CS3011 Symbolic Programming
First of Two Assessed Labs, due Oct 31[1]

Recall from **LPN**, Chapter 3

```
numeral(0).
numeral(s(X)) :- numeral(X).

add(0,X,X).
add(s(X),Y,s(Z)) :- add(X,Y,Z).
```

Exercise 1

Suppose we were to extend the knowledge base above with the clause

```
numeral(X+Y) :- numeral(X), numeral(Y).
```

Define a predicate `add2(X,Y,Z)` such that for instance,

```
?- add2(s(0)+s(s(0)), s(s(0)), Z).
Z = s(s(s(s(s(0)))))

?- add2(0, s(0)+s(s(0)), Z).
Z = s(s(s(0)))

?- add2(s(s(0)), s(0)+s(s(0)), Z).
Z = s(s(s(s(s(0)))))

?- add2(s(0)+s(0), s(0+s(s(0))), Z).
Z = s(s(s(s(s(0)))))
```

etc.

Exercise 2

Next we introduce negative numbers via the function symbol `p` (for predecessor or -1, just as `s` stands for successor or $+1$).

```
numeral(p(X)) :- numeral(X).
```

Extend the predicate `add2` such that for instance,

```
?- add2(p(s(0)), s(s(0)), Z).
Z = s(s(0))

?- add2(0, s(p(0)), Z).
Z = 0

?- add2(p(0)+s(s(0)),s(s(0)),Z).
Z = s(s(s(0)))

?- add2(p(0), p(0)+s(p(0)), Z).
Z = p(p(0))
```

[1] Submit to Blackboard. For any extension, email your request to your demonstrator, David Woods (dwoods@tcd.ie).

Exercise 3

Define a predicate `minus(X,Y)` such that for instance,

```
?- minus(0, Z).
Z = 0

?- minus(s(s(0)), Z).
Z = p(p(0))

?- minus(s(p(0)), Z).
Z = 0

?- minus(p(s(p(0))), Z).
Z = s(0)
```

Exercise 4

Let us extend `numeral` further to

```
numeral(-X) :- numeral(X).
```

Revise the predicate `add2(X,Y,Z)` such that for instance,

```
?- add2(-p(s(0)), s(s(0)), Z).
Z = s(s(0))

?- add2(p(0)+s(s(0)), -s(s(0)), Z).
Z = p(0)
```

Exercise 5

Define the predicate `subtract(X,Y,Z)` for subtracting `Y` from `X` to get `Z` such that for instance,

```
?- subtract(p(s(0)), s(s(0)), Z).
Z = p(p(0))

?- subtract(p(0), -s(s(0)), Z).
Z = s(0)
```

Exercise 6

Extend the predicates `add2` and `subtract` to handle the new rule

```
numeral(X-Y) :- numeral(X), numeral(Y).
```

For instance,

```
?- add2(-(s(0)-p(0)),s(0),X).
X = p(0)

?- subtract(p(0), p(s(0))-s(s(0)), Z).
Z = s(0)
```