8 Prolog (ACR)

In this question you should ensure that your predicates behave appropriately with backtracking. You **may not** make use of extra-logical built-in predicates such as `findAll`. Use of the cut operator is permitted unless specified otherwise. You may ignore the possibility of overflow or division by zero.

(*a*) A term can either be an *atom*, *variable* or a *compound term*. Define each of these. [3 marks]

(*b*) Euclid's algorithm for computing the greatest common divisor of two integers can be implemented in ML as:

```
fun gcd(a,0) = a
 |  gcd(a,b) = gcd(b, a mod b);
```

Provide an implementation in Prolog without using the cut operator. [4 marks]

(*c*) We can represent fractions using the compound term `div/2`. For example `div(1,3)` represents $\frac{1}{3}$.

Implement a predicate `simplify` which transforms a fraction into its smallest exact representation. For example, `simplify(div(8,4),B)` should unify `B` with `2`, and `simplify(div(4,8),A)` should unify `A` with `div(1,2)`. Your predicate should avoid unnecessary computation. [5 marks]

(*d*) We can also represent arithmetic expressions involving addition, subtraction, multiplication and division. For example, the expression $3\frac{5}{2-1}+4$ is represented as `add(mul(3,div(5,sub(2,1))),4)`.

Implement a predicate `reduce` which reduces an arithmetic expression to its smallest exact representation e.g. `reduce(add(div(1,2),div(1,4)),A)` should unify `A` with `div(3,4)`. [8 marks]