Problem Set 3
Numerical Methods for Option Pricing

- This homework consists of three questions, each carrying equal marks. Your code will be graded by a Python script which compares your result against the baseline result. It is therefore important that your Python stack is identical to the baseline so that you avoid losing marks even though your code is correct.
- Submit a zip compressed archive containing three modules, `BinomialAmerican.py`, `FiniteDifferenceEuropean.py` and `FiniteDifferenceAmerican.py` and a short report describing your implementation and results in PDF format.
- Be sure to include your name as part of the archive filename, and at the top of your report.

Problem 1. *Implement a binomial tree function for pricing American options in the Black-Scholes framework. The function interface is*

```
def binomialAmerican(S0, Ks, r, T, sigma, q, callputs, M)
```

where `Ks` *and* `callputs` *might be arrays, but all other inputs are scalars*[1]. `M` *is the number of timesteps in your tree(s).*

Problem 2. *Implement a finite difference scheme for pricing European options in the Black-Scholes framework. You should use an explicit finite difference scheme. The function interface is*

```
def fdEuropean(callput, S0, K, r, T, sigma, q,  M, N, S_max)
```

`M+1` *is the number of spatial steps in the uniform grid* `N+1` *is the number of time steps in the uniform grid S_{max} is the value where the upper spatial boundary condition is applied*

Problem 3. *Implement a finite difference scheme for pricing American options in the Black-Scholes framework. You should use the Gauss-Seidel method to solve a Crank-Nicolson formulation of the finite difference scheme. The function interface is*

```
def fdAmerican(callput, S0, K, r, T, sigma, q,  M, N, S_max)
```

`M+1` *is the number of spatial steps in the uniform grid* `N+1` *is the number of time steps in the uniform grid S_{max} is the value where the upper spatial boundary condition is applied*

[1]For example, your code should return a 3-element array of prices at t_0 if I call `binomialAmerican(40, [35,40,45], 0.05, 1.2, 0.4, 0.01, [1,-1,-1], 10)`