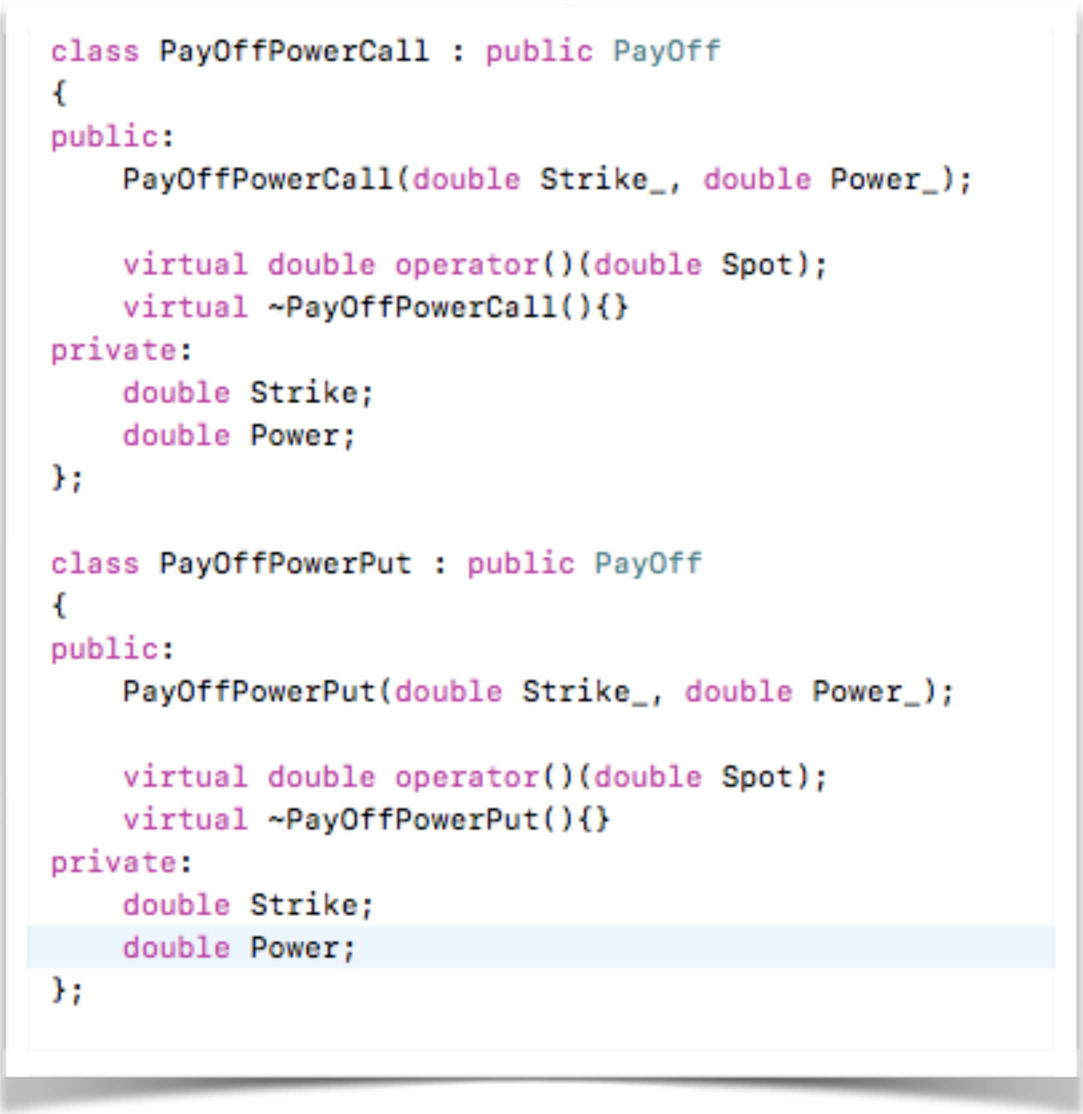
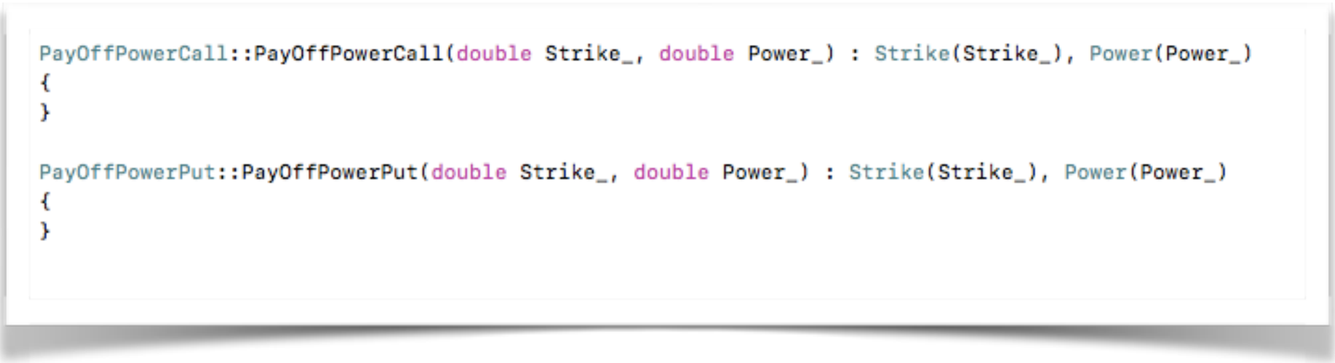
Assignment 3 - FE 545

Question 3.1)
The first question asked to build inheritance for the Power Call Options
I declared two classes called as
PayOffPowerCall & PayOffPowerPut and inherited the PayOff class. Built the operator function for evaluating the value of the options.



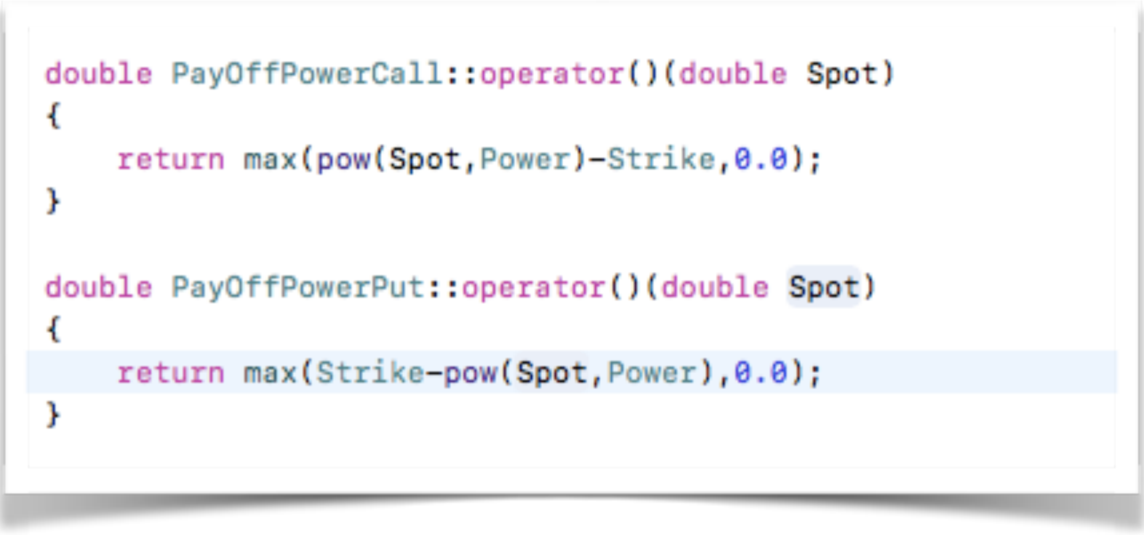
```
class PayOffPowerCall : public PayOff
{
public:
    PayOffPowerCall(double Strike_, double Power_);

    virtual double operator()(double Spot);
    virtual ~PayOffPowerCall(){}
private:
    double Strike;
    double Power;
};

class PayOffPowerPut : public PayOff
{
public:
    PayOffPowerPut(double Strike_, double Power_);

    virtual double operator()(double Spot);
    virtual ~PayOffPowerPut(){}
private:
    double Strike;
    double Power;
};
```

In the constructor I accepted the Strikes as well as the Power. Also declared are the operator and a destructor.



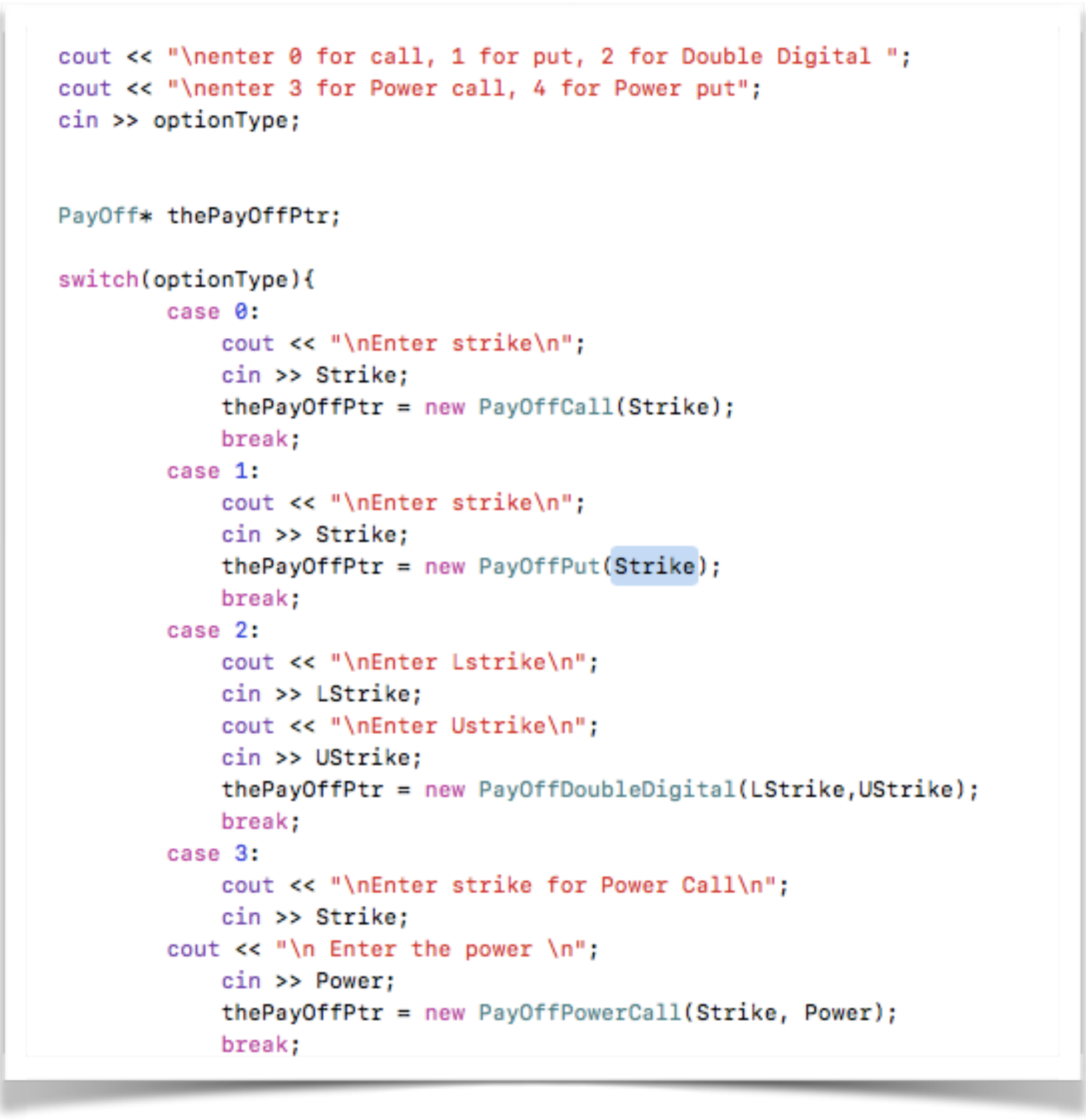
```
PayOffPowerCall::PayOffPowerCall(double Strike_, double Power_) : Strike(Strike_), Power(Power_)
{
}

PayOffPowerPut::PayOffPowerPut(double Strike_, double Power_) : Strike(Strike_), Power(Power_)
{
}
```

For the PayOffPowerCall and Put I used the pow function for which I also included the cmath library



```
double PayOffPowerCall::operator()(double Spot)
{
    return max(pow(Spot,Power)-Strike,0.0);
}

double PayOffPowerPut::operator()(double Spot)
{
    return max(Strike-pow(Spot,Power),0.0);
}
```

To give user functionality I added options for user to make choice out of what kind of options the user wants to make use of



```
cout << "\nenter 0 for call, 1 for put, 2 for Double Digital ";
cout << "\nenter 3 for Power call, 4 for Power put";
cin >> optionType;


PayOff* thePayOffPtr;

switch(optionType){
        case 0:
            cout << "\nEnter strike\n";
            cin >> Strike;
            thePayOffPtr = new PayOffCall(Strike);
            break;
        case 1:
            cout << "\nEnter strike\n";
            cin >> Strike;
            thePayOffPtr = new PayOffPut(Strike);
            break;
        case 2:
            cout << "\nEnter Lstrike\n";
            cin >> LStrike;
            cout << "\nEnter Ustrike\n";
            cin >> UStrike;
            thePayOffPtr = new PayOffDoubleDigital(LStrike,UStrike);
            break;
        case 3:
            cout << "\nEnter strike for Power Call\n";
            cin >> Strike;
        cout << "\n Enter the power \n";
            cin >> Power;
            thePayOffPtr = new PayOffPowerCall(Strike, Power);
            break;
```

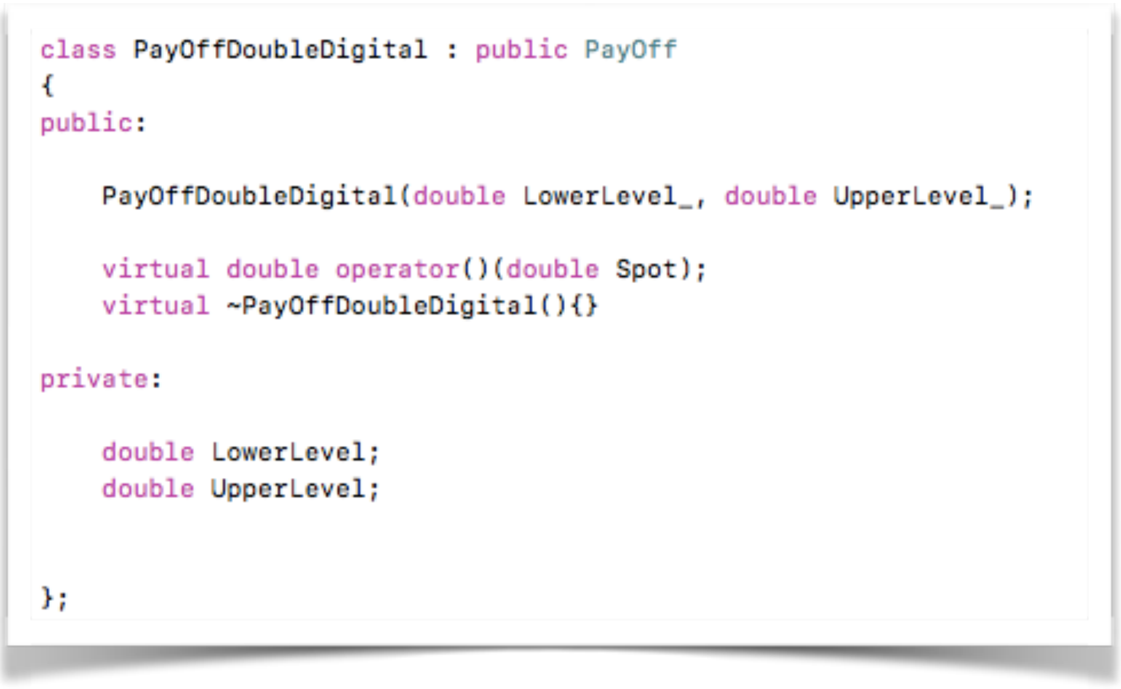
Below is the output for Both the types of Power options.
What I learn that with Call Power options your power value should be as high as possible eg 2,3,4 to maximise profit however with Put Power options the power value should be in fractions e.g. 0.2, 0.03 to get best retutrns

```
Enter expiry
1

Enter spot
100

Enter vol
0.6

r
0.0142

Number of paths
1000

enter 0 for call, 1 for put, 2 for Double Digital
enter 3 for Power call, 4 for Power put4

Enter strike for Power Put
90

 Enter the power
1/6

the price is 16.9172
Program ended with exit code: 0
```

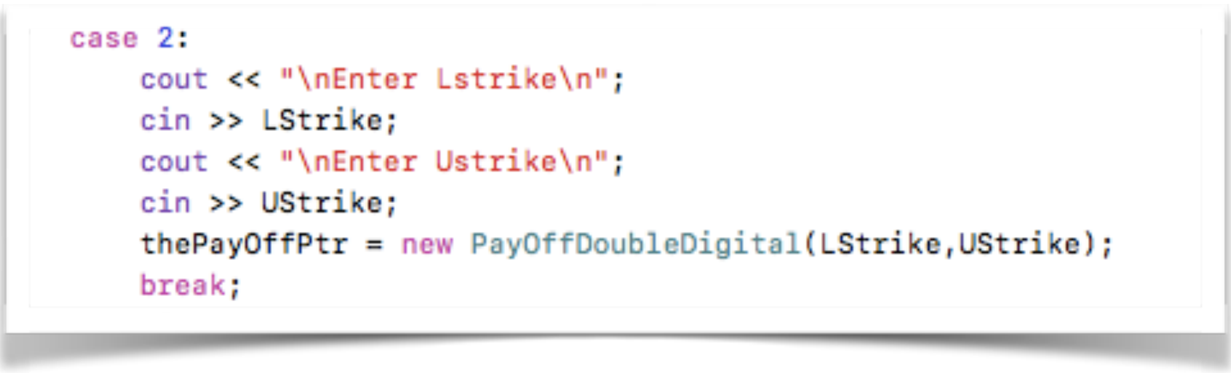
```
Enter expiry
1

Enter spot
100

Enter vol
0.6

r
0.0142

Number of paths
1000

enter 0 for call, 1 for put, 2 for Double Digital
enter 3 for Power call, 4 for Power put3

Enter strike for Power Call
110

 Enter the power
2

the price is 15710.5
```

Question 3.3)

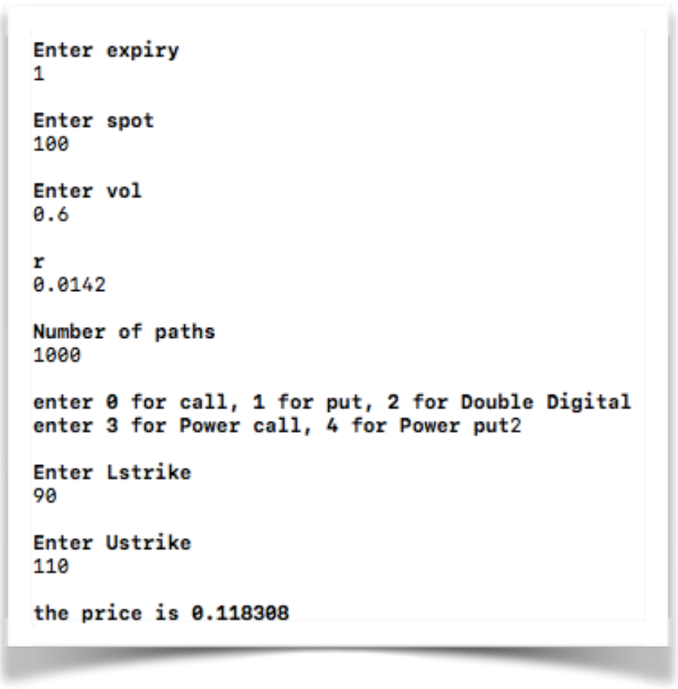
Double Digital Options



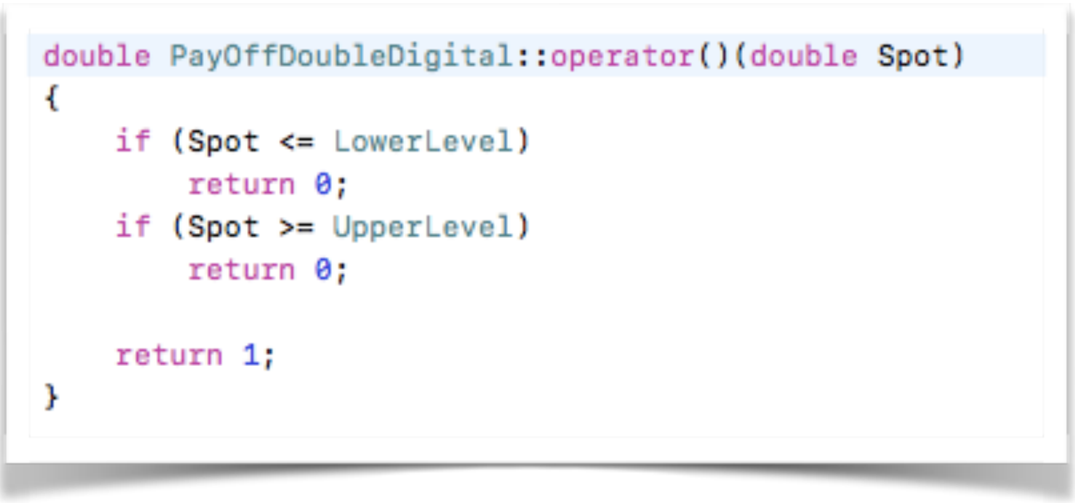
```
class PayOffDoubleDigital : public PayOff
{
public:

    PayOffDoubleDigital(double LowerLevel_, double UpperLevel_);

    virtual double operator()(double Spot);
    virtual ~PayOffDoubleDigital(){}

private:

    double LowerLevel;
    double UpperLevel;


};
```



```
case 2:
    cout << "\nEnter Lstrike\n";
    cin >> LStrike;
    cout << "\nEnter Ustrike\n";
    cin >> UStrike;
    thePayOffPtr = new PayOffDoubleDigital(LStrike,UStrike);
    break;
```



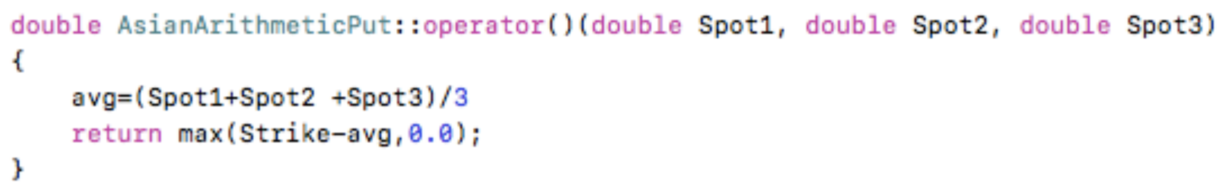
```
Enter expiry
1

Enter spot
100

Enter vol
0.6

r
0.0142

Number of paths
1000

enter 0 for call, 1 for put, 2 for Double Digital
enter 3 for Power call, 4 for Power put2

Enter Lstrike
90

Enter Ustrike
110

the price is 0.118308
```



```
double PayOffDoubleDigital::operator()(double Spot)
{
    if (Spot <= LowerLevel)
        return 0;
    if (Spot >= UpperLevel)
        return 0;

    return 1;
}
```

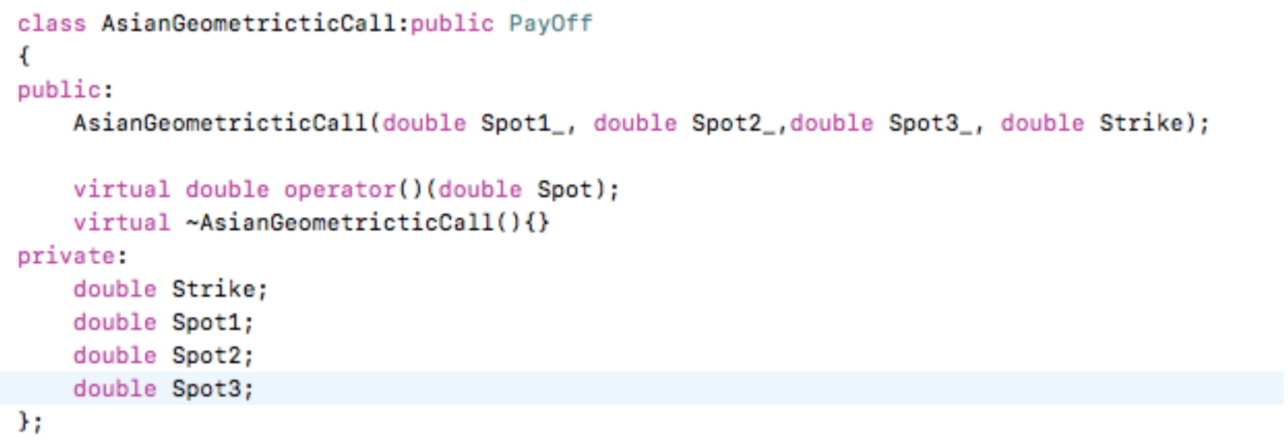
Arithmetic Average Asian Puts

Take N spot prices from user with N rates. Call the AsianArithmeticPut function with the spot prices calculate the average and return the max. Again this class is inherited from the PayOff class

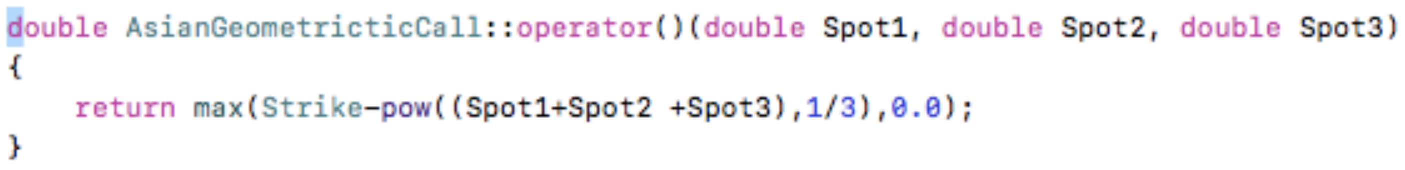


```
double AsianArithmeticPut::operator()(double Spot1, double Spot2, double Spot3)
{
    avg=(Spot1+Spot2 +Spot3)/3
    return max(Strike-avg,0.0);
}
```

Geometric Average Asian Calls



```
class AsianGeometricticCall:public PayOff
{
public:
    AsianGeometricticCall(double Spot1_, double Spot2_,double Spot3_, double Strike);

    virtual double operator()(double Spot);
    virtual ~AsianGeometricticCall(){}
private:
    double Strike;
    double Spot1;
    double Spot2;
    double Spot3;
};
```



```
double AsianGeometricticCall::operator()(double Spot1, double Spot2, double Spot3)
{
    return max(Strike-pow((Spot1+Spot2 +Spot3),1/3),0.0);
}
```

The first question includes implementation of double digital price options with inheritance. Only the files with the changes have been included this time in the code