# Computational Finance - Mini Task 1

10134621

March 1st 2021

| $S$ | $d_1$ | $d_2$ | $\Pi(S, t = 0)$ |
|---|---|---|---|
| 1125 | -0.08390 | -0.09239 | 790.82 |
| 1200 | -0.06682 | -0.07532 | 918.43 |
| 1275 | -0.04991 | -0.05841 | 1048.24 |
| 1350 | -0.03313 | -0.04162 | 1180.34 |
| 1425 | -0.01642 | -0.02492 | 1314.81 |
| 1500 | 0.00024 | -0.00826 | 1451.72 |
| 1575 | 0.01690 | 0.00840 | 1591.17 |
| 1650 | 0.03360 | 0.02510 | 1733.25 |
| 1725 | 0.05038 | 0.04189 | 1878.05 |
| 1800 | 0.06730 | 0.05880 | 2025.69 |
| 1875 | 0.08437 | 0.07588 | 2176.28 |

Table 1: A table showing the share price, S, the numerical value of $d_1$ and $d_2$, to 5 d.p., and the value of the portfolio, $\Pi(S, t = 0)$, to 2 d.p.

Listing 1: C++ code for calculating portfolio values

```
// Header
// Student ID: 10134621
// File title: Mini task 1
// Date created: 24/02/21
// Last Edited: 26/02/21

#define _USE_MATH_DEFINES_

// Includes
#include<iostream>
#include<iomanip>
#include<cmath>
#include<math.h>
#include<chrono>
#include<vector>
```

```
// Declare functions

// calulcate d1
double d1(const double& S, const double& X, const double& T, const double& t, const double& r, const double& q, const double& sigma);

// calculate d2
double d2(const double& S, const double& X, const double& T, const double& t, const double& q, const double& sigma);

// calculate Pi portfolio
double Pi(const double& S, const double& X, const double& T, const double& t, const double& r, const double& q, const double& sigma,
    const double& d1, const double& d2);

// calculate cummulative normal distribution
double N(const double& x);


// Begin main program
int main()
{
    // define variables
    double T{ 1 };
    double X{ 1500 };
    double r{ 0.0319 };
    double q{ 0.0207 };
    double sigma{ 0.3153 };

    const double S[11] = { 1125, 1200, 1275,1350,1425,1500,1575,1650,1725,1800,1875 }; // input S data
    double t = 0;  // set time
    std::vector<double> pi;  // vector for pi values
    std::vector<double> d1_store;  // vector for d1
    std::vector<double> d2_store;  // vector for d2

    // get start time
    auto start = std::chrono::steady_clock::now();

    // for loop over all S values
```

```cpp
    for (int i{ 0 }; i < sizeof(S)/sizeof(S[0]); i++) {
        d1_store.push_back(d1(S[i], X, T, t, r, q, sigma));
        d2_store.push_back(d2(S[i], X, T, t, q, sigma));
        pi.push_back(Pi(S[i], X, T, t, r, q, sigma, d1_store[i], d2_store[i]));
    }

    // output results
    std::cout << std::setprecision(10);
    for (int i{ 0 }; i < sizeof(S) / sizeof(S[0]); i++) {
        std::cout << "S = " << S[i] << ", d1 = " << d1_store[i] << ", d2 = " << d2_store[i] << ", Pi(S, 0) = " << pi[i] << std::endl;
    }

    // end timer
    auto finish = std::chrono::steady_clock::now();

    // convert into real time in seconds
    auto elapsed = std::chrono::duration_cast<std::chrono::duration<double>> (finish - start);

    // output the time
    std::cout << "Elapsied time: " << elapsed.count() << std::endl;

    return 0;
}  // End of main program


// Function definitions

// calculate d1
double d1(const double& S, const double& X, const double& T, const double& t, const double& r, const double& q, const double& sigma)
{
    return (sinh((S / X) - 1) + r * (T - t) * exp(1 - (pow(sigma, 2) / q))) / (exp(1 + pow(sigma, 2) * (T - t)));
}

// calculate d2
double d2(const double& S, const double& X, const double& T, const double& t, const double& q, const double&
    sigma)
{
    return (sinh((S / X) - 1) - sigma * sin(pow(sigma, 2) - q) * pow(T - t, 0.5)) / (exp(1 + pow(sigma, 2) * (T - t)));
}

// calculate cummulative normal distribution
double N(const double& x)
{
    return 0.5 * erfc(-x / pow(2, 0.5));
}

// calculate portfolio value
double Pi(const double& S, const double& X, const double& T, const double& t, const double& r, const double& q, const double& sigma,
    const double& d1, const double& d2)
{
    return S * exp(1 + pow(sigma, 2) * (T - t)) * exp(-r * (T - t)) * N(d1) - pow(pow(X, 1 + (r / q)) * pow(S, 1 - (r / q)), 0.5) * exp(-q * (T - t)) * N(d2);
}
```