(VIRTUAL FUNCTIONS)

(MADE BY ALI AKBER)

(BSCS 2ND SS1)

//NORMAL MEMBER FUNCTIONS ACCESSED WITH POINTERS.

```
#include<iostream>
using namespace std;
class Base{
	public:
	void show(){
		cout<<"Base"<<endl;
	}
};
class Derive1: public Base{
	public:
	void show(){
		cout<<"Derive1"<<endl;
	}
};
class Derive2: public Base{
	public:
	void show(){
		cout<<"Derive2"<<endl;
	}
};
int main(){
	Derive1 dev1;
```

```
	Derive2 dev2;


	Base* ptr;
	ptr=&dev1;
	ptr->show();
	ptr=&dev2;
	ptr->show();


	return 0;
}
```

//Virtual MEMBER FUNCTIONS ACCESSED WITH POINTERS.

```
#include<iostream>
using namespace std;
class Base{
	public:
	virtual void show(){
		cout<<"Base"<<endl;
	}
};
class Derive1: public Base{
	public:
	void show(){
		cout<<"Derive1"<<endl;
	}
};
class Derive2: public Base{
```

```
	public:
	void show(){
		cout<<"Derive2"<<endl;
	}
};
int main(){
	Derive1 dev1;
	Derive2 dev2;

	Base* ptr;
	ptr=&dev1;
	ptr->show();
	ptr=&dev2;
	ptr->show();

	return 0;
}
```

//Virtual MEMBER FUNCTIONS ACCESSED WITH POINTERS.

//USING AN ARRAY OF POINTER.

```
#include<iostream>
using namespace std;
class Base{
	public:
	virtual void show(){
		cout<<"Base"<<endl;
	}
```

```
};
class Derive1: public Base{
	public:
	void show(){
		cout<<"Derive1"<<endl;
	}
};
class Derive2: public Base{
	public:
	void show(){
		cout<<"Derive2"<<endl;
	}
};
class Derive3: public Base{
	public:
	void show(){
		cout<<"Derive3"<<endl;
	}
};
int main(){
	Base* ptr[3];
	/*Derive1 dev1;
	Derive2 dev2;
	Derive3 dev3;

	ptr[0]=&dev2;		//static memory allocation.
```

```
	ptr[1]=&dev1;
	ptr[2]=&dev3; */


	ptr[0]=new Derive2();		//dynamic memory allocation.
	ptr[1]=new Derive1();
	ptr[2]=new Derive3();

	for (int i=0;i<3;i++){
		ptr[i]->show();
	}

	return 0;
}
```

//PURE VIRTUAL FUNCTION.

//VIRTUAL DESTRUCTOR.

//ANIMAL QUESTION ASSIGNMENT.

```
#include<iostream>
using namespace std;
class Animal{
	public:
	virtual void speak()=0;	//pure virtual function.
	virtual	~Animal(){
		cout<<"I am a animal"<<endl;
	}
};
```

```
class Cat: public Animal{
	void speak(){
		cout<<"meeyaon meeyaon"<<endl;
	}
	virtual  ~Cat(){
		cout<<"I am a Cat destructor."<<endl;
	}
};
class Dog: public Animal{
	void speak(){
		cout<<"Bhaao Bhaaoo"<<endl;
	}
	virtual  ~Dog(){
		cout<<"I am a Dog destructor."<<endl;
	}
};
class Cow: public Animal{
	void speak(){
		cout<<"Gaan Gaan"<<endl;
	}
	virtual  ~Cow(){
		cout<<"I am a Cow destructor."<<endl;
	}
};
class Donkey: public Animal{
	void speak(){
```

```
		cout<<"Dhainchu Dhainchu"<<endl;
	}
	virtual ~Donkey(){
		cout<<"I am a Donkey destructor."<<endl;
	}
};
class Goat: public Animal{
	void speak(){
		cout<<"Mainn Mainn"<<endl;
	}
	virtual ~Goat(){
		cout<<"I am a Goat destructor."<<endl;
	}
};
class Snake: public Animal{
	void speak(){
		cout<<"SSSS SSSS"<<endl;
	}
	virtual ~Snake(){
		cout<<"I am a Snake destructor."<<endl;
	}
};
class Crow: public Animal{
	void speak(){
		cout<<"Kaaan Kaan"<<endl;
	}
```

```
        virtual   ~Crow(){
                cout<<"I am a Snake destructor."<<endl;
        }
};
int main(){
        Animal *ptr[100];
    int choice, n = 0;
    char opt = 'y';
    do
    {
        cout<<"1-Cat"<<endl;
        cout<<"2-Dog"<<endl;
        cout<<"3-Cow"<<endl;
        cout<<"4-Goat"<<endl;
        cout<<"5-Donkey"<<endl;
        cout<<"6-Snake"<<endl;
        cout<<"7-Crow"<<endl;

        cout<<"Enter option ";
        cin>>choice;
        switch(choice)
        {
            case 1:
                    ptr[n++] = new Cat();
                break;
            case 2:
```

```
                ptr[n++] = new Dog();
            break;
        case 3:
                ptr[n++] = new Cow();
            break;
        case 4:
                ptr[n++] = new Goat();
            break;
        case 5:
                ptr[n++] = new Donkey();
            break;
        case 6:
                ptr[n++] = new Snake();
            break;
        case 7:
                ptr[n++] = new Crow();
            break;
        default:
            cout<<"Invalid choice"<<endl;
    }
    cout<<"Do you want to continue (y) ";
    cin>>opt;
}while(opt == 'Y' || opt == 'y');


cout<<endl;
```

```
	for(int j = 0; j < n; j++)
	{
		ptr[j]->speak();
		//delete ptr[j];
	}

	return 0;
}
```

//COLLEGE QUESTION.

```
#include<iostream>
#include<stdlib.h>
using namespace std;

class College
{
	public:
	   virtual void fee() = 0;
};
class preeng:public College
{
	public:
		void fee()
		{
			cout<<" For Fsc.Pre Engineering : 1Lakh/per year"<<endl;
		}
};
```

```
class premed:public College
{
	public:
		void fee()
		{
			cout<<" For Fsc.Pre Medical : 1Lakh/per year"<<endl;
		}
};
class ics:public College
{
	public:
		void fee()
		{
			cout<<" For I.CS : 80Thousand /per year"<<endl;
		}
};
class icom:public College
{
	public:
		void fee()
		{
			cout<<" For I.Com : 60Thousand/per year"<<endl;
		}
};
class fait:public College
{
```

```
	public:
		void fee()
		{
			cout<<" For FA-IT: 50Thousand/per year"<<endl;
		}
};
class fa:public College
{
	public:
		void fee()
		{
			cout<<" For FA: 40Thousand/per year"<<endl;
		}
};
class scholarship:public College
{
	public:
		void fee()
		{
			cout<<" 100% Scholarship on 95% marks "<<endl;
			cout<<" 80% Scholarship on 90% marks "<<endl;
			cout<<" 60% Scholarship on 85% marks "<<endl;
			cout<<" 50% Scholarship on 80% marks "<<endl;
		}
};
int main()
```

```
{
        College *ptr[100];
        int choice,i=0;
        char opt;
        do
        {
                cout<<"\n\t SARGODHA COLLEGES \t\n"<<endl;
                cout<<" \n FEE STRUCTURE \n"<<endl;
                cout<<" Select your desired Degree:"<<endl;
                cout<<" 1 For Fsc Pre Engineering"<<endl;
                cout<<" 2 For Fsc Pre Medicl"<<endl;
                cout<<" 3 For ICS"<<endl;
                cout<<" 4 For Icom"<<endl;
                cout<<" 5 For FA-IT"<<endl;
                cout<<" 6 For FA"<<endl;
                cout<<" 7 For Scholarships"<<endl;
                cout<<" 8 For Exit"<<endl;
                cin>>choice;
        cout<<endl;
        switch(choice)
        {
                case 1:
                        ptr[i++]=new preeng();
                break;
        case 2:
                        ptr[i++]=new premed();
```

```
            break;
        case 3:
                    ptr[i++]=new ics();
            break;
        case 4:
                    ptr[i++]=new icom();
            break;
        case 5:
                    ptr[i++]=new fait();
            break;
        case 6:
                    ptr[i++]=new fa();
            break;
        case 7:
                    ptr[i++]=new scholarship();
            break;
        case 8:
                    exit(1);
            break;
            default:
                cout<<"\n\t you entered invalid option\t\n"<<endl;
        }
        cout<<"\n\t DO YOU WANT TO CONTINUE (y/n)): \t\n"<<endl;
        cin>>opt;
}
        while(opt == 'Y'|| opt =='y');
```

```
        cout<<endl;
        for(int j=0;j<i;j++)
        {
                ptr[j]->fee();
        }
        return 0;
}
```

//PERSON QUESTION.

```
#include <iostream>
using namespace std;

class person
{
    protected:
        char name[40];
    public:
        virtual void getData() /// virtual func
        {
            cout << "Enter name: ";
            cin >> name;
        }
        void putName()
        {
            cout << "Name is: " << name << endl;
        }
        virtual bool isOutstanding() = 0; ///pure virtual func
```

```
};
class student : public person ///student class
{
	private:
		float gpa;
	public:
		void getData() /// virtual function override
		{
			person::getData();
			cout << "Enter student\'s GPA: ";
			cin >> gpa;
		}
		bool isOutstanding()
		{
			return (gpa >= 3.5) ? true : false;
		}
};
class professor : public person
{
	private:
		int numPubs;
	public:
		void getData() ///virtual function override
		{
			person::getData();
			cout << "Enter number of professor\'s publications: ";
```

```
            cin >> numPubs;
        }
        bool isOutstanding()
        {
            return (numPubs >= 10) ? true : false;
        }
};

int main()
{
    person *ptr[50];
    int i = 0;
    char choice;
    do
    {
        cout<<"P for professor"<<endl;
        cout<<"S for Student"<<endl;
        cout<<"Enter choice ";
        cin>>choice;
        if(choice == 'p' || choice == 'P')
        {
            ptr[i] = new professor();
            ptr[i++]->getData();
        }
        else    if(choice == 'S' || choice == 's')
        {
```

```
            ptr[i] = new student();
            ptr[i++]->getData();
        }
        else
            cout<<"Invalid Option"<<endl;

        cout<<"Do you want to continue (y/n) ";
        cin>>choice;
        cout<<endl;

    }while(choice == 'Y' || choice == 'y');

    cout<<endl;

    for(int j = 0; j < i; j++)
    {
        ptr[j]->putName();
        if(ptr[j]->isOutstanding() == true)
            cout<<"you are out standing"<<endl;
        cout<<endl;
    }
    return 0;
}
```

ASSIGNMENT QUESTION.

A publishing company that markets both book and audiocassette versions of its works. Create a class called publication that stores the title (a string or c-string) and price (type float) of a publication. From this class derive two

classes: book, which has a page count (type int) and tape: which has a playing time in minutes (type float). Each of the three classes should have a getdata() function to get its data from the user at the keyboard, and a putdata () function to display the data. Add a member function Oversize () to the book and tape classes. Let's say that a book with more than 500 pages, or a tape with a playing time longer than 90 minutes, is considered oversize. You can access this function from main () and display the string "Oversize" for oversized books and tapes when you display their data. Define appropriate constructors and detractor into the above said classes.

Write a main () program that creates an array of pointers to publication. In a loop, ask the user for data about a particular book or tape.

```
//ASSIGNMENT QUESTION.
#include<iostream>
#include<string.h>
using namespace std;
class Publication{
    protected:
    string title;
    float price;
    public:
    Publication():title(""),price(0){}        //no arg constructor.
    Publication(string tit,float pr):title(tit),price(pr){}        //two arg constructor
    virtual void getdata(){
        cin.ignore();
        cout<<"Enter the title of the publication:"<<endl;
        getline(cin,title);
        cout<<"Enter the price of the publication:"<<endl;
        cin>>price;
    }
```

```
    virtual void putdata(){
        cout<<"Title of the publication is:"<<title<<endl;
        cout<<"Price of the publication is:"<<price<<endl;
    }
    virtual ~Publication(){
        cout<<"I am Publication destructor"<<endl;
    }
    virtual bool oversize()=0;
};
class Book: public Publication{
    protected:
    int pages;
    public:
    Book():pages(0){}    //no arg constructor.
    Book(string tit,float pr,int p):Publication(tit,pr),pages(p){}      //three arg constructor.
    void getdata(){
        cout<<"Enter the pages of the book:"<<endl;
        cin>>pages;
    }
    void putdata(){
        cout<<"Pages of the book is:"<<pages<<endl;
    }
    virtual ~Book(){
        cout<<"I am Book destructor"<<endl;
    }
    bool oversize(){
```

```
        return (pages>500) ?true:false;
    }
};
class Tape: public Publication{
    protected:
    float minutes;
    public:
    Tape():minutes(0){}    //no arg constructor.
    Tape(string tit,float pr,float m):Publication(tit,pr),minutes(m){}      //three arg constructor.
    void getdata(){
        cout<<"Enter the minutes of the tape:"<<endl;
        cin>>minutes;
    }
    void putdata(){
        cout<<"Minutes of the tape is:"<<minutes<<endl;
    }
    virtual ~Tape(){
        cout<<"I am Tape destructor"<<endl;
    }
    bool oversize(){
        return (minutes>90) ?true:false;
    }
};

int main(){
Publication *ptr[100];
```

```
int i=0;
char choice;
do{
	cout<<"B for Book:"<<endl;
	cout<<"T for Tape:"<<endl;
	cout<<"Enter choice:"<<endl;
	cin>>choice;
	if(choice=='b' || choice=='B'){
		ptr[i]= new Book();
		ptr[i++]->getdata();	}
		else if(choice=='t' || choice=='T'){
		ptr[i]= new Tape();
		ptr[i++]->getdata();	}
		else{
	cout<<"Invalid option:"<<endl;
		}
		cout<<"Do you want to continue(y/n)"<<endl;
		cin>>choice;
		cout<<endl;
}
while(choice=='y' || choice=='Y');
cout<<endl;
for(int j=0;j<i;j++){
	ptr[j]->putdata();
	if(ptr[j]->oversize()==true){
		cout<<"OVERSIZED!!!"<<endl;
```

```
            cout<<endl;
        }
}
  return 0;
}
```