Philadelphia University Faculty of Engineering Department of Computer Engineering		Second Semester 2017/2018 Date:- 03/05/2018 Allowed time:- 60 minutes

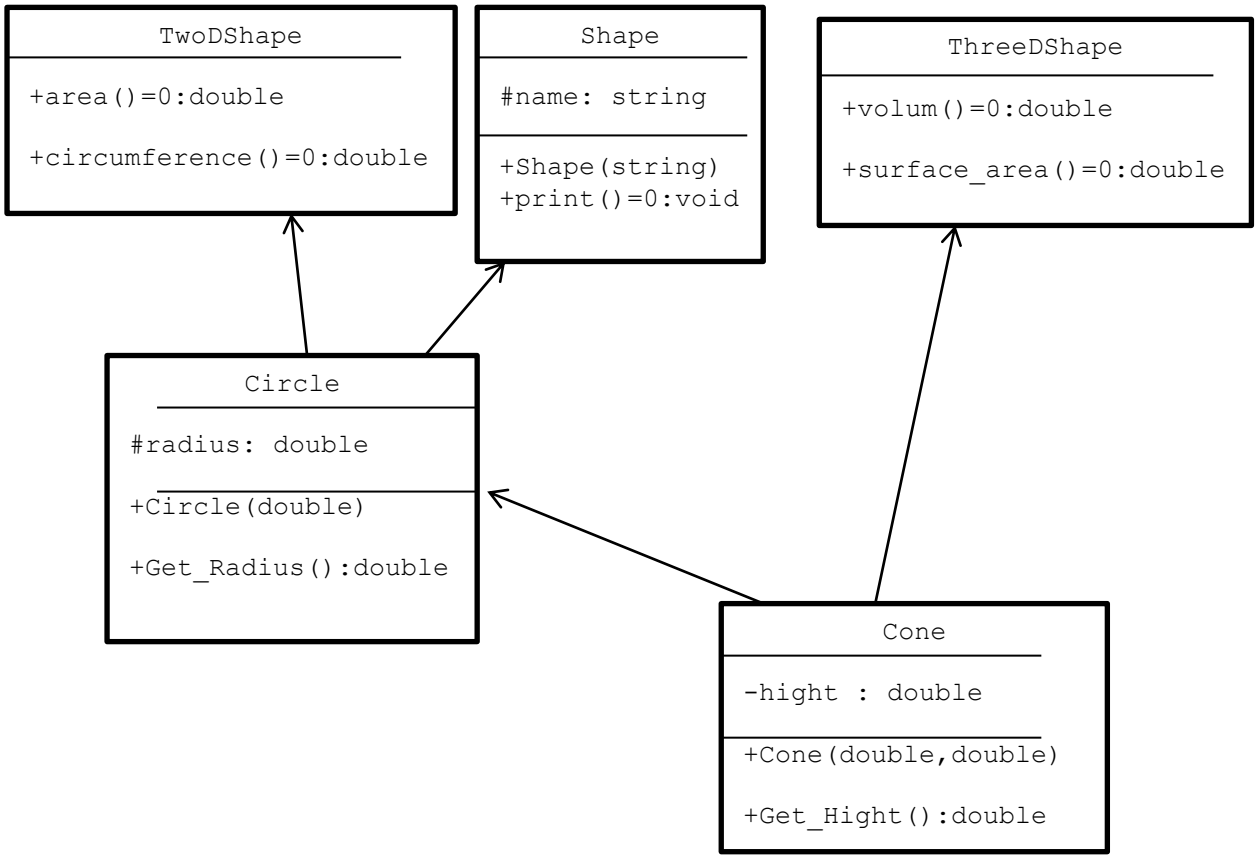
Object Oriented Programming (630221) Second Exam

Student Name: - …… ID: - ……

Question 1: Mark the following statements as TRUE or False. 20 points

1- Public data member in base class are public in derived class regardless of type of inherence used.	F
2- Abstract class cannot have a member data	F
3- Friend function can only access public member of the class	F
4- Derived class can only have one base class.	F
5- Interface is an abstract class with all its methods are pure virtual methods	T
6- Pure virtual methods may or may not be overridden in derived class	F
7- Operator overloading function must be friend function of a class	T
8- Constructor of base class must be explicitly called from derived class	F
9- Template data types are specified at compile time	F
10- derived class pointer can be used to reference base class object	F

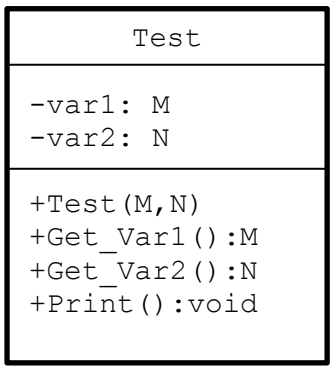
Question 2:Given the following UML diagram 25 points



Where TwoDShape and ThreeDShape are Interfaces, shape is an abstract class. Write a C++ code that construct the classes above and define the required methods

```
class TwoDShape
{
public:
	virtual double area()=0;
	virtual double circumference()=0;
};
class ThreeDShape
{
public:
	virtual double volume()=0;
	virtual double surface_area()=0;
};
class Shape
{
protected:
	string name;
public:
	Shape(string n){name=n;}
	virtual void print()=0;
};
class Circle:public Shape,public TwoDShape
{
protected:
	double radius;
public:
	Circle(double r):Shape("circle"){radius=r;}
	double Get_Radius(){return radius;}
	double area(){return pow(radius,2.0)*3.14;}
	double circuference(){return 2*radius*3.14;}
	void print()
	{
		cout<<"radius="<<radius<<endl;
		cout<<"area="<<area()<<endl;
		cout<<"circuference="<<circumference()<<endl;
	}
};
class Cone:public Circle,public ThreeDShape
{
private:
	double hight;
public:
	Cone(double r,double h):Circle(r){hight=h; name="Cone";}
	double Get_Hight(){return hight;}
	double volume(){return 1.0/3.0*hight*area();}
	double curface_area(){return
area()+3.14*radius*sqrt(pow(radius,2.0)+pow(hight,2.0));}
	void print()
	{
		cout<<"radius="<<radius<<endl;
		cout<<"hight="<<hight<<endl;
		cout<<"volume="<<volume()<<endl;
		cout<<"surface_area="<<curface_area()<<endl;
	}
};
```

Question 3: Given the following class diagram 25 points



Where M , N are template variables. Write a C++ code that Define the class above using templates. Method Print should print var1 and var2 and the constructor should set var1 and var2.

```
template<class M,class N> class Test
{
private:
	M var1;
	N var2;
public:
	Test(M,N);
	M Get_Var1();
	N Get_Var2();
	void Print();
};
template<class M,class N> Test<M,N>::Test(M v1,N v2)
{
	var1=v1;
	var2=v2;
}
template<class M,class N> M Test<M,N>::Get_Var1(){return var1;}
template<class M,class N> N Test<M,N>::Get_Var2(){return var2;}
template<class M,class N> void Test<M,N>::Print()
{cout<<"var1="<<var1<<"\nvar2="<<var2<<endl;}
```

Question 4:Given the following C++ code 10 points

```
class Test
{
private:
      int num;
public:
      Test(int n){num=n;}
      int get_num(){return num;}
};
```

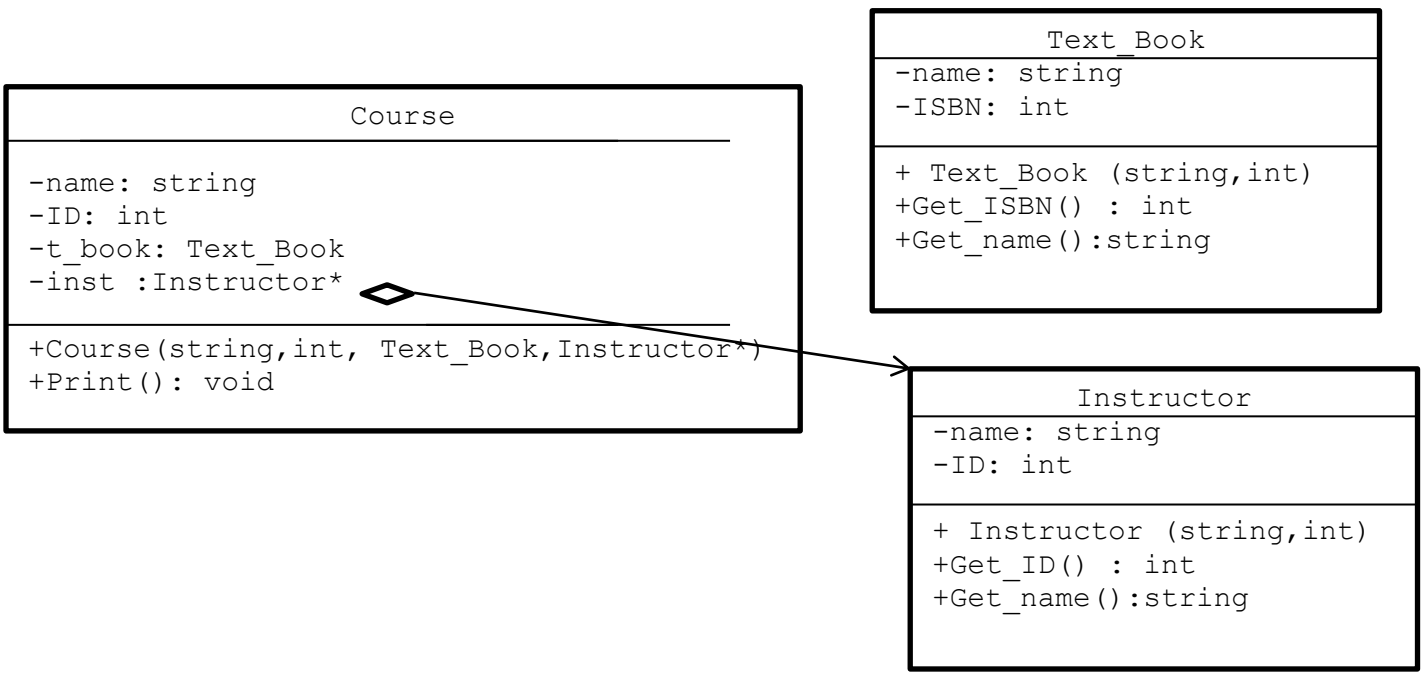
1- Override the input >> operator so class Test can be used with cin statement.
2- Override the % (reminder) operator so the result is class test that contain the reminder of num from the first object devided by num of the second object.

```
class Test
{
      friend istream& operator>>(istream& in,Test& T);
      friend Test operator%(Test& T1,Test& T2);
private:
      int num;
public:
      Test(int n){num=n;}
      int get_num(){return num;}
};
istream& operator>>(istream& in,Test& T)
{
      in>>T.num;
      return in;
}

Test operator%(Test& T1,Test& T2)
{
      Test T(T1.num%T2.num);
      return T;
}
```

Question 5:Given the following UML diagram that represent an aggregation and composition relationships. 20 points

1- write C++ code that define the structure bellow.
2- Write a main function that test the structure bellow.



```
class Text_Book
{
private:
	string name;
	int ISBN;
public:
	Text_Book(){}
	Text_Book(string n,int isbn)
	{
		name=n;
		ISBN=-isbn;
	}
	int Get_ISBN(){return ISBN;}
	string Get_Name(){return name;}
};
class Instructor
{
private:
	string name;
	int ID;
public:
	Instructor(string n,int id)
	{
		name=n;
		ID=id;
	}
	int Get_ID(){return ID;}
	string Get_Name(){return name;}
};

class Course
{
private:
	string name;
	int ID;
	Text_Book t_book;
	Instructor* inst;
public:
	Course(string n,int id,Text_Book tb,Instructor* INST)
	{
		name=n;
		ID=id;
		t_book=tb;
		inst=INST;
	}
	void Print()
	{
		cout<<"name: "<<name<<endl;
		cout<<"ID: "<<ID<<endl;
		cout<<"text book: "<<t_book.Get_Name()<<endl;
		cout<<"inst: "<<inst->Get_Name();
	}
};

int main()
{
	Text_Book tb("C++ programming",123456);
	Instructor inst("John Smith",123);
	Course C("OOP",630221,tb,&inst);
	C.Print();


	return 0;
}
```