**Ex: 13 Dt:**

**Aim:**

Create a class that stores the details about a room in a hotel (private: room no, type, cost). Create subclasses like Lounge (no.of people it can accommodate, A/C type(centralized/window), food preference, recreational facilities (as a string array)) and deluxe room(A/C or non A/C, single/double bedded). Create a class that maintains the customer details (name, age, address, phoneno). Allow booking of the room by the customer after checking the status of its availability. Overload the booking function such that it can book either a lounge or deluxe room for a customer.

**Program:**

#include<iostream>

#include<conio.h>

using namespace std;

class base

{ private:

int room\_no,cost;

char typ[20];

public:

void inp(int x,char y[20])

{ cost=x; strcpy(typ,y);

cout<<"ROOM\_NO: "; cin>>room\_no;

cout<<"TYPE: "<<typ;

cout<<"\nCOST: "<<cost;

}

};

class room1:public base

{ private:

int ppl,a,c1,c2;

char a\_ctp1[20],a\_ctp2[20],food\_pr[20],recre\_fac[20];

public:

room1()

{ ppl=10; strcpy(a\_ctp1,"CENTRALIZE"); strcpy(a\_ctp2,"WINDOW");

strcpy(food\_pr,"1.a\n2.b\n3.c\n4.d"); strcpy(recre\_fac,"1.x\n2.y\n3.z"); a=10; c1=3000; c2=2500;

}

int inp1();

};

class room2:public base

{ private:

int a,c1,c2,c3,c4;

char a\_tp1[10],a\_tp2[10],bd\_tp1[20],bd\_tp2[20];

public:

room2()

{ strcpy(a\_tp1,"AC"); strcpy(a\_tp2,"NON\_AC"); a=10;

strcpy(bd\_tp1,"SINGLE\_BED"); strcpy(bd\_tp2,"DOUBLE\_BED");

c1=2000; c2=3000; c3=1000; c4=2000;

}

int inp2();

};

class customer

{ private:

char name[20],adrs[20],p\_no[11];

int age;

public:

void inp()

{cout<<"\nNAME: "; cin>>name;

cout<<"AGE: "; cin>>age;

cout<<"ADDRESS: "; cin>>adrs;

cout<<"PHONE\_NO: "; cin>>p\_no;

}

};

int room1::inp1()

{ int n;

cout<<"ROOM\_TYPES";

cout<<"\n1."<<a\_ctp1<<"\nPRICE:"<<c1;

cout<<"\n2."<<a\_ctp2<<"\nPRICE:"<<c2;

cin>>n;

if(n==1 && a>=1)

{ inp(c1,a\_ctp1); --a; return(1); }

else if(n==2 && a>=1)

{ inp(c2,a\_ctp2); --a; return(2); }

else

{ return(0); }

}

int room2::inp2()

{ int n;

cout<<"ROOM\_TYPES";

cout<<"\n1."<<a\_tp1<<"\n"<<bd\_tp1<<"\nPRICE:"<<c1;

cout<<"\n2."<<a\_tp1<<"\n"<<bd\_tp2<<"\nPRICE:"<<c2;

cout<<"\n3."<<a\_tp2<<"\n"<<bd\_tp1<<"\nPRICE:"<<c3;

cout<<"\n4."<<a\_tp2<<"\n"<<bd\_tp2<<"\nPRICE:"<<c4;

cin>>n;

if(a>=1)

{ switch(n)

{ case 1: inp(c1,a\_tp1); --a; return(1);

break;

case 2: inp(c2,a\_tp1); --a; return(2);

break;

case 3: inp(c3,a\_tp2); --a; return(3);

break;

case 4: inp(c4,a\_tp2); --a; return(4);

break;

default: return(0);

}

}

else

{ return(0); }

}

int main()

{ room1 r1; room2 r2; customer c[10];

int n,x,i=0;

while(1)

{ system("CLS");

cout<<"\n1.LOUNGE\_ROOM\n2.DELUX\_ROOM\n3.EXIT"; cin>>n; system("CLS");

if(n==1)

{ x=r1.inp1();

if(x==0)

{ cout<<"NOT AVAILABLE!!"; }

else

{ c[i].inp(); i++; }

}

else if(n==2)

{ x=r2.inp2();

if(x==0)

{ cout<<"NOT AVAILABLE!!"; }

else

{ c[i].inp(); i++; }

}

else

{ exit(0); }

}

getch();

return(0);

}

**Sample Input-Output:**

**LOUNGE\_ROOM**

**DELUX\_ROOM**

**EXIT 1**

**ROOM\_TYPES**

**CENTRALIZE**

**PRICE: 3000**

**WINDOW**

**PRICE: 2500 1**

**ROOM\_NO: 101**

**TYPE: CENTRALIZE**

**COST: 3000**

**NAME: ABC**

**AGE:12**

**ADDRESS: ABC**

**PHONE\_NO: 1234**

**LOUNGE\_ROOM**

**DELUX\_ROOM**

**EXIT 1**

**ROOM\_TYPES**

**AC**

**SINGLE\_BED**

**PRICE: 2000**

**AC**

**DOUBLE\_BED**

**PRICE:3000**

**NON\_AC**

**SINGLE\_BED**

**PRICE: 1000**

**4. NON\_AC**

**DOUBLE\_BED**

**PRICE:2000 4**

**ROOM\_NO: 102**

**TYPE: NON\_AC**

**COST: 2000**

**NAME: XYZ**

**AGE:15**

**ADDRESS: XYZ**

**PHONE\_NO: 1234**

**LOUNGE\_ROOM**

**DELUX\_ROOM**

**EXIT 5**

**Ex: 14 Dt:**

**Aim:**

Create a c++ program that Create a base class person and derived classes general public, government official and VVIP. Maintain a static array of objects using inside the persons class. Access the shared memory from all the classes to store and retrieve data. Reserve zones in that array such that they are to be used by appropriate category of people. Ensure that the space is available in each zone before you create a dynamic object of each type and store it inside the shared array.

**Program:**

#include<iostream>

#include<stdlib.h>

#include<conio.h>

#include<string.h>

using namespace std;

class person

{ char name[20];

int age;

public:

static person \*p[10];

void inp()

{ cout<<"Name : "; cin>>name;

cout<<"Age : "; cin>>age;

}

void display()

{ cout<<"\nName: "<<name;

cout<<"\nAge: "<<age;

}

};

person \*person::p[]={NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL};

class vvip :public person

{ char designation[20];

public:

void vinp()

{ cout<<"Designation: "; cin>>designation; }

void vdisplay()

{ cout<<"\nDesignation: "<<designation; }

};

class general:public person

{ char job[20];

int wexp;

public:

void ginp()

{ cout<<"Job: "; cin>>job;

cout<<"Working\_Expirence: "; cin>>wexp;

}

void gdisplay()

{ cout<<"\nJob: "<<job;

cout<<"\nWorking\_Expirence: "<<wexp;

}

};

class official:public person

{ char department[20];

char position[20];

int wyears;

public:

void oinp()

{ cout<<"Department : "; cin>>department;

cout<<"Position : "; cin>>position;

cout<<"Working\_Years : "; cin>>wyears;

}

void odisplay()

{ cout<<"\nDepartment : "<<department;

cout<<"\nPosition : "<<position;

cout<<"\nWorking\_Years : "<<wyears;

}

};

int main()

{ vvip \*v[3];

general \*g[3];

official \*o[4];

int n,nv=0,ng=3,no=6,i;

while(1)

{ l: system("CLS");

cout<<"1.INSERT\n2.DISPLAY\n3.Exit"; cin>>n;

switch(n)

{case 1: l1: system("CLS");

cout<<"1.VVIP\n2.General\n3.Officials\n4.EXIT"; cin>>n;

switch(n)

{ case 1: if(nv<3)

{ v[nv]=new vvip;

person::p[nv]=v[nv];

person::p[nv]->inp();

v[nv]->vinp();

nv++;

}

else

{ cout<<"\nNot available !!!"; }

getch();

goto l1;

case 2: if(ng<6)

{ g[ng-3]=new general;

person::p[ng]=g[ng-3];

person::p[ng]->inp();

g[ng-3]->ginp();

ng++;

}

else

{ cout<<"\nNot available !!!"; }

getch();

goto l1;

case 3: if(no<10)

{ o[no-6]=new official;

person::p[no]=o[no-6];

person::p[no]->inp();

o[no-6]->oinp();

no++;

}

else

{ cout<<"\nNot available !!!"; }

getch();

goto l1;

case 4: goto l;

default: cout<<"\nWrong Choice !!!";

}break;

case 2: l2: system("CLS");

cout<<"1.VVIP\n2.General\n3.Officials\n4.EXIT"; cin>>n;

switch(n)

{ case 1: if(nv!=0)

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

{ cout<<"\nPerson "<<i+1;

person::p[i]->display();

v[i]->vdisplay();

}

}

else

{ cout<<"Not available !!!"; }

getch();

goto l2;

case 2: if(ng!=3)

{ for(i=3;i<ng;i++)

{ cout<<"\nPerson "<<i-2;

person::p[i]->display();

g[i-3]->gdisplay();

}

}

else

{ cout<<"Not available !!!"; }

getch();

goto l2;

case 3: if(no!=6)

{ for(i=6;i<no;i++)

{ cout<<"\nPerson "<<i-5;

person::p[i]->display();

o[i-6]->odisplay();

}

}

else

{ cout<<"Not available !!!"; }

getch();

goto l2;

case 4: goto l;

default: cout<<"Wrong Choice !!!";

}break;

case 3: exit(0);

default: cout<<"Wrong Choice !!!";

}

getch();

}

}

**Sample Input-Output:**

**1.INSERT**

**2.DISPLAY**

**3.Exit**

**1.VVIP**

**2.GENERAL**

**3.OFFICIAL**

**4.EXIT 1.**

**Name : abc Age : 12**

**Address : abc Designation : abc**

**1.VVIP**

**2.GENERAL**

**3.OFFICIAL**

**4.EXIT 1.**

**Name : xyz Age : 12**

**Address : abc Designation : abc**

**1.VVIP**

**2.GENERAL**

**3.OFFICIAL**

**4.EXIT 1.**

**Name : esc Age : 12**

**Address : abc Designation : abc**

**1.VVIP**

**2.GENERAL**

**3.OFFICIAL**

**4.EXIT 1.**

**Not available !!!**

**1.VVIP**

**2.GENERAL**

**3.OFFICIAL**

**4.EXIT 4.**

**1.INSERT**

**2.DISPLAY**

**3.Exit 2.**

**1.VVIP**

**2.GENERAL**

**3.OFFICIAL**

**4.EXIT 1.**

**Person 1**

**Name : abc Age : 12**

**Address : abc Designation : abc**

**Person 2**

**Name : xyz Age : 12**

**Address : abc Designation : abc**

**Person 3**

**Name : esc Age : 12**

**Address : abc Designation : abc**

**1.VVIP**

**2.GENERAL**

**3.OFFICIAL**

**4.EXIT 1.**

**1.INSERT**

**2.DISPLAY**

**3.Exit 3.**

**Ex: 15 Dt:**

**Aim:**

Create a class employee that maintains the details like empid, years of exp, address, phone). Create separate class for type of skills the employee possesses like carpentry (what type of wood, years, type of creations, tools used), masonry (type of constructions, years), Driving (license no, years of exp, types of vehicles). Make the employee class to inherit the three classes and have the details that are relevant for the employee. Have constructors to initialize all the classes and maintain the status in employee according to the skills (111- means have all skills). If a skill status is 0, then have the default values initialized with constructor.

**Program:**

#include<iostream>

#include<conio.h>

using namespace std;

class carpentry

{ char typ1[20];

char wd[20];

char tool[20];

int y1;

public:

carpentry()

{ \*typ1=0;

\*wd=0;

\*tool=0;

y1=0;

}

carpentry(int s)

{ cout<<"TYPE\_OF\_WOOD: " ;

cin>>wd;

cout<<"TYPE\_OF\_CREATION: ";

cin>>typ1;

cout<<"TOOL\_USED: ";

cin>>tool;

cout<<"YEAR\_OF\_EXP: ";

cin>>y1;

}

};

class masonry

{ char typ2[20];

int y2;

public:

masonry()

{ \*typ2=0;

y2=0;

}

masonry(int s)

{ cout<<"TYPE\_OF\_CONSTRUCTION: ";

cin>>typ2;

cout<<"YEAR\_OF\_EXP: ";

cin>>y2;

}

};

class driving

{ char lic[20];

char typ3[20];

int y3;

public:

driving()

{ \*typ3=0;

\*lic=0;

y3=0;

}

driving(int s)

{ cout<<"LICENSE\_NO: ";

cin>>lic;

cout<<"TYPES\_OF\_VEHICLES: ";

cin>>typ3;

cout<<"YEAR\_OF\_EXP: ";

cin>>y3;

}

};

class employee:public carpentry,public masonry,public driving

{ char id[20];

int y0;

char address[20];

char phone[11];

int status[4];

public:

employee()

{ cout<<"ID: ";

cin>>id;

cout<<"YEAR\_OF\_EXP.:";

cin>>y0;

cout<<"ADDRESS: ";

cin>>address;

cout<<"PHONE\_NO: ";

cin>>phone;

cout<<"CARPENTRY\_SKILL\n1.YES\n2.NO";

cin>>status[1];

if(status[1]==2)

{ carpentry(); }

else

{ carpentry c(status[1]); }

cout<<"MASONRY\_SKILL\n1.YES\n2.NO";

cin>>status[2];

if(status[2]==2)

{ masonry(); }

else

{ masonry m(status[2]); }

cout<<"DRIVING\_SKILL\n1.YES\n2.NO";

cin>>status[3];

if(status[3]==2)

{ driving(); }

else

{ driving d(status[3]); }

}

void display()

{ cout<<"ID: "<<id<<endl;

cout<<"YEAR\_OF\_EXP.:"<<y0<<endl;

cout<<"ADDRESS: "<<address<<endl;

cout<<"PHONE\_NO: "<<phone<<endl;

cout<<"SKILLS\n";

if(status[1]==1)

{ cout<<"CARPENTRY\_SKILL"<<endl; }

if(status[2]==1)

{ cout<<"MASONRY\_SKILL"<<endl; }

if(status[3]==1)

{ cout<<"DRIVING\_SKILL"<<endl; }

if(status[1]!=1 && status[2]!=1 && status[3]!=1)

{ cout<<"SKILLS 000"<<endl; }

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

}

}\*e[3];

int main()

{ int i=0,ch;

while(1)

{ system("CLS");

cout<<"1.EMPLOYEE\_REG.\n2.DISPLAY\n3.EXIT";

cin>>ch;

switch(ch)

{ case 1: i++;

e[i]=new employee();

break;

case 2: for(int x=1;x<=i;x++)

{ e[x]->display(); }

break;

case 3: exit(0);

default: cout<<"INVALID\_ENTRY!!";

}getch();

}

}

**Sample Input-Output:**

**1. EMPLOYEE\_REG.**

**2. DISPLAY**

**3. EXIT 1**

**ID: 123@**

**YEAR\_OF\_EXP.: 7**

**ADDRESS: ABC**

**PHONE\_NO: 1234**

**CARPENTRY\_SKILL**

**1. YES**

**2. NO 1**

**TYPE\_OF\_WOOD: ABC,XYZ,ESC**

**TYPE\_OF\_CREATION: QWERTY,ASDF,QW**

**TOOL\_USED: LK,JHG,OP,DF**

**YEAR\_OF\_EXP.: 2**

**MASONRY\_SKILL**

**1. YES**

**2. NO 2**

**DRIVING\_SKILL**

**1. YES**

**2. NO 1**

**LICENSE\_NO: 1092EFR23**

**TYPE\_OF\_VEHICLES: AS,CD,EW,RT,RW,FWM**

**YEAR\_OF\_EXP.: 5**

**1. EMPLOYEE\_REG.**

**2. DISPLAY**

**3. EXIT 2**

**ID: 123@**

**YEAR\_OF\_EXP.: 7**

**ADDRESS: ABC**

**PHONE\_NO: 1234**

**SKILLS**

**CARPENTRY\_SKILL**

**DRIVING\_SKILL**

**1. EMPLOYEE\_REG.**

**2. DISPLAY**

**3. EXIT 3**

**Ex: 16 Dt:**

**Aim:**

Create a class template for list that can store names or numbers of students. Also perform the operations like insert, delete, update and display by either overloading template functions or using generic template functions.

**Program:**

#include<iostream>

#include<conio.h>

using namespace std;

template<class t>

class student

{ t \*p[10];

int i;

public:

student()

{ i=0; }

void insert(char \*a)

{ p[i]=new char[20];

strcpy(p[i],a);

i++;

cout<<"Successfully Inserted !!!";

}

void del(char \*a)

{ if(i==0)

{ cout<<"Empty"; }

else

{ int f=0,j;

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

{ if(strcmp(p[j],a)==0)

{ f=1; break; }

}

for(int k=j;k<i;k++) //deletion element by overwrite

{ p[k]=p[k+1]; }

if(f==0)

{ cout<<"Not Found !!!"; }

else

{ i--; cout<<"Successfully Deleted !!!"; }

}

}

void update(char \*a)

{ int f=0,j;

if(i==0)

{ cout<<"Empty List !!!"; }

else

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

{ if(strcmp(p[j],a)==0)

{ f=1; break; }

}

if(f==0)

{ cout<<"Not Found !!!"; }

else

{ char a[20];

cout<<"New Value : "; cin>>a;

strcpy(p[j],a);

cout<<"Successfully Updated !!!";

}

}

}

void display(char \*a)

{ int j;

if(i==0)

{ cout<<"Empty"; }

else

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

{ cout<<j+1<<"."<<p[j];

cout<<"\n------------------------------\n";

}

}

}

};

int main()

{ student <char> s;

char n[20];

int ch,i=0;

while(1)

{ system("CLS");

cout<<"1.INSERT\n2.DELETE\n3.UPDATE\n4.DISPLAY\n5.EXIT"; cin>>ch;

switch(ch)

{ case 1: cout<<"NAME: "; cin>>n;

s.insert(n);

break;

case 2: cout<<"NAME: "; cin>>n;

s.del(n);

break;

case 3: cout<<"NAME: "; cin>>n;

s.update(n);

break;

case 4: s.display(n);

break;

case 5: exit(0);

}getch();

}

}

**Sample Input-Output:**

1. **INSERT**
2. **DELETE**
3. **UPDATE**
4. **DISPLAY**
5. **EXIT 1**

**NAME: ABC**

**SUCCESSFULLY INSERTED!!**

1. **INSERT**
2. **DELETE**
3. **UPDATE**
4. **DISPLAY**
5. **EXIT 3**

**NAME: ABC**

**NEW NAME: YUVRAJ**

**SUCCESSFULLY INSERTED!!**

1. **INSERT**
2. **DELETE**
3. **UPDATE**
4. **DISPLAY**
5. **EXIT 2**

**NAME: YUVRAJ**

**SUCCESSFULLY DELETED!!**

1. **INSERT**
2. **DELETE**
3. **UPDATE**
4. **DISPLAY**
5. **EXIT 4**

**EMPTY**

1. **INSERT**
2. **DELETE**
3. **.UPDATE**
4. **DISPLAY**
5. **.EXIT 5**

**Ex: 17 Dt:**

**Aim:**

Create a class powered device (type(electronic/mechanical), power range(10W- 20W), self\_powered/adapter) and inherit it in two classes scanner (no.of pages per second, resolution, size, cost, brand) and printer (brand, dpi, color/b/w, pages per second, laser/inkjet). Create another class named copier that inherits the features of printer and scanner along with additional features (storage capacity, stored copy printing speed). Ensure that the duplicate copies of powered\_device object don’t exist in copier.

**Program:**

#include<iostream>

#include<conio.h>

using namespace std;

class powered\_device

{ public:

char type1[20]; //(electronic/mechanical)

int power\_range; //(10W- 20W)

char type2[20]; //(self\_powered/adapter)

};

class scanner:virtual public powered\_device

{ public:

int typ1; //no.of pages per second

char resolution[10];

int size,cost;

char brand1[20];

};

class printer:virtual public powered\_device

{ public:

int typ2; //no.of pages per second

char typ3[20]; //laser/inkjet

char dpi[10]; //output\_resolution(dotes\_per\_inch)

char color[10];

char brand2[20];

};

class copier:public printer,public scanner

{ int storage; //capacity

int speed; //stored copy printing speed

public:

void inp()

{ cout<<"POWERED\_DEVICE\n";

cout<<"1.ELECTRONIC\t2.MECHANICAL\n"; cin>>type1;

cout<<"POWER\_RANGE: "; cin>>power\_range;

cout<<"1.SELF\_POWERED\t2.ADAPTOR\n"; cin>>type2;

cout<<"SCANNER\n";

cout<<"no.of pages per second: "; cin>>typ1;

cout<<"resolution: "; cin>>resolution;

cout<<"size: "; cin>>size;

cout<<"cost: "; cin>>cost;

cout<<"brand: "; cin>>brand1;

cout<<"PRINTER\n";

cout<<"no.of pages per second: "; cin>>typ2;

cout<<"1.laser\t2.inkjet\n"; cin>>typ3;

cout<<"output\_resolution(dotes\_per\_inch): "; cin>>dpi;

cout<<"COLOR: "; cin>>color;

cout<<"BRAND: "; cin>>brand2;

}

void display()

{cout<<"POWERED\_DEVICE\n";

cout<<"1.ELECTRONIC\t2.MECHANICAL\n"<<type1<<endl;

cout<<"POWER\_RANGE: "<<power\_range<<endl;

cout<<"1.SELF\_POWERED\t2.ADAPTOR\n"<<type2<<endl;

cout<<"SCANNER\n";

cout<<"no.of pages per second: "<<typ1<<endl;

cout<<"resolution: "<<resolution<<endl;

cout<<"size: "<<size<<endl;

cout<<"cost: "<<cost<<endl;

cout<<"brand: "<<brand1<<endl;

cout<<"PRINTER\n";

cout<<"no.of pages per second: "<<typ2<<endl;

cout<<"1.laser\t2.inkjet\n"<<typ3<<endl;

cout<<"output\_resolution(dotes\_per\_inch): "<<dpi<<endl;

cout<<"COLOR: "<<color<<endl;

cout<<"BRAND: "<<brand2<<endl;

}

};

int main()

{ copier c[5];

int ch,i=-1,j;

while(1)

{ system("CLS");

cout<<"1.CREATE\_OBJECT\n2.DISPLAY\n3.EXIT"; cin>>ch;

switch(ch)

{ case 1: if(i<5)

{ i++; c[i].inp(); }

else

{ cout<<"NO\_SPACE"; } break;

case 2: if(i==-1)

{ cout<<"NO\_DATA\_IS\_THERE!!"; }

else

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

{ c[i].display(); } }break;

case 3: exit(0);

default: cout<<"INVALID\_ENTRY";

}getch();

}}

**Sample Input-Output:**

1. **CREATE\_OBJECT**
2. **DISPLAY**
3. **EXIT 1**

**POWERED\_DEVICE**

**1.ELECTRONIC 2.MECHANICAL**

**ELECTRONIC**

**POWER\_RANGE: 20**

**1.SELF\_POWERED 2.ADAPTOR**

**ADAPTOR**

**SCANNER**

**no of pages per second: 4**

**RESOLUTION: 20\*40**

**SIZE: 20**

**COST: 1000**

**BRAND: ABC**

**PRINTER**

**no\_of\_pages\_per\_second: 5**

**1.LASER 2.INKJET**

**LASER**

**OUTPUT\_RESOLUTION(DOTES\_PER\_INCH): 20\*10**

**COLOR: BLACK**

**BRAND: ESC**

1. **CREATE\_OBJECT**
2. **DISPLAY**
3. **EXIT 2**

**POWERED\_DEVICE**

**ELECTRONIC POWER\_RANGE: 20 ADAPTOR**

**SCANNER**

**no of pages per second: 4 RESOLUTION: 20\*40 SIZE: 20 COST: 1000**

**BRAND: ABC**

**PRINTER**

**no\_of\_pages\_per\_second: 5 LASER**

**OUTPUT\_RESOLUTION(DOTES\_PER\_INCH): 20\*10 COLOR: BLACK**

**BRAND: ESC**

1. **CREATE\_OBJECT**
2. **DISPLAY**
3. **EXIT 3**

**Ex: 18 Dt:**

**Aim:**

Create a class telephone\_directory (name, occupation, phone, address) and an virtual function search (), that searches the object contents in a linear fashion. Extend the class in a specific class called doctor (qualification, clinic, visiting time) and override the virtual function such that it searches using binary search. Also create another class lawyer (qualification, civil/criminal, cases attended, contact\_no, office\_address) that extends the telephone\_directory such that it wont override the virtual function. Search for a doctor and a lawyer by taking the option from the user.

**Program:**

#include<iostream>

#include<conio.h>

using namespace std;

class telephone\_directory //class telephone\_directory (name, occupation, phone, address)

{ public:

char name[20],occupation[20],phone[11],address[20];

static int x; static int y;

telephone\_directory()

{ cout<<"NAME: "; cin>>name;

cout<<"OCCUPATION: "; cin>>occupation;

cout<<"PHONE\_NO: "; cin>>phone;

cout<<"ADDRESS: "; cin>>address;

}

virtual void search()

{ cout<<"NOT\_FOUND!!"; }

};

//Extend the class in a specific class doctor (qualification, clinic, visiting time)

class doctor:public telephone\_directory

{ char qualification[20],clinic[20],visiting\_time[10];

public:

doctor()

{ cout<<"qualification: "; cin>>qualification;

cout<<"clinic: "; cin>>clinic;

cout<<"visiting\_time: "; cin>>visiting\_time;

}

void search(); //override the virtual function such that it searches using binary search.

}\*d[10];

//class lawyer (qualification, civil/criminal, casesattended, contact\_no, office\_address)

class lawyer:public telephone\_directory

{ char qualification[20],contact\_no[20],office\_address[20];

char type[20]; //civil/criminal

int no; //casesattended

public:

lawyer()

{ cout<<"qualification: "; cin>>qualification;

cout<<"TYPE:civil/criminal: "; cin>>type;

cout<<"cases\_attended: "; cin>>no;

cout<<"contact\_no: "; cin>>contact\_no;

cout<<"office\_address: "; cin>>office\_address;

}

void search();

}\*l[10];

int telephone\_directory::x;

int telephone\_directory::y;

int main()

{ int ch,i;

while(1)

{ cout<<"1.INSERT\n2.SEARCH\n3.EXIT"; cin>>ch;

switch(ch)

{ case 1: system("CLS"); cout<<"1.DOCTOR\_insert\n2.LAWYER\_insert"; cin>>ch;

switch(ch)

{ case 1: d[++telephone\_directory::x]=new doctor(); break;

case 2: l[++telephone\_directory::y]=new lawyer(); break;

}break;

case 2: system("CLS"); cout<<"1.DOCTOR\_search\n2.LAWYER\_search"; cin>>ch;

switch(ch)

{ case 1: for(i=1;i<=telephone\_directory::x;i++)

{ cout<<i<<"\n\n"; d[i]->search(); cout<<"\_\_\_\_\_\_\_\_\n"; }break;

case 2: for(i=1;i<=telephone\_directory::y;i++)

{ cout<<i<<"\n\n"; l[i]->search(); cout<<"\_\_\_\_\_\_\_\_"; }break;

}break;

case 3: exit(0);

}getch();

}return 0;

}

void doctor::search()

{ cout<<"NAME: "<<name<<endl;

cout<<"OCCUPATION: "<<occupation<<endl;

cout<<"PHONE\_NO: "<<phone<<endl;

cout<<"ADDRESS: "<<address<<endl;

cout<<"qualification: "<<qualification<<endl;

cout<<"clinic: "<<clinic<<endl;

cout<<"visiting\_time: "<<visiting\_time<<endl;

}

void lawyer::search()

{ cout<<"NAME: "<<name<<endl;

cout<<"OCCUPATION: "<<occupation<<endl;

cout<<"PHONE\_NO: "<<phone<<endl;

cout<<"ADDRESS: "<<address<<endl;

cout<<"qualification: "<<qualification<<endl;

cout<<"TYPE:civil/criminal: "<<type<<endl;

cout<<"cases\_attended: "<<no<<endl;

cout<<"contact\_no: "<<contact\_no<<endl;

cout<<"office\_address: "<<office\_address<<endl;

}

**Sample Input-Output:**

1. **INSERT**
2. **SEARCH**
3. **EXIT 1**
4. **DOCTOR\_insert**
5. **LAWYER\_insert 1**

**NAME: ABC**

**OCCUPATION: ABC**

**PHONE\_NO: 123**

**ADDRESS: ABC**

**QUALIFICATION: ABC  
CLINIC: ABC**

**VISITING\_TIME: 123**

1. **INSERT**
2. **SEARCH**
3. **EXIT 2**

**1 . DOCTOR\_SEARCH**

**2 . LAWYER\_search 1**

**1.**

**NAME: ABC**

**OCCUPATION: ABC**

**PHONE\_NO: 123**

**ADDRESS: ABC**

**QUALIFICATION: ABC  
CLINIC: ABC**

**VISITING\_TIME: 123**

1. **INSERT**
2. **SEARCH**
3. **EXIT 3**