Name: Swarnabh Paul

Section: Y

Roll no: 19CS8122

Assignment no: 7

Questions attempted: 1,2

**Question (1)**

**Code:**

#include<iostream>

#include<stdlib.h>

**using** **namespace** std;

**class** **Tool**

{ /\* Fill in \*/

**protected:**

**int** strength;

**char** type;

**public:**

**void** **setStrength**(**int**);

**int** **getStrength**();

**char** **getType**();

};

**void** Tool::setStrength(**int** str)

{

strength=str;

}

**int** Tool::getStrength()

{

**return** strength;

}

**char** Tool::getType()

{

**return** type;

}

/\*Implement class Scissors \*/

**class** **Scissors**: **public** Tool

{

**public:**

Scissors(**int**);

**bool** **fight**(Tool);

};

Scissors::Scissors(**int** str)

{

setStrength(str);

type='s';

}

**bool** Scissors::fight(Tool t)

{

**char** OpType=t.getType();

**int** OpStrength=t.getStrength();

**if**(OpType=='r')

**return** (strength>**2**\*OpStrength);

**else** **if**(OpType=='p')

**return** (**2**\*strength>OpStrength);

**else**

**return** (strength>OpStrength);

}

/\*Implement class Paper \*/

**class** **Paper**: **public** Tool

{

**public:**

Paper(**int**);

**bool** **fight**(Tool);

};

Paper::Paper(**int** str)

{

setStrength(str);

type='p';

}

**bool** Paper::fight(Tool t)

{

**char** OpType=t.getType();

**int** OpStrength=t.getStrength();

**if**(OpType=='s')

**return** (strength>**2**\*OpStrength);

**else** **if**(OpType=='r')

**return** (**2**\*strength>OpStrength);

**else**

**return** (strength>OpStrength);

}

/\*Implement class Rock \*/

**class** **Rock**: **public** Tool

{

**public:**

Rock(**int**);

**bool** **fight**(Tool);

};

Rock::Rock(**int** str)

{

setStrength(str);

type='r';

}

**bool** Rock::fight(Tool t)

{

**char** OpType=t.getType();

**int** OpStrength=t.getStrength();

**if**(OpType=='p')

**return** (strength>**2**\*OpStrength);

**else** **if**(OpType=='s')

**return** (**2**\*strength>OpStrength);

**else**

**return** (strength>OpStrength);

}

**int** main() {

// Example main function

// You may add your own testing code if you like

Scissors s1(**5**);

Paper **p1**(**7**);

Rock **r1**(**15**);

cout << s1.fight(p1) << p1.fight(s1) << endl;

cout << p1.fight(r1) << r1.fight(p1) << endl;

cout << r1.fight(s1) << s1.fight(r1) << endl;

**return** **0**;

}

**Output:**

10

01

10

**Question (2)**

**Code:**

#include<iostream>

#include<stdbool.h>

**using** **namespace** std;

**class** **Employee**

{

**int** empid;

**float** grossSalary, basic, pension;

**const** **int** bonus;

string designation;

**static** **float** da;

**void** **CalcSalary**();

**protected:**

**int** leave;

**void** **displayEmployeeDetails**();

**void** **inputEmployeeDetails**();

**public:**

Employee(**int**,**float**,string,**int**);

**virtual** **void** display()=**0**;

**virtual** **void** input()=**0**;

**virtual** **void** takeLeave(**int**)=**0**;

**static** **void** **dahike**(**float** d);

};

**float** Employee::da=**1.5**;

**void** Employee::dahike(**float** f)

{

da+=f;

}

Employee::Employee(**int** bn,**float** pn,string desgn,**int** l): bonus(bn)

{

grossSalary=basic=**0**;

empid=**0**;

designation=desgn;

pension=pn;

leave=l;

}

**void** Employee::CalcSalary()

{

grossSalary=basic+(bonus+(basic\*da/**100.0**));

}

**void** Employee::displayEmployeeDetails()

{

CalcSalary();

cout<<"Emp id: "<<empid<<endl;

cout<<"Designation: "<<designation<<endl;

cout<<"Salary: "<<grossSalary<<endl;

cout<<"Pension: "<<pension<<endl;

cout<<"Leave: "<<leave<<endl;

}

**void** Employee::inputEmployeeDetails()

{

cout<<"Input emp id: "; cin>>empid;

cout<<"Input basic salary: "; cin>>basic;

}

**class** **AdminOfficer**: **virtual** **public** Employee

{

string dept;

**protected:**

**int** officeId;

**void** **displayAdminDetails**();

**void** **inputAdminDetails**();

**public:**

AdminOfficer(**int**);

**void** **display**();

**void** **input**();

**void** **takeLeave**(**int**);

**virtual** **void** **callMeeting**();

**virtual** **void** **doOfficeWork**();

};

**void** AdminOfficer::callMeeting()

{

cout<<"Meeting called by administrative officer in office "<<officeId<<endl;

}

**void** AdminOfficer::doOfficeWork()

{

cout<<"Office work done by administrative officer in office "<<officeId<<endl;

}

AdminOfficer::AdminOfficer(**int** bn): Employee(bn,**25000**,"Administrative Officer",**15**)

{

}

**void** AdminOfficer::inputAdminDetails()

{

cout<<"Input department: "; cin>>dept;

cout<<"Input office id: "; cin>>officeId;

}

**void** AdminOfficer::displayAdminDetails()

{

cout<<"Department: "<<dept<<endl;

cout<<"Office id: "<<officeId<<endl;

}

**void** AdminOfficer::display()

{

displayEmployeeDetails();

displayAdminDetails();

}

**void** AdminOfficer::input()

{

inputEmployeeDetails();

inputAdminDetails();

}

**void** AdminOfficer::takeLeave(**int** l)

{

cout<<"For Admin Officer: ";

**if**(leave-l>=**0**)

{

cout<<"Taken leave for "<<l<<" days"<<endl;

leave-=l;

}

**else**

{

cout<<"Not enough credit left to take leave"<<endl;

}

}

**class** **Faculty**: **virtual** **public** Employee

{

**bool** PHDsupervision;

**protected:**

string course;

**void** **inputFacultyDetails**();

**void** **displayFacultyDetails**();

**public:**

Faculty(**int**);

**void** **display**();

**void** **input**();

**void** **takeLeave**(**int**);

**virtual** **void** **takeLecture**();

};

**void** Faculty::takeLecture()

{

cout<<"Lecture taken for "<<course<<" by faculty"<<endl;

}

Faculty::Faculty(**int** bn): Employee(bn,**20000**,"Faculty",**10**)

{

}

**void** Faculty::inputFacultyDetails()

{

cout<<"Input course: "; cin>>course;

**char** ch;

cout<<"Doing PHD supervision? <y/n>: "; cin>>ch;

**if**(ch=='y'||ch=='Y')

PHDsupervision=true;

**else**

PHDsupervision=false;

}

**void** Faculty::displayFacultyDetails()

{

cout<<"Course: "<<course<<endl;

**if**(PHDsupervision)

cout<<"Doing PHD supervision"<<endl;

}

**void** Faculty::display()

{

displayEmployeeDetails();

displayFacultyDetails();

}

**void** Faculty::input()

{

inputEmployeeDetails();

inputFacultyDetails();

}

**void** Faculty::takeLeave(**int** l)

{

cout<<"For Faculty: ";

**if**(leave-l>=**0**)

{

cout<<"Taken leave for "<<l<<" days"<<endl;

leave-=l;

}

**else**

{

cout<<"Not enough credit left to take leave"<<endl;

}

}

**class** **Dean**: **public** AdminOfficer, **public** Faculty

{

string section;

**protected:**

**void** **inputDeanDetails**();

**void** **displayDeanDetails**();

**public:**

Dean(**int**);

**void** **input**();

**void** **display**();

**void** **takeLeave**(**int**);

**void** **takeLecture**();

**void** **callMeeting**();

**void** **doOfficeWork**();

};

**void** Dean::takeLecture()

{

cout<<"Lecture taken for "<<course<<" by dean"<<endl;

}

**void** Dean::callMeeting()

{

cout<<"Meeting called by dean in seminar hall"<<endl;

}

**void** Dean::doOfficeWork()

{

cout<<"Office work done by dean in office "<<officeId<<endl;

}

Dean::Dean(**int** bn): AdminOfficer(bn), Faculty(bn), Employee(bn,**40000**,"Dean",**30**)

{

}

**void** Dean::inputDeanDetails()

{

cout<<"Input section: "; cin>>section;

}

**void** Dean::displayDeanDetails()

{

cout<<"Section: "<<section<<endl;

}

**void** Dean::input()

{

inputEmployeeDetails();

inputAdminDetails();

inputFacultyDetails();

inputDeanDetails();

}

**void** Dean::display()

{

displayEmployeeDetails();

displayAdminDetails();

displayFacultyDetails();

displayDeanDetails();

}

**void** Dean::takeLeave(**int** l)

{

cout<<"For Dean: ";

**if**(leave-l>=**0**)

{

cout<<"Taken leave for "<<l<<" days"<<endl;

leave-=l;

}

**else**

{

cout<<"Not enough credit left to take leave"<<endl;

}

}

**int** main()

{

AdminOfficer a1(**5000**);

a1.input();

cout<<endl;

a1.takeLeave(**12**);

cout<<endl;

a1.display();

cout<<endl;

a1.callMeeting();

cout<<endl;

a1.doOfficeWork();

cout<<endl;

Faculty **f1**(**3000**);

f1.input();

cout<<endl;

f1.takeLeave(**7**);

cout<<endl;

f1.display();

cout<<endl;

f1.takeLecture();

cout<<endl;

Dean **d1**(**10000**);

d1.input();

cout<<endl;

d1.takeLeave(**22**);

cout<<endl;

d1.display();

cout<<endl;

d1.callMeeting();

cout<<endl;

d1.doOfficeWork();

cout<<endl;

d1.takeLecture();

cout<<endl;

Employee::dahike(**0.5**);

a1.display();

cout<<endl;

f1.display();

cout<<endl;

d1.display();

cout<<endl;

**return** **0**;

}

**Output:**

Input emp id: 121

Input basic salary: 50000

Input department: Technical

Input office id: 32

For Admin Officer: Taken leave for 12 days

Emp id: 121

Designation: Administrative Officer

Salary: 55750

Pension: 25000

Leave: 3

Department: Technical

Office id: 32

Meeting called by administrative officer in office 32

Office work done by administrative officer in office 32

Input emp id: 123

Input basic salary: 30000

Input course: CSC401

Doing PHD supervision? <y/n>: y

For Faculty: Taken leave for 7 days

Emp id: 123

Designation: Faculty

Salary: 33450

Pension: 20000

Leave: 3

Course: CSC401

Doing PHD supervision

Lecture taken for CSC401 by faculty

Input emp id: 155

Input basic salary: 80000

Input department: Educational

Input office id: 25

Input course: CSC402

Doing PHD supervision? <y/n>: y

Input section: Academic

For Dean: Taken leave for 22 days

Emp id: 155

Designation: Dean

Salary: 91200

Pension: 40000

Leave: 8

Department: Educational

Office id: 25

Course: CSC402

Doing PHD supervision

Section: Academic

Meeting called by dean in seminar hall

Office work done by dean in office 25

Lecture taken for CSC402 by dean

Emp id: 121

Designation: Administrative Officer

Salary: 56000

Pension: 25000

Leave: 3

Department: Technical

Office id: 32

Emp id: 123

Designation: Faculty

Salary: 33600

Pension: 20000

Leave: 3

Course: CSC401

Doing PHD supervision

Emp id: 155

Designation: Dean

Salary: 91600

Pension: 40000

Leave: 8

Department: Educational

Office id: 25

Course: CSC402

Doing PHD supervision

Section: Academic