1. Construct a namespace called "BS" (Black Scholes). It will include the following classes:

- A base class (Black-Scholes European)(BSEur) with the following members:
  - Constructors.
  - Destructor.
  - Parameters: \( S, K, r, \sigma, T \) and type of option.
  - Black Scholes price (Call and Put).
  - Function that displays the price on the screen.

- A derived class from BSEur (say MCarlo) with the following members:
  - Constructors.
  - Destructor.
  - Parameters: number of iterations.
  - MonteCarlo price (Call and Put).
  - Function that displays the price on the screen.

- A derived class from Mcarlo (say MCAnti) with the following members:
  - Constructors.
  - Destructor.
  - MonteCarlo price (using Antithetic variables).
  - Function that displays the price on the screen.

**Remark:** Test your code in C++; Use Armadillo for your random number generation; Don’t forget to define your derived-class constructors as inline functions.

Test your code with the following set-up:

\[
S=100, \ K=100, \ r=0.1, \ \sigma=0.25, \ T=1, \ N=50000; \\
type1 = "call" \\
type2 = "put"
\]

Here "N" is the number of iterations.

2. Construct a namespace called "POFF" (payoff) which includes:

- An abstract base class "Payoff" with at least the following members:
  - Constructor
  - Destructor
  - Copy Constructor
  - "=" Operator Overloading
- member function "virtual double payoff(double S)" with input Spot price S (pure virtual function).

- A derived class "CallPayoff" from "Payoff" with the following members:
  - Constructors
  - Destructor
  - Copy Constructor
  - "=" Operator Overloading
  - member function "payoff" Override
  - Private member "K" (Strike)
  - Public function "double Strike() const" which returns the value of "K"
  - Public function "void Strike (double NewStrike)" which sets the strike price

- A derived class "PutPayoff" from "Payoff" which has same structures as "CallPayoff".

- A derived class "BullSpreadPayoff" from "Payoff" which has the following members:
  - Constructors
  - Destructor
  - Copy Constructor
  - "=" Operator Overloading
  - member function "payoff" Override
  - Private member "K1" and "K2" (Strikes)
  - Protected member "costBuy" and "sellPrice" which represents the cost to buy the Call(K1) and the sell price of the Call(K2).
  - Public function "vector<double> Strike() const" which returns the value of "K1" and "K2"
  - Public function "void Strike (double NewStrike1, double NewStrike2)" which sets the strike prices
  - Public function "double profit (double S) const" which returns the profit of the Bull Spread

**Remark:**
A Bull Spread is a strategy that buys a call option on a stock with strike K1 and sells a call option on the same stock with strike K2 (K2 > K1).

Test your codes with a main function which at least contains the following code:

```cpp
int main()
{
    // some code
    CallPayoff call1;
    call1.Strike (20.0);
    cout << "Strike for call1 is" << call1.Strike() << endl;
```
CallPayoff call2(call1);
CallPayoff call3 = call2;

cout << "Give a stock price (plain Call): ";
double S;
cin >> S;
cout << "Call Payoff is: " << call3.payoff(S) << endl;

double K1 = 30.0; // Strike price of bought call
double K2 = 35.0; // Strike price of sell call
double costBuy = 3.0; // Cost to buy a call
double sellPrice = 1.0; // Sell price for call
BullSpreadPayoff bs(K1, K2, costBuy, sellPrice);
BullSpreadPayoff bs2(bs);
BullSpreadPayoff bs3 = bs2;
cout << "Give a stock price (BullSpread): ";
cin >> S;
cout << "Bull Spread Payoff is: " << bs3.payoff(S) << endl;
cout << "Bull Spread Profit is: " << bs3.profit(S) << endl;

return 0;