# Technical Interview

The following exercises are meant to be discussed between the interviewer and the interviewee.

## Algorithms

### Exercise 1

Write a method to receive the row and column (assume the board starts at 1,1) of an infinite chess board and return a Boolean value indicating whether the color of the cell is black or white.

Example:

IsWhite(Row:1, Column:1) returns NO

IsWhite(Row:1, Column:2) returns YES

IsWhite(Row:999, Column:888) returns YES

### Exercise 2

Given two strings, write a method to decide if they are anagrams of each other.

### Exercise 3

In a faraway country, there is a single farm which contains apples.

The farm is surrounded by three concentric fences, each one of them guarded by a guard.

Mister XY wished to get through the fences and eat ONE apple so he went to the first guard and reached an agreement:

He would give the guard half the apples he would bring back plus half an additional apple.

The guard agreed, indicating that he only wanted full apples (e.g. if he brought back three apples, one and half wouldn't do for the guard, so Mister XY would give two to the guard and keep one for himself). Consider that giving two of fours apples to the guard won't be valid, because it doesn't take into account the additional half.

Mr XY reached the same agreement with the remaining two guards.

Consequently, how many apples must Mr XY cut to fulfill the agreements and have just the one apple he wishes when leaving the fences?

How would you generalize this algorithm in a method that returns the number of apples when provided with the number of fences?

double ApplesByFences(int fences){...}

# Simplification

## Exercise 1

How would you make the following method more readable?

```
double GetPaymentAmount()
{
        double result;
        if (_isDead) result = GetDeadAmount();
        else {
                if (_isSeparated) result = GetSeparatedAmount();
                else {
                        if (_isRetired) result = GetRetiredAmount();
                        else result = GetNormalPayAmount();
                };
        };
        return result;
};
```

## Exercise 2

Is the following code easily maintainable? How would you improve it?

```
public class Product
{
        ...
        public decimal Price()
        {
                if (UOM.Equals("grams"))
                return Quantity * 6m / 1000;

                if (UOM.Equals("bottle"))
                return Quantity * 3m;

                if (UOM.Equals("bag"))
                {
                        var total += Quantity * .2m;
```

```
                    var setsOfFour = Quantity / 4;
                    total -= setsOfFour * .15m; //discount on groups of 4 items
                    return total;
                }

                return 0m;
        }
}
```

## Exercise 3

How would you improve the maintainability of the following code?

```
public class Employee
{
        private int _type;
        static final int ENGINEER = 0;
        static final int SALESMAN = 1;
        static final int MANAGER = 2;

        public double MonthlySalary {get; set;}
        public double Commission {get; set;}
        public double Bonus {get; set;}

        public Employee (int type)  { _type = type; }

        public int GetPaymentAmount()
        {
                switch (_type) {
                default:
                case 0:
                        return MonthlySalary;
                case 1:
                        return MonthlySalary + Commission;
                case 2:
                        return MonthlySalary + Bonus;
                }
        }
}
```