]);

printf("\n");

}

int\* shell(int a[],int n)

{

int incr,i,j,tmp;

for(incr=n/2;incr>0;incr/=2)

{

for(i=incr/2;i<n;i++)

{ tmp=a[i];

for(j=i;j>=incr;j-=incr)

{

if(tmp<a[j-incr])

a[j]=a[j-incr];

else

break;

}

a[j]=tmp;

}

}

return a;

}

void main()

{

int n,\*arr;

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

scanf("%d",&n);

arr=input(n);

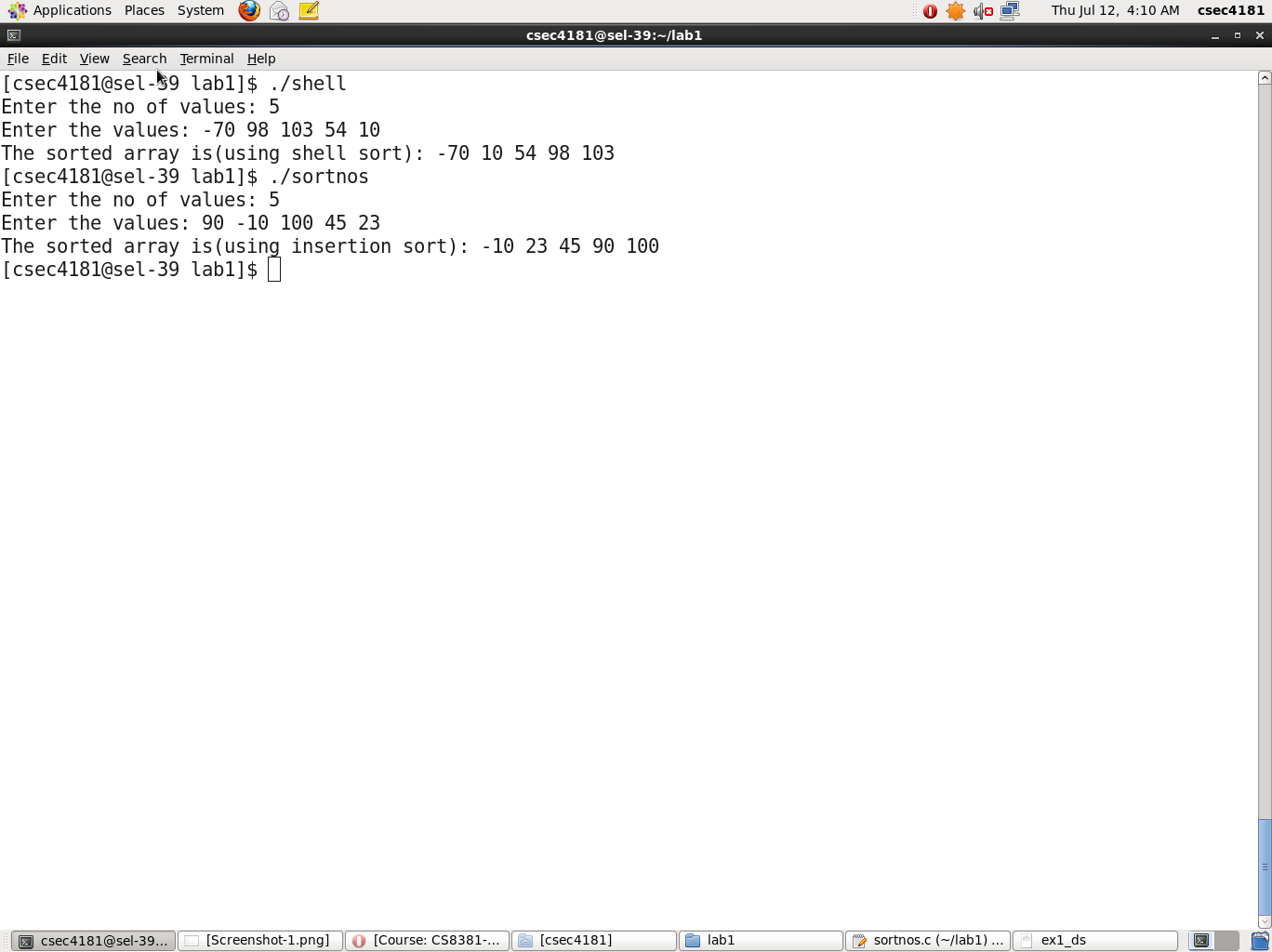
arr=shell(arr,n);

printf("The sorted array is(using shell sort): ");

display(arr,n);

}

**OUTPUT:**



**PROGRAM CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include "listadt.h"

void main()

{

int n,c,ch;

LIST l;

char k[50];

printf ("\nEnter the maximum number of books: ");

scanf ("%d",&n);

l=Createn(n);

printf("\nEnter the no of books to be added: ");

scanf ("%d",&n);

l=Input(l,n);

do{

printf("\n\n1.Insert a new book after a given book\n2.Delete a particular book\n3.Search a given book by name\n");

printf("4.Count the number of books under a particular genre\n5.Sort the books by author name and display\n");

printf("6. Display list of books\n7.Exit\nEnter your choice: ");

scanf ("%d",&ch);

switch(ch)

{

case 1:

Insert(l);

break;

case 2:

printf ("\nEnter book name to be deleted: ");

gets(k);

Delete (k,l);

break;

case 3:

printf ("\nEnter book name to be searched: ");

gets(k);

Search(k,l);

break;

case 4:

printf ("\nEnter genre to be counted: ");

gets(k);

c=Count (k,l);

printf ("\nNumber of records under the given genre %s : %d",k,c);

break;

case 5:

Sort(l);

break;

case 6:

Display(l);

break;

case 7:

break;

default:

printf ("\nInvalid choice");

}

printf("\nPress 1 to go back to menu\nPress 2 to exit\n");

scanf("%d",&ch);

}while(ch==1);

}

**listadt header file:**

typedef struct list \*ptrToNode;

typedef ptrToNode LIST;

typedef int position;

typedef struct

{

long id;

char bname[40];

char genre[25];

char author[40];

long yr;

}book;

struct list

{

int capacity;

int size;

book \*arr;

};

LIST Createn(int max)

{

LIST l;

l=(struct list\*)malloc(sizeof(struct list));

if(l== NULL)

printf("\nError");

else

{

l->capacity=max;

l->arr=(book\*)malloc(sizeof(book)\*max);

if(l->arr==NULL)

printf("\nList creation error");

else

{

l->size=0;

printf("\nList is Created successfully\n");

}

}

return l;

}

LIST Input(LIST l,int n)

{

int i;

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

{

printf("Enter the details for book-%d: \n",i+1);

printf("Enter the ID: ");

scanf("%ld",&(l->arr[i].id));

getchar();

printf("Enter the name of the book: ");

gets(l->arr[i].bname);

printf("Enter the genre: ");

gets(l->arr[i].genre);

printf("Enter the author: ");

gets(l->arr[i].author);

printf("Enter the year of publication: ");

scanf("%ld",&(l->arr[i].yr));

}

l->size=n;

return l;

}

void Display(LIST l)

{

int i;

printf("\nThe details of the books are:");

printf("\n ID TITLE GENRE AUTHOR YOP\n");

for(i=0;i<l->size;i++)

{

printf("%ld ",l->arr[i].id);

printf("%s\t\t",l->arr[i].bname);

printf("%s\t",l->arr[i].genre);

printf("%s\t",l->arr[i].author);

printf("%ld\n\n",l->arr[i].yr);

}

}

position Findpos(char b[],LIST l)

{

position P;

P=0;

while(P!=l->size && (strcmp(l->arr[P].bname,b)!=0))

P++;

return P;

}

void Search(char x[],LIST l)

{

position p=Findpos(x,l);

if (p==l->size)

printf("Element not found");

else

{

printf("\nThe details of the books are:");

printf("\n ID TITLE GENRE AUTHOR YOP\n");

printf("%ld ",l->arr[p].id);

printf("%s\t\t",l->arr[p].bname);

printf("%s\t",l->arr[p].genre);

printf("%s\t",l->arr[p].author);

printf("%ld\n\n",l->arr[p].yr);

}

}

book InsertInput(LIST l)

{ book x;

printf("Enter the details for book to be inserted: \n");

printf("Enter the ID: ");

scanf("%ld",&x.id);

getchar();

printf("Enter the name of the book: ");

gets(x.bname);

printf("Enter the genre: ");

gets(x.genre);

printf("Enter the author: ");

gets(x.author);

printf("Enter the year of publication: ");

scanf("%ld",&x.yr);

return x;

}

void Insert(LIST l)

{

int i;

char b[40]; //b is the name of the book after which a book is to be inserted

position P;

if(l->size==l->capacity)

printf("\n List is Full");

else

{

book x=InsertInput(l); //x is a structure variable

getchar();

printf("Enter the name of the book after which the given book is to be inserted: ");

//gets(b);

scanf("%s",b);

P=Findpos(b,l);

for(i=l->size;i>P+1;i--) //book needs to be inserted at p+1 position

l->arr[i]=l->arr[i-1];

l->size++;

l->arr[P+1]=x; //will insert at position p+1

printf("\nInserted successfully");

printf("\nThe list after inserting a book:\n");

Display(l);

}

}

void Delete(char b[],LIST l)

{

int i;

position P;

P=Findpos(b,l);

if(P==l->size)

printf("\n Book not found in the list");

else

{

for(i=P;i<l->size;i++)

l->arr[i]=l->arr[i+1];

l->size--;

printf("Deleted successfully");

printf("\nThe list after deleting a book:\n");

Display(l);

}

}

int Count(char g[],LIST l)

{

int i,cnt=0;

for(i=0;i<l->size;i++)

if(strcmp(l->arr[i].genre,g)==0)

cnt++;

return cnt;

}

void Sort(LIST l)

{

int i,j,minidx;

book temp;

for(i=0;i<l->size;i++)

{

minidx=i;

for(j=i+1;j<l->size;j++)

if (strcmp(l->arr[minidx].author,l->arr[j].author)>0)

minidx=j;

temp=l->arr[i];

l->arr[i]=l->arr[minidx];

l->arr[minidx]=temp;

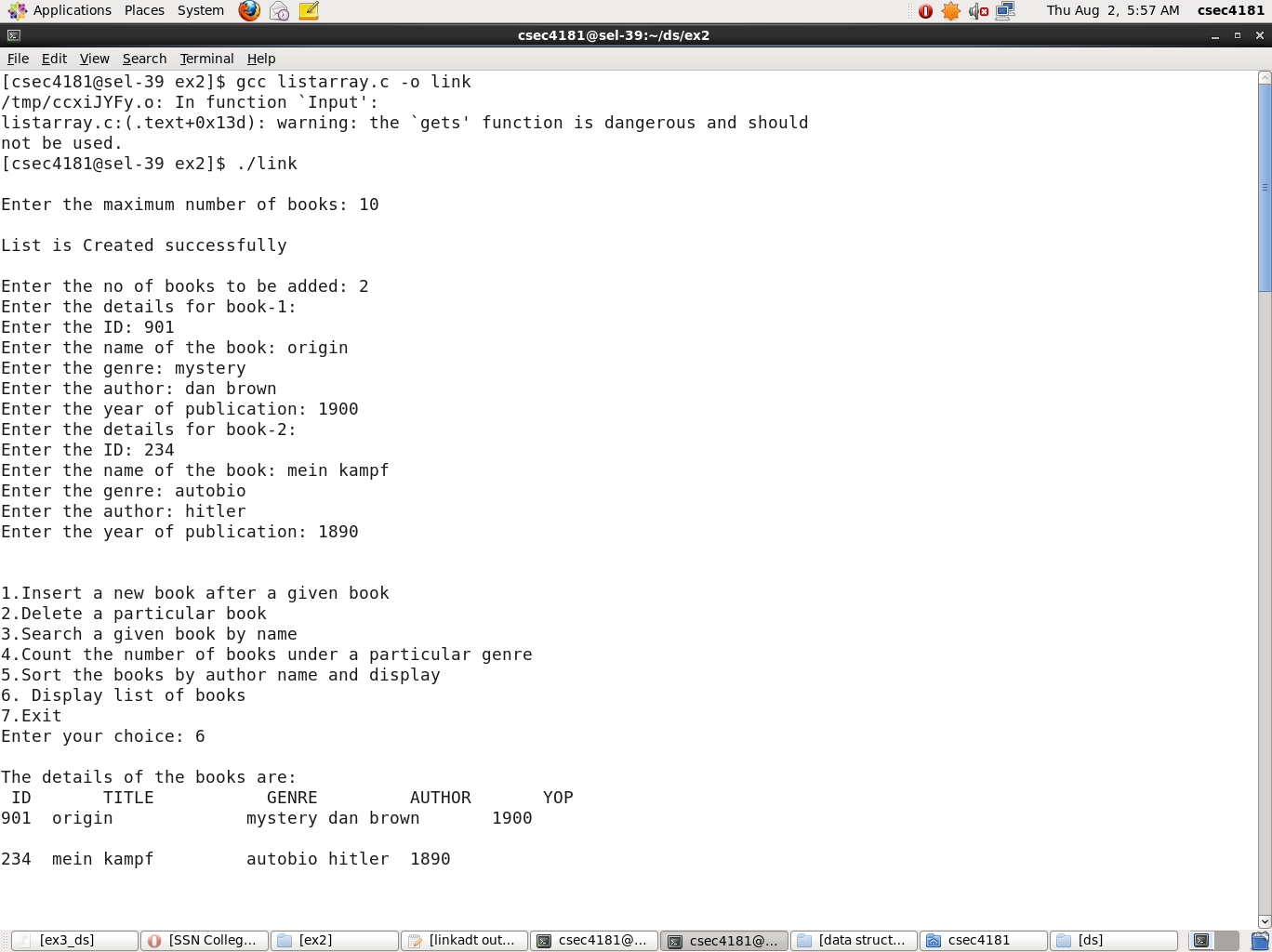
}

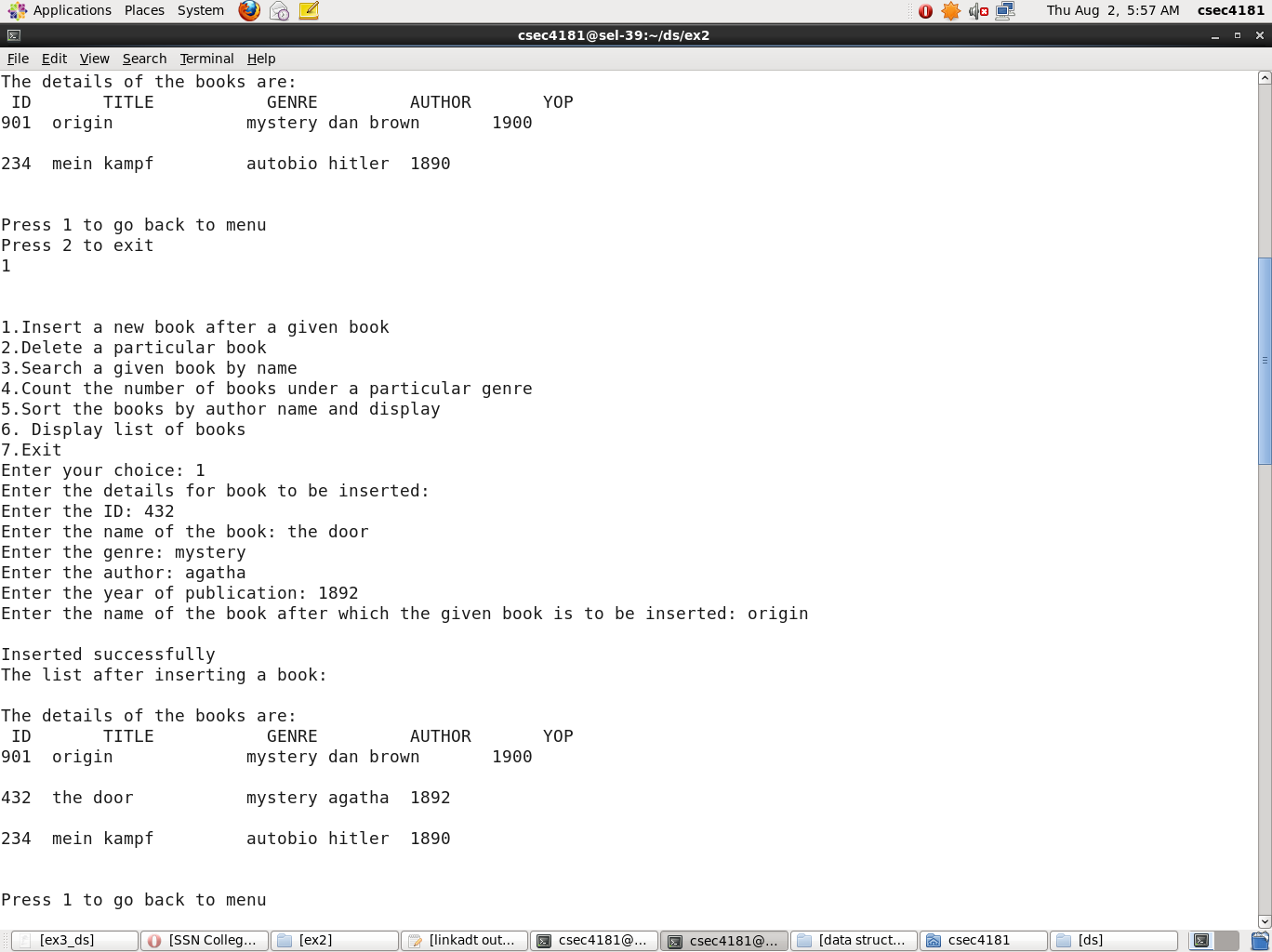
printf("The sorted list based on author name: \n");

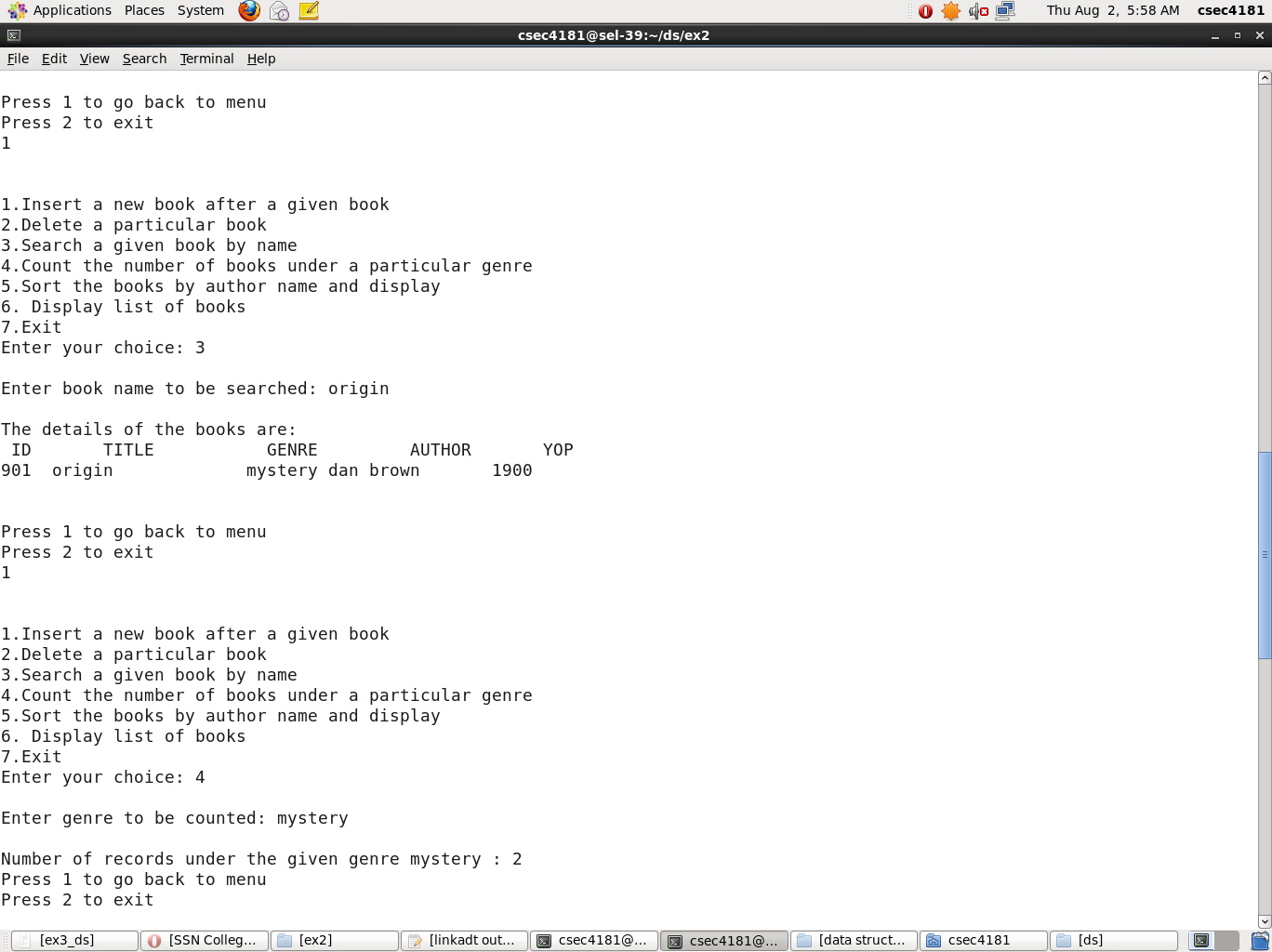
Display(l);

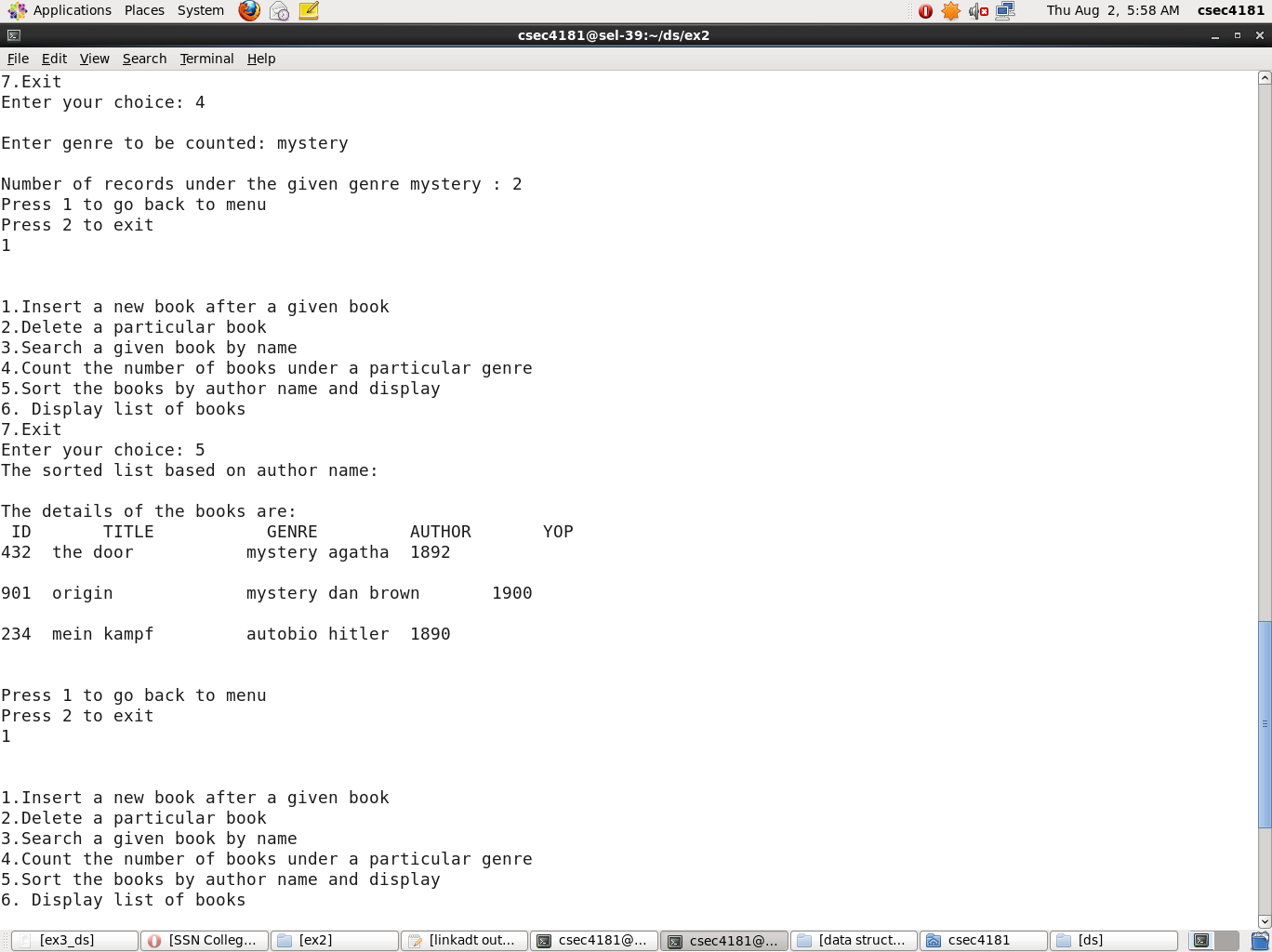
}

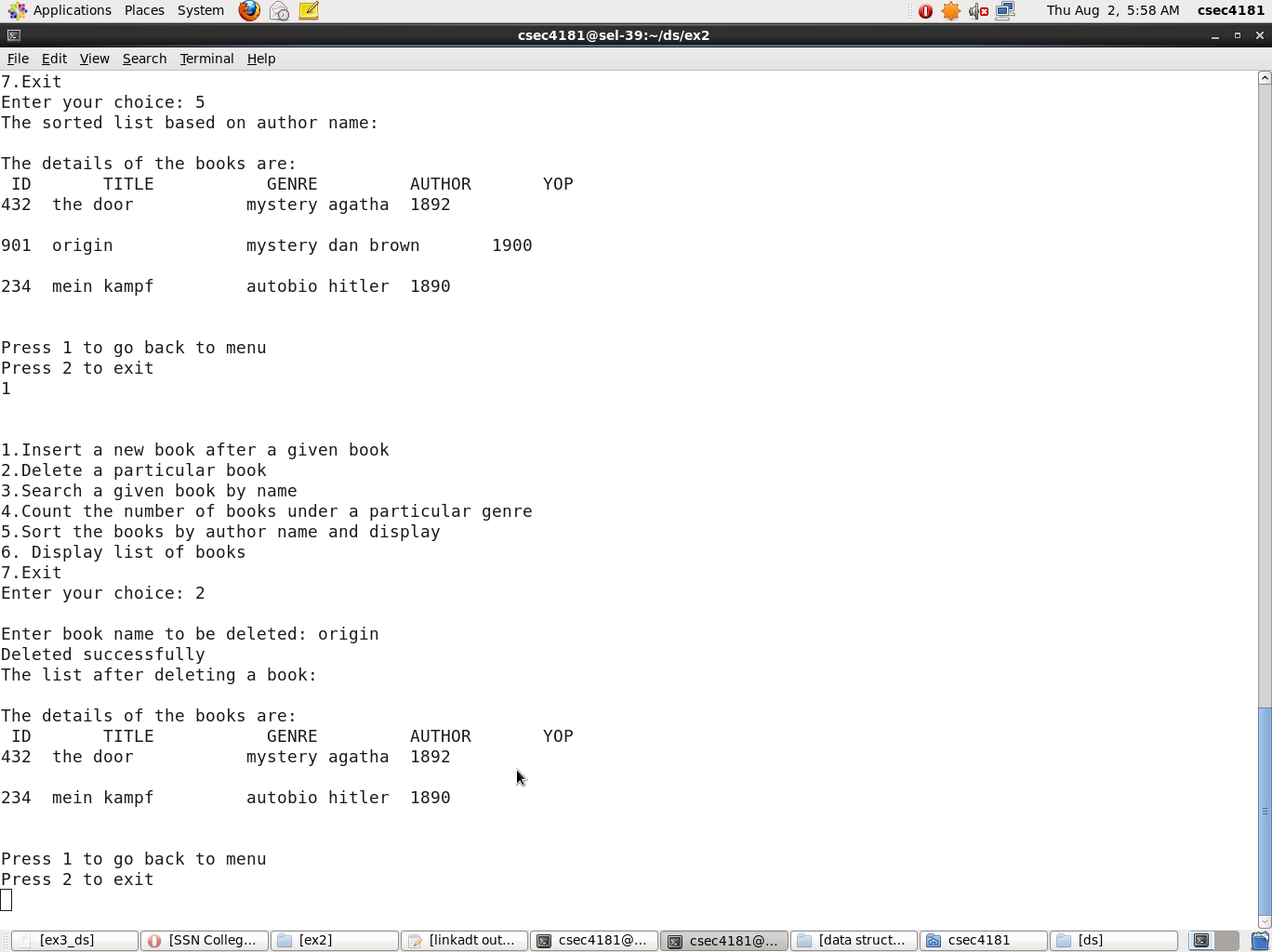
**OUTPUT:**











**PROGRAM CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include "linkadt.h"

void main()

{

int n,s,pos,c,ch;

list l;

char k[50];

printf ("\nEnter number of books to be added to the list: ");

scanf ("%d",&n);

l=Createlist(l,n);

do{

printf("\n\n1.Insert a new book after a given book\n2.Delete a particular book\n3.Search a given book by name\n");

printf("4.Count the number of books under a particular genre\n5.Sort the books by author name and display\n");

printf("6. Display list of books\n7.Exit\nEnter your choice: ");

scanf ("%d",&ch);

switch(ch)

{

case 1:

Insert(l);

break;

case 2:

printf ("\nEnter book name to be deleted: ");

gets(k);

Delete (k,l);

break;

case 3:

printf ("\nEnter book name to be searched: ");

gets(k);

Search(k,l);

break;

case 4:

printf ("\nEnter genre to be counted: ");

gets(k);

c=Count (k,l);

printf ("\nNumber of records under the given genre: %d",c);

break;

case 5:

printf ("\nSorted records: ");

Sort(l);

break;

case 6:

Display(l);

break;

case 7:

break;

default:

printf ("\nInvalid choice");

}

printf("\nPress 1 to go back to menu\nPress 2 to exit\n");

scanf("%d",&ch);

}while(ch==1);

}

**linkadt header file:**

typedef struct node \*ptrtonode;

typedef ptrtonode list;

typedef ptrtonode position;

typedef struct

{

long id;

char bname[40];

char genre[25];

char author[40];

long yr;

}book;

struct node

{

book b;

position next;

};

list Createhead()

{

ptrtonode head;

head=(list)malloc(sizeof(struct node));

if(head==NULL)

printf("\nList creation error");

else

{

head->next=NULL;

printf("\nList created successfully\n");

}

return head;

}

list Createlist(list l,int num)

{

ptrtonode new\_node,ptr;

int i;

l=Createhead();

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

{

new\_node = (list)malloc(sizeof(struct node));

printf("Enter the details for book-%d: \n",i+1);

printf("Enter the ID: ");

scanf("%ld",&new\_node->b.id);

getchar();

printf("Enter the name of the book: ");

gets(new\_node -> b.bname);

printf("Enter the genre: ");

gets(new\_node -> b.genre);

printf("Enter the author: ");

gets(new\_node -> b.author);

printf("Enter the year of publication: ");

scanf("%ld",&new\_node -> b.yr);

new\_node -> next = NULL;

if(l->next == NULL)

l->next = new\_node;

else

{ ptr = l->next;

while(ptr -> next != NULL)

ptr = ptr -> next;

ptr -> next = new\_node;

}

}

return l;

}

void Display(list l)

{

list ptr;

ptr=l->next;

printf("\nThe details of the books are:");

printf("\n ID TITLE GENRE AUTHOR YOP\n");

while(ptr != NULL)

{

printf("%ld ",ptr->b.id);

printf("%s\t\t",ptr->b.bname);

printf("%s\t",ptr->b.genre);

printf("%s\t",ptr->b.author);

printf("%ld\n\n",ptr->b.yr);

ptr = ptr -> next;

}

}

int islast(position p, list l)

{

return p->next == NULL;

}

position Findpos(char x[],list l)

{

position p=l->next;

while(p!=NULL && strcmp(p->b.bname,x)!=0)

p=p->next;

return p;

}

void Search(char x[],list l)

{

position ptr=Findpos(x,l);

if (ptr==NULL)

printf("Element not found");

else

{

printf("\nThe details of the books are:");

printf("\n ID TITLE GENRE AUTHOR YOP\n");

printf("%ld ",ptr->b.id);

printf("%s\t\t",ptr->b.bname);

printf("%s\t",ptr->b.genre);

printf("%s\t",ptr->b.author);

printf("%ld\n\n",ptr->b.yr);

}

}

book InsertInput(list l)

{ book x;

printf("Enter the details for book to be inserted: \n");

printf("Enter the ID: ");

scanf("%ld",&x.id);

getchar();

printf("Enter the name of the book: ");

gets(x.bname);

printf("Enter the genre: ");

gets(x.genre);

printf("Enter the author: ");

gets(x.author);

printf("Enter the year of publication: ");

scanf("%ld",&x.yr);

return x;

}

void Insert(list l)

{

int i;

char b[40];

position p;

position tmpcell = (list)malloc(sizeof(struct node));

if(tmpcell == NULL)

printf("Insertion error");

tmpcell->b=InsertInput(l); //x is a structure variable

getchar();

printf("Enter the name of the book after which the given book is to be inserted: ");

gets(b);

p=Findpos(b,l);

tmpcell->next = p->next;

p->next = tmpcell; //will insert at position p+1

printf("Inserted successfully");

printf("\nThe list after inserting a book:\n");

Display(l);

}

position Findprev(char x[], list l) //retuerns ptr to last element if element not found

{

position p=l;

while(p->next!=NULL && (strcmp(p->next->b.bname,x)!=0))

p=p->next;

return p;

}

void Delete(char x[],list l)

{

list tmpcell;

position p;

p=Findprev(x,l);

if(islast(p,l)) //element present

printf("\n Element not found in the list");

else

{

tmpcell = p->next;

p->next = tmpcell->next;

free(tmpcell);

printf("Deleted successfully");

printf("\nThe list after deleting a book:\n");

Display(l);

}

}

int Count(char g[],list l)

{

int cnt=0;

list ptr=l->next;

while(ptr != NULL)

{ if(strcmp(ptr->b.genre,g)==0)

cnt++;

ptr=ptr->next;

}

return cnt;

}

void Sort(list l)

{

ptrtonode ptr1, ptr2;

book temp;

ptr1=l->next;

while(ptr1 -> next != NULL)

{

ptr2 = ptr1 -> next;

while(ptr2 != NULL)

{

if(strcmp(ptr1 -> b.author,ptr2 -> b.author)>0)

{

temp = ptr1 -> b;

ptr1 -> b = ptr2 -> b;

ptr2 -> b = temp;

}

ptr2 = ptr2 -> next;

}

ptr1 = ptr1 -> next;

}

printf("The sorted list based on author name: \n");

Display(l);

**OUTPUT:**

