Sir Syed CASE Institute of Technology

OOP Lab #6

Friend Classes

SUBMITTED BY:

UBAID AHMAD

ROLL NO:

2410-0011

SUBMITTED TO:

Mam Laiba Tanveer

DATE:

28/10/2025

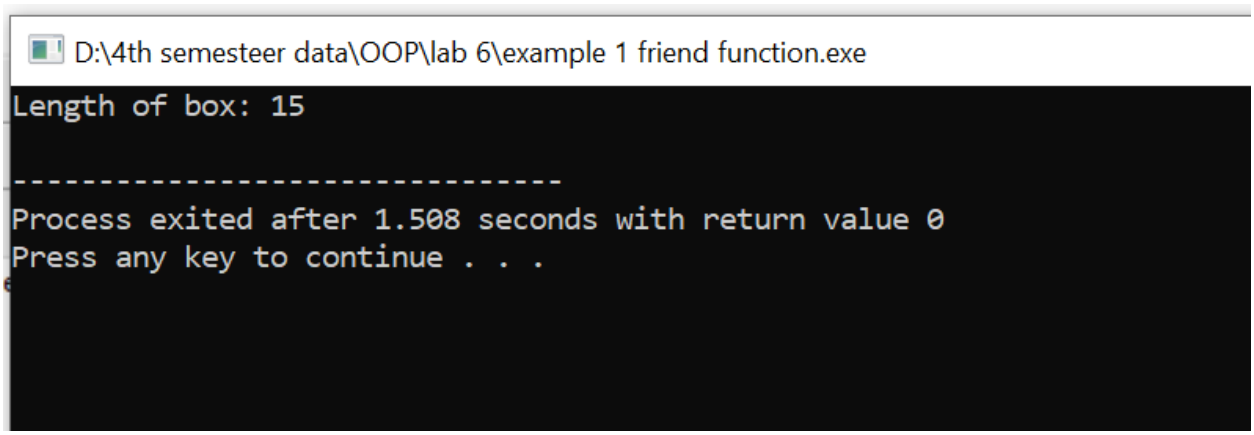
Example 1 Friend Funciton

Code:

```
#include <iostream>
using namespace std;
class Box {
private:
    int length;
public:
        Box(int l) {
           length = l;
        }
     // Declaration only (not definition)
    friend void showLength(Box b);
};
// Definition outside the class
void showLength(Box b) {
    cout<<"Length of box: "<< b.length <<endl;
}

int main() {
        Box box1(15);
        showLength(box1); // friend function accessing private data
        return 0;
}
```

Output:



Example 2 Friend Funciton

Code:

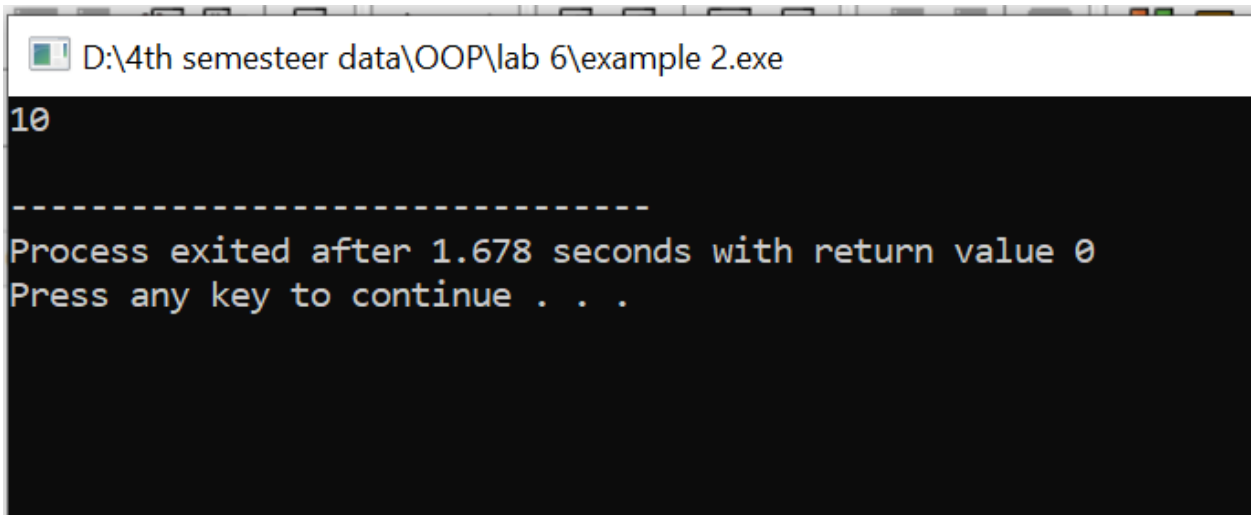
```
#include <iostream>
using namespace std;
class B;

class A
{
	int x;
	public:
	void setdata(int i)
	{
		x = i;
	}
	friend void min(A, B);
};

class B
```

```
{
        int y;
        public:
        void setdata(int i)
        {
                y = i;
        }
        friend void min(A, B);
};

void min(A a, B b)
{
        if(a.x <= b.y)
                cout << a.x << endl;
        else
                cout << b.y << endl;
}
int main()
{
        A a;
        B b;
        a.setdata(10); // Set x = 10
        b.setdata(20); // Set y = 20
        min(a, b);   //friend function call
        return 0;
}
```



Example 3 Friend Class

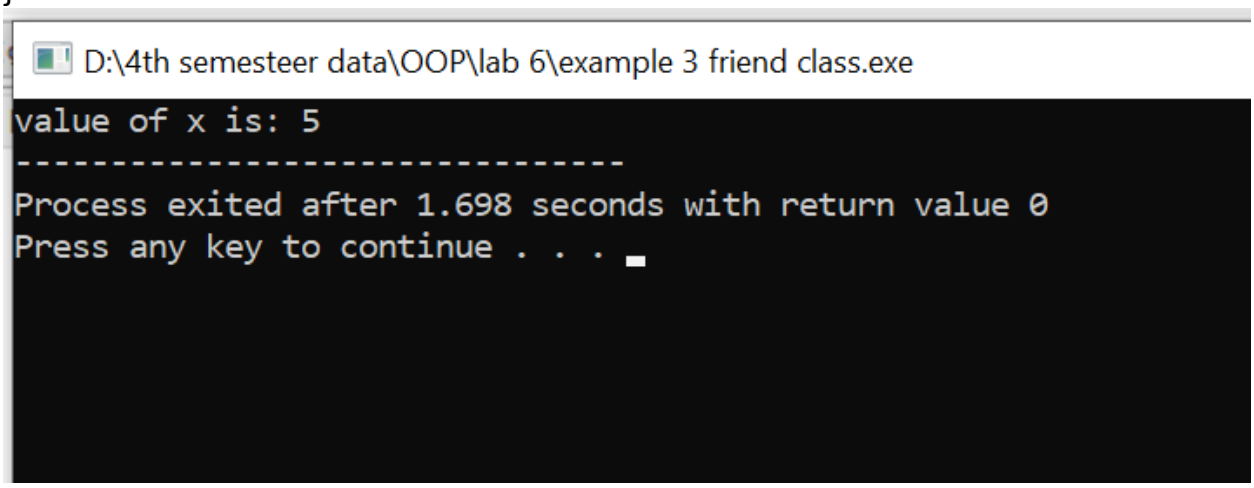
Code:

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

class A {
    int x = 5;
    friend class B;   // friend class
};

class B {
public:
    void display(A &a) {     //we need the actuall object reference not a copy here
        cout << "value of x is: " << a.x;
    }
};
```

```
int main() {
    A a;
    B b;
    b.display(a);
    return 0;
}
```



Task 1 Friend fun Compare area

Code:

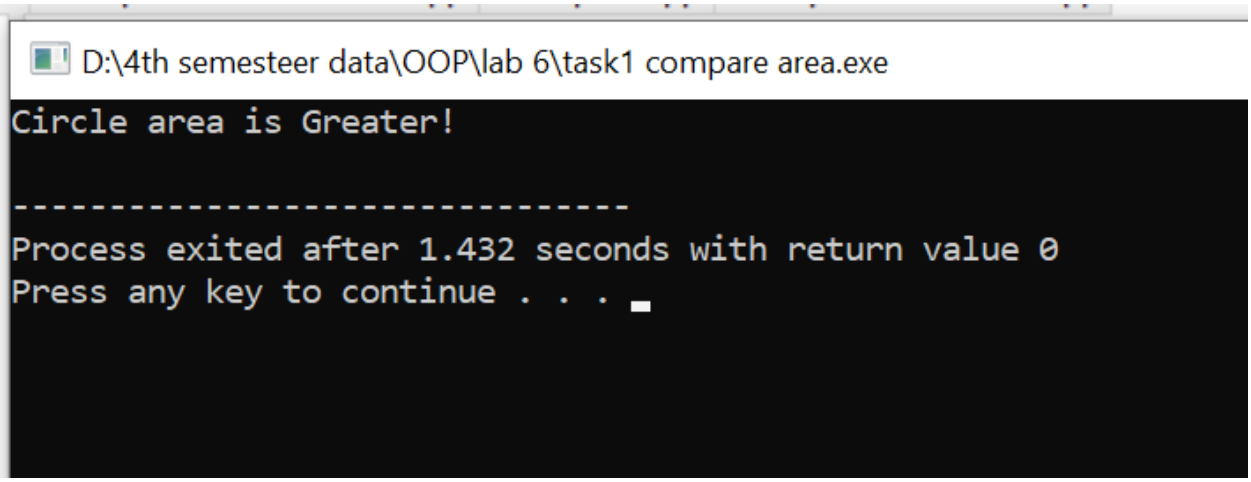
```
#include <iostream>
using namespace std;
class Circle;
class Rectangle{
        double area;
        public:
        Rectangle(double a){
                area=a;
        }
        //friend class
```

```
        friend void compareArea(Rectangle r,Circle c);

};
class Circle{
        double area;
        public:
                Circle(double a){
                        area=a;
                }

        friend void compareArea(Rectangle r,Circle c);
};
void compareArea(Rectangle r,Circle c){
        if(r.area>c.area){
                cout<<"Rectangle area is greater!"<<endl;
        }
        else if(r.area<c.area)
        {
                cout<<"Circle area is Greater!"<<endl;
        }else{
                cout<<"Both have same area!"<<endl;
        }
}
int main(){
        Rectangle r(20);
        Circle c(30);
        compareArea(r,c);
```

```
    return 0;
}
```



Task 2 Friend Class

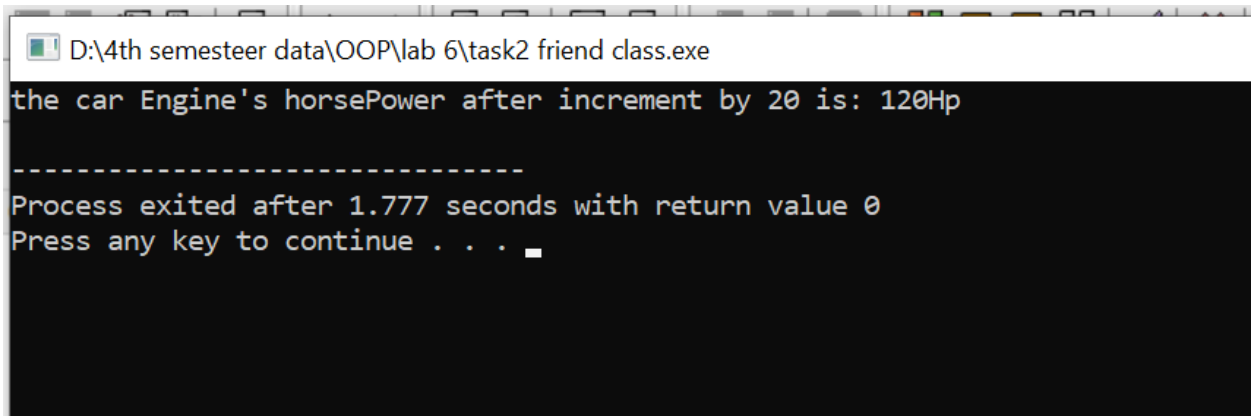
Code:

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

class Car{
        private:
          int horsepower;
        public:
          Car(int value){
                horsepower=value;
          }

        friend class Engine;
};
```

```
class Engine{
                public:
                  int incrementHP(Car &c){
                        c.horsepower+=20;
                        return c.horsepower;
                  }
};
int main(){
                Car c(100);
                Engine e;
                int newHp=e.incrementHP(c);
                cout<<"the car Engine's horsePower after increment by 20 is: "<<newHp<<"Hp"<<endl;
                return 0;
}
```

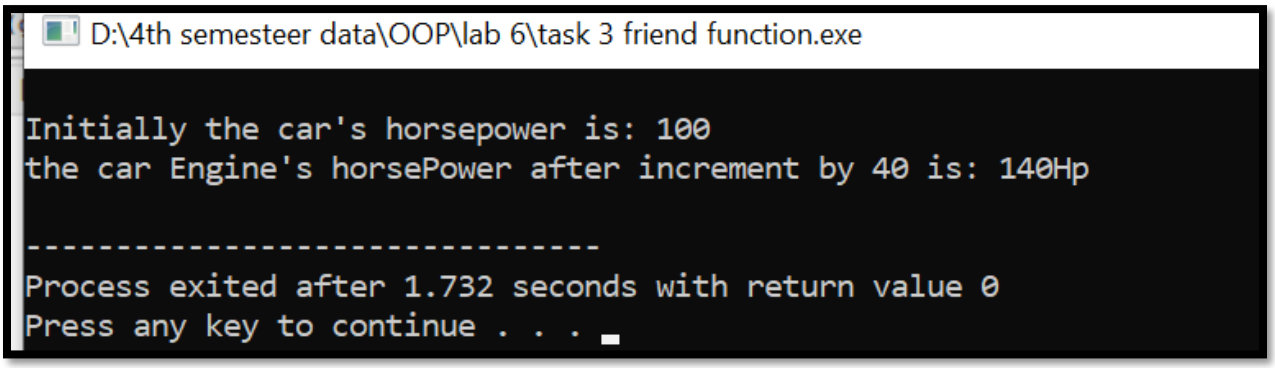


Task 3 friend function

Code:

```
#include <iostream>
using namespace std;
class Car{
```

```
                    private:
                        int horsepower;
                    public:
                        Car(int value){
                            horsepower=value;
                        }
                    friend void incrementHP(Car &c,int val);
};
void incrementHP(Car &c,int val){
                    cout<<"\nInitially the car's horsepower is:
                        "<<c.horsepower<<endl;
                    c.horsepower+=val;
                    cout<<"the car Engine's horsePower after increment
                        by "<<val<<" is: "<<c.horsepower<<"Hp"<<endl;
}
int main(){
                    Car c(100);
                    incrementHP(c,40);
                    return 0;
}
```



Task 4: 2dd array sum and transpose using friend class

Code:

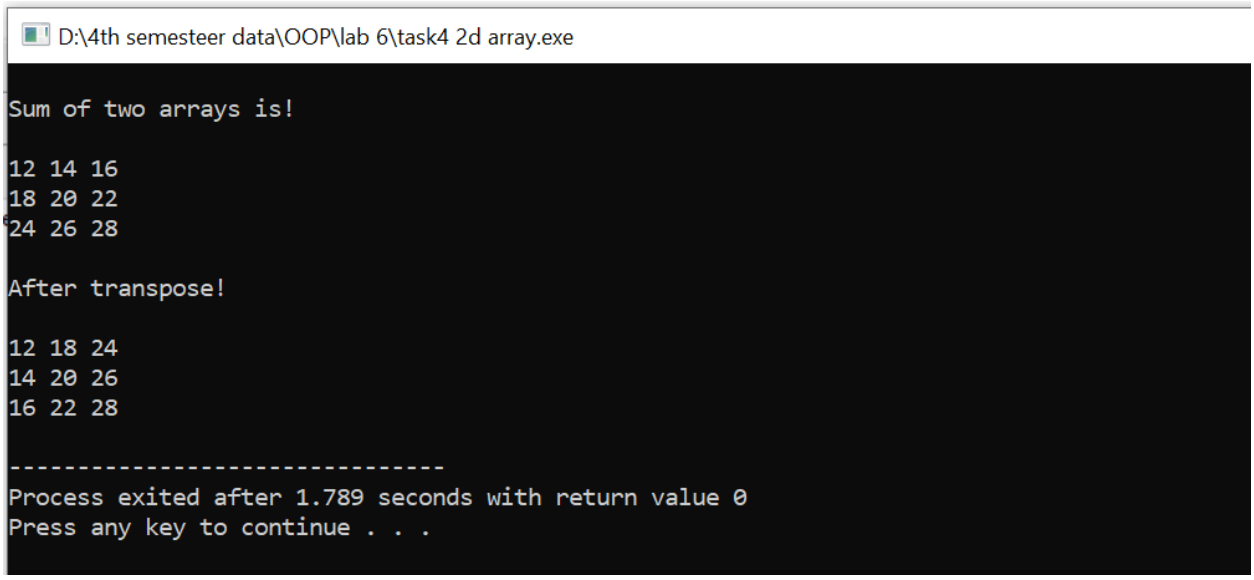
```
#include <iostream>
using namespace std;
class Matrix{
        int arr1[3][3];
        int arr2[3][3];
        public:
           Matrix(int a[3][3],int b[3][3]){
                for(int i=0;i<3;i++){
                        for(int j=0;j<3;j++){
                                arr1[i][j]=a[i][j];
                                arr2[i][j]=b[i][j];
                                }
                        }

                }
                friend class Modify;
};
class Modify{
        public:
                int arr3[3][3];
                void add(Matrix &m){
                        for(int i=0;i<3;i++){
                                for(int j=0;j<3;j++){
```

```
                    arr3[i][j]=m.arr1[i][j]+m.arr2[i][j];
                }
            }
        }
void display(){
            for(int i=0;i<3;i++){
                for(int j=0;j<3;j++){
                    cout<<arr3[i][j]<<" ";
                }
                cout<<endl;
            }

        }
void transpose() {
    int newArr[3][3];
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            newArr[i][j] = arr3[j][i];
        }
    }
    //now storing the result again in arr3
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            arr3[i][j]=newArr[i][j];
        }
    }
     }
```

```
};
int main(){
	int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};
	int b[3][3]={{11,12,13},{14,15,16},{17,18,19}};
	Matrix m(a,b);
	Modify x;
	x.add(m);
	cout<<"\nSum of two arrays is!\n"<<endl;
	x.display();
	x.transpose();
	cout<<"\nAfter transpose!\n"<<endl;
	x.display();
	return 0;
}
```



END,,,