```
namespace LatticeModel
{

    internal class LatticeStrategy
    {

        internal static LatticeStrategy InitialiseStrategy(double Spot, int TimeSteps,
SDE sde)
        {
            if (Spot > 0 && TimeSteps > 0 && sde!=null)
            {
                return new LatticeStrategy(Spot, TimeSteps, sde);
            }
            else
                return null;
        }

        internal double[] SimulateUnderlying()
        {
            if (this!=null)
            {
                int numPrices = timeSteps + 1;
                double[] prices = new double[numPrices];
                int u = timeSteps;
                int d = 0;
                double upFactor = sde.GetUpFactor();
                double downFactor = sde.GetDownFactor();
                for (int i = 0; i < numPrices; i++)
                {
                    prices[i] = this.spot * System.Math.Pow(upFactor, u) *
System.Math.Pow(downFactor, d);
                    u--;
                    d++;
                }
                    return prices;

            }
            else
            {
                return null;
            }
        }

        private LatticeStrategy(double spot, int timeSteps, SDE sde)
        {
            this.spot = spot;
            this.timeSteps = timeSteps;
            this.sde = sde;
        }

        private int timeSteps;
        private double spot;
        private SDE sde;

    }
}
```