Lemma

On the set of natural numbers, addition is associative. In other words, for all natural numbers a, b and c, we have

$$(a+b)+c=a+(b+c).$$

```
lemma add_assoc (a b c : mynat) : (a + b) + c = a + (b + c) :=
```

Proof :

```
begin
41 induction c with d hd,
42 induction b with d hd,
43 induction a with d hd,
```

44:0: goal

```
4 goals
⊢ 0 + 0 + 0 = 0 + (0 + 0)

case mynat.succ
d : mynat,
hd : d + 0 + 0 = d + (0 + 0)
⊢ succ d + 0 + 0 = succ d + (0 + 0)

case mynat.succ
a d : mynat,
hd : a + d + 0 = a + (d + 0)
⊢ a + succ d + 0 = a + (succ d + 0)

case mynat.succ
a b d : mynat,
hd : a + b + d = a + (b + d)
⊢ a + b + succ d = a + (b + succ d)
```

Proof :

```
begin
41 induction a with d hd,
42
43
```

42:0: goal

```
2 goals
b c : mynat
⊢ 0 + b + c = 0 + (b + c)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

Proof:

```
begin
induction a with d hd,
rw zero_add,

```

43:0: goal

```
2 goals
b c : mynat
⊢ b + c = 0 + (b + c)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

.

Proof:

```
begin
induction a with d hd,
rw zero_add,
induction b with d hd,
rw zero_add,

```

45:0: goal

```
3 goals
c : mynat
⊢ c = 0 + c

case mynat.succ
c d : mynat,
hd : d + c = 0 + (d + c)
⊢ succ d + c = 0 + (succ d + c)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

Proof :

```
begin
41 induction a with d hd,
42 rw zero_add,
43 induction b with d hd,
44 rw zero_add,
45 rw zero_add,
46 refl,
```

46:5: goal

```
2 goals
case mynat.succ
c d : mynat,
hd : d + c = 0 + (d + c)
⊢ succ d + c = 0 + (succ d + c)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

Proof :

```
begin
41 induction a with d hd,
42 rw zero_add,
43 induction b with d hd,
44 rw zero_add,
45 rw zero_add,
46 refl,
47 induction c with d hd,
48 rw add_zero,
```

48:0: goal

```
3 goals
d : mynat,
hd : d + 0 = 0 + (d + 0)
⊢ succ d + 0 = 0 + (succ d + 0)

case mynat.succ
d d : mynat,
hd : d + d = 0 + (d + d) → succ d + d = 0 + (succ d + d),
hd : d + succ d = 0 + (d + succ d)
⊢ succ d + succ d = 0 + (succ d + succ d)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

Proof :

```
begin
  induction a with d hd,
  rw zero_add,
  induction b with d hd,
  rw zero_add,
  rw zero_add,
  refl,
  induction c with d hd,
  rw add_zero,
```

49:0: goal

```
3 goals
d : mynat,
hd : d + 0 = 0 + (d + 0)
⊢ succ d = 0 + succ d

case mynat.succ
d d : mynat,
hd : d + d = 0 + (d + d) → succ d + d = 0 + (succ d + d),
hd : d + succ d = 0 + (d + succ d)
⊢ succ d + succ d = 0 + (succ d + succ d)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

Proof :

```
begin
  induction a with d hd,
  rw zero_add,
  induction b with d hd,
  rw zero_add,
  rw zero_add,
  refl,
  induction c with d hd,
  rw add_zero,
  rw add_succ,

```

50:0: goal

```
3 goals
d : mynat,
hd : d + 0 = 0 + (d + 0)
⊢ succ d = succ (0 + d)

case mynat.succ
d d : mynat,
hd : d + d = 0 + (d + d) → succ d + d = 0 + (succ d + d),
hd : d + succ d = 0 + (d + succ d)
⊢ succ d + succ d = 0 + (succ d + succ d)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

Proof :

```
begin
  induction a with d hd,
  rw zero_add,
  induction b with d hd,
  rw zero_add,
  rw zero_add,
  refl,
  induction c with d hd,
  rw add_zero,
  rw add_succ,
  rw zero_add,
  refl,
```

50:0: goal

```
3 goals
d : mynat,
hd : d + 0 = 0 + (d + 0)
⊢ succ d = succ (0 + d)

case mynat.succ
d d : mynat,
hd : d + d = 0 + (d + d) → succ d + d = 0 + (succ d + d),
hd : d + succ d = 0 + (d + succ d)
⊢ succ d + succ d = 0 + (succ d + succ d)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

51:5: goal

```
2 goals
case mynat.succ
d d : mynat,
hd : d + d = 0 + (d + d) → succ d + d = 0 + (succ d + d),
hd : d + succ d = 0 + (d + succ d)
⊢ succ d + succ d = 0 + (succ d + succ d)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

.

Proof :

```
begin
41 induction a with d hd,
42 rw zero_add,
43 induction b with d hd,
44 rw zero_add,
45 rw zero_add,
46 refl,
47 induction c with d hd,
48 rw add_zero,
49 rw add_succ,
50 rw zero_add,
51 refl,
52 rw add_succ,
```

53:0: goal

```
2 goals
case mynat.succ
d d : mynat,
hd : d + d = 0 + (d + d) → succ d + d = 0 + (succ d + d),
hd : d + succ d = 0 + (d + succ d)
⊢ succ (succ d + d) = 0 + succ (succ d + d)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

Proof :

```
begin
 42 rw zero_add,
 43 induction b with d hd,
 44 rw zero_add,
 45 rw zero_add,
 46 refl,
 47 induction c with d hd,
 48 rw add_zero,
 49 rw add_succ,
 50 rw zero_add,
 51 refl,
 52 rw add_succ,
 53 rw zero_add,
```

54:0: goal

```
2 goals
case mynat.succ
d d : mynat,
hd : d + d = 0 + (d + d) → succ d + d = 0 + (succ d + d),
hd : d + succ d = 0 + (d + succ d)
⊢ succ (succ d + d) = succ (succ d + d)

case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

Proof :

```
begin
  induction b with d hd,
  rw zero_add,
  rw zero_add,
  refl,
  induction c with d hd,
  rw add_zero,
  rw add_succ,
  rw zero_add,
  refl,
  rw add_succ,
  rw zero_add,
  refl,
```

55:0: goal

```
case mynat.succ
b c d : mynat,
hd : d + b + c = d + (b + c)
⊢ succ d + b + c = succ d + (b + c)
```

```
begin
  rw zero_add,
  rw zero_add,
  refl,
  induction c with d hd,
  rw add_zero,
  rw add_succ,
  rw zero_add,
  refl,
  rw add_succ,
  rw zero_add,
  refl,
  induction b with d hd,
```

56:0: goal

```
2 goals
c d : mynat,
hd : d + 0 + c = d + (0 + c)
⊢ succ d + 0 + c = succ d + (0 + c)

case mynat.succ
c d d : mynat,
hd : d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c),
hd : d + succ d + c = d + (succ d + c)
⊢ succ d + succ d + c = succ d + (succ d + c)
```

Proof :

```
begin
44 rw zero_add,
45 rw zero_add,
46 refl,
47 induction c with d hd,
48 rw add_zero,
49 rw add_succ,
50 rw zero_add,
51 refl,
52 rw add_succ,
53 rw zero_add,
54 refl,
55 induction b with d hd,
56 rw zero_add,
57 rw add_zero,
```

57:12: goal

```
2 goals
c d : mynat,
hd : d + 0 + c = d + (0 + c)
⊢ succ d + c = succ d + c

case mynat.succ
c d d : mynat,
hd : d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c),
hd : d + succ d + c = d + (succ d + c)
⊢ succ d + succ d + c = succ d + (succ d + c)
```

Proof:

```
begin
46 refl,
47 induction c with d hd,
48 rw add_zero,
49 rw add_succ,
50 rw zero_add,
51 refl,
52 rw add_succ,
53 rw zero_add,
54 refl,
55 induction b with d hd,
56 rw zero_add,
57 rw add_zero,
58 refl,
```

59:0: goal

```
case mynat.succ
c d d : mynat,
hd : d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c),
hd : d + succ d + c = d + (succ d + c)
⊢ succ d + succ d + c = succ d + (succ d + c)
```

Proof:

```
begin
47 induction c with d hd,
48 rw add_zero,
49 rw add_succ,
50 rw zero_add,
51 refl,
52 rw add_succ,
53 rw zero_add,
54 refl,
55 induction b with d hd,
56 rw zero_add,
57 rw add_zero,
58 refl,
59 rw add_succ,
```

60:0: goal

```
case mynat.succ
c d d : mynat,
hd : d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c),
hd : d + succ d + c = d + (succ d + c)
⊢ succ (succ d + d) + c = succ d + (succ d + c)
```

Proof :

```
begin
48 rw add_zero,
49 rw add_succ,
50 rw zero_add,
51 refl,
52 rw add_succ,
53 rw zero_add,
54 refl,
55 induction b with d hd,
56 rw zero_add,
57 rw add_zero,
58 refl,
59 rw add_succ,
60 induction d with d hd,
```

61:0: goal

```
2 goals
c d : mynat,
hd : d + 0 + c = d + (0 + c) → succ d + 0 + c = succ d + (0 + c),
hd_1 : d + succ 0 + c = d + (succ 0 + c)
⊢ succ (succ d + 0) + c = succ d + (succ 0 + c)

case mynat.succ
c d d : mynat,
hd :
  (d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c)) →
  d + succ d + c = d + (succ d + c) → succ (succ d + d) + c = succ d + (succ d + c),
hd : d + succ d + c = d + (succ d + c) → succ d + succ d + c = succ d + (succ d + c),
hd : d + succ (succ d) + c = d + (succ (succ d) + c)
⊢ succ (succ d + succ d) + c = succ d + (succ (succ d) + c)
```

```
begin
  rw add_succ,
  rw zero_add,
  refl,
  rw add_succ,
  rw zero_add,
  refl,
  induction b with d hd,
  rw zero_add,
  rw add_zero,
  refl,
  rw add_succ,
  induction d with d hd,
  rw add_zero,
```

```
2 goals
c d : mynat,
hd : d + 0 + c = d + (0 + c) → succ d + 0 + c = succ d + (0 + c),
hd_1 : d + succ 0 + c = d + (succ 0 + c)
⊢ succ (succ d) + c = succ d + (succ 0 + c)

case mynat.succ
c d d : mynat,
hd :
  (d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c)) →
  d + succ d + c = d + (succ d + c) → succ (succ d + d) + c = succ d + (succ d + c),
hd : d + succ d + c = d + (succ d + c) → succ d + succ d + c = succ d + (succ d + c),
hd : d + succ (succ d) + c = d + (succ (succ d) + c)
⊢ succ (succ d + succ d) + c = succ d + (succ (succ d) + c)
```

Proof:

```
begin
  rw zero_add,
  refl,
  rw add_succ,
  rw zero_add,
  refl,
  induction b with d hd,
  rw zero_add,
  rw add_zero,
  refl,
  rw add_succ,
  induction d with d hd,
  rw add_zero,
  induction c with d hd,
```

63:0: goal

```
3 goals
d : mynat,
hd : d + 0 + 0 = d + (0 + 0) → succ d + 0 + 0 = succ d + (0 + 0),
hd_1 : d + succ 0 + 0 = d + (succ 0 + 0)
⊢ succ (succ d) + 0 = succ d + (succ 0 + 0)

case mynat.succ
d d : mynat,
hd :
  (d + 0 + d = d + (0 + d) → succ d + 0 + d = succ d + (0 + d)) →
  d + succ 0 + d = d + (succ 0 + d) → succ (succ d) + d = succ d + (succ 0 + d),
hd : d + 0 + succ d = d + (0 + succ d) → succ d + 0 + succ d = succ d + (0 + succ d),
hd_1 : d + succ 0 + succ d = d + (succ 0 + succ d)
⊢ succ (succ d) + succ d = succ d + (succ 0 + succ d)

case mynat.succ
c d d : mynat,
hd :
  (d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c)) →
  d + succ d + c = d + (succ d + c) → succ (succ d + d) + c = succ d + (succ d + c),
hd : d + succ d + c = d + (succ d + c) → succ d + succ d + c = succ d + (succ d + c),
hd : d + succ (succ d) + c = d + (succ (succ d) + c)
⊢ succ (succ d + succ d) + c = succ d + (succ (succ d) + c)
```

Proof:

```
begin
51 refl,
52 rw add_succ,
53 rw zero_add,
54 refl,
55 induction b with d hd,
56 rw zero_add,
57 rw add_zero,
58 refl,
59 rw add_succ,
60 induction d with d hd,
61 rw add_zero,
62 induction c with d hd,
63 rw add_zero,
```

64:0: goal

```
3 goals
d : mynat,
hd : d + 0 + 0 = d + (0 + 0) → succ d + 0 + 0 = succ d + (0 + 0),
hd_1 : d + succ 0 + 0 = d + (succ 0 + 0)
⊢ succ (succ d) = succ d + (succ 0 + 0)

case mynat.succ
d d : mynat,
hd :
  (d + 0 + d = d + (0 + d) → succ d + 0 + d = succ d + (0 + d)) →
  d + succ 0 + d = d + (succ 0 + d) → succ (succ d) + d = succ d + (succ 0 + d),
hd : d + 0 + succ d = d + (0 + succ d) → succ d + 0 + succ d = succ d + (0 + succ d),
hd_1 : d + succ 0 + succ d = d + (succ 0 + succ d)
⊢ succ (succ d) + succ d = succ d + (succ 0 + succ d)

case mynat.succ
c d d : mynat,
hd :
  (d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c)) →
  d + succ d + c = d + (succ d + c) → succ (succ d + d) + c = succ d + (succ d + c),
hd : d + succ d + c = d + (succ d + c) → succ d + succ d + c = succ d + (succ d + c),
hd : d + succ (succ d) + c = d + (succ (succ d) + c)
⊢ succ (succ d + succ d) + c = succ d + (succ (succ d) + c)
```

```
begin
  52 rw add_succ,
  53 rw zero_add,
  54 refl,
  55 induction b with d hd,
  56 rw zero_add,
  57 rw add_zero,
  58 refl,
  59 rw add_succ,
  60 induction d with d hd,
  61 rw add_zero,
  62 induction c with d hd,
  63 rw add_zero,
  64 rw add_zero,
```

65:0: goal

```
3 goals
d : mynat,
hd : d + 0 + 0 = d + (0 + 0) → succ d + 0 + 0 = succ d + (0 + 0),
hd_1 : d + succ 0 + 0 = d + (succ 0 + 0)
⊢ succ (succ d) = succ d + succ 0

case mynat.succ
d d : mynat,
hd :
  (d + 0 + d = d + (0 + d) → succ d + 0 + d = succ d + (0 + d)) →
  d + succ 0 + d = d + (succ 0 + d) → succ (succ d) + d = succ d + (succ 0 + d),
hd : d + 0 + succ d = d + (0 + succ d) → succ d + 0 + succ d = succ d + (0 + succ d),
hd_1 : d + succ 0 + succ d = d + (succ 0 + succ d)
⊢ succ (succ d) + succ d = succ d + (succ 0 + succ d)

case mynat.succ
c d d : mynat,
hd :
  (d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c)) →
  d + succ d + c = d + (succ d + c) → succ (succ d + d) + c = succ d + (succ d + c),
hd : d + succ d + c = d + (succ d + c) → succ d + succ d + c = succ d + (succ d + c),
hd : d + succ (succ d) + c = d + (succ (succ d) + c)
⊢ succ (succ d + succ d) + c = succ d + (succ (succ d) + c)
```

```
begin
55 induction b with d hd,
56 rw zero_add,
57 rw add_zero,
58 refl,
59 rw add_succ,
60 induction d with d hd,
61 rw add_zero,
62 induction c with d hd,
63 rw add_zero,
64 rw add_zero,
65 rw add_succ,
66 rw add_zero,
67 refl,
```

68:0: goal

```
2 goals
case mynat.succ
d d : mynat,
hd :
  (d + 0 + d = d + (0 + d) → succ d + 0 + d = succ d + (0 + d)) →
  d + succ 0 + d = d + (succ 0 + d) → succ (succ d) + d = succ d + (succ 0 + d),
hd : d + 0 + succ d = d + (0 + succ d) → succ d + 0 + succ d = succ d + (0 + succ d),
hd_1 : d + succ 0 + succ d = d + (succ 0 + succ d)
⊢ succ (succ d) + succ d = succ d + (succ 0 + succ d)

case mynat.succ
c d d : mynat,
hd :
  (d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c)) →
  d + succ d + c = d + (succ d + c) → succ (succ d + d) + c = succ d + (succ d + c),
hd : d + succ d + c = d + (succ d + c) → succ d + succ d + c = succ d + (succ d + c),
hd : d + succ (succ d) + c = d + (succ (succ d) + c)
⊢ succ (succ d + succ d) + c = succ d + (succ (succ d) + c)
```

```
begin
56 rw zero_add,
57 rw add_zero,
58 refl,
59 rw add_succ,
60 induction d with d hd,
61 rw add_zero,
62 induction c with d hd,
63 rw add_zero,
64 rw add_zero,
65 rw add_succ,
66 rw add_zero,
67 refl,
68 rw add_succ,
69 rw add_succ,
```

68:0: goal

```
2 goals
case mynat.succ
d d : mynat,
hd :
  (d + 0 + d = d + (0 + d) → succ d + 0 + d = succ d + (0 + d)) →
  d + succ 0 + d = d + (succ 0 + d) → succ (succ d) + d = succ d + (succ 0 + d),
hd : d + 0 + succ d = d + (0 + succ d) → succ d + 0 + succ d = succ d + (0 + succ d),
hd_1 : d + succ 0 + succ d = d + (succ 0 + succ d)
⊢ succ (succ d) + succ d = succ d + (succ 0 + succ d)

case mynat.succ
c d d : mynat,
hd :
  (d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c)) →
  d + succ d + c = d + (succ d + c) → succ (succ d + d) + c = succ d + (succ d + c),
hd : d + succ d + c = d + (succ d + c) → succ d + succ d + c = succ d + (succ d + c),
hd : d + succ (succ d) + c = d + (succ (succ d) + c)
⊢ succ (succ d + succ d) + c = succ d + (succ (succ d) + c)
```

```
begin
58 refl,
59 rw add_succ,
60 induction d with d hd,
61 rw add_zero,
62 induction c with d hd,
63 rw add_zero,
64 rw add_zero,
65 rw add_succ,
66 rw add_zero,
67 refl,
68 rw add_succ,
69 rw add_succ,
70 rw add_succ,
```

71:0: goal

```
2 goals
case mynat.succ
d d : mynat,
hd :
  (d + 0 + d = d + (0 + d) → succ d + 0 + d = succ d + (0 + d)) →
  d + succ 0 + d = d + (succ 0 + d) → succ (succ d) + d = succ d + (succ 0 + d),
hd : d + 0 + succ d = d + (0 + succ d) → succ d + 0 + succ d = succ d + (0 + succ d),
hd_1 : d + succ 0 + succ d = d + (succ 0 + succ d)
⊢ succ (succ (succ d) + d) = succ (succ d + (succ 0 + d))

case mynat.succ
c d d : mynat,
hd :
  (d + d + c = d + (d + c) → succ d + d + c = succ d + (d + c)) →
  d + succ d + c = d + (succ d + c) → succ (succ d + d) + c = succ d + (succ d + c),
hd : d + succ d + c = d + (succ d + c) → succ d + succ d + c = succ d + (succ d + c),
hd : d + succ (succ d) + c = d + (succ (succ d) + c)
⊢ succ (succ d + succ d) + c = succ d + (succ (succ d) + c)
```

```
begin
58 refl,
59 rw add_succ,
60 induction d with d hd,
61 rw add_zero,
62 induction c with d hd,
63 rw add_zero,
64 rw add_zero,
65 rw add_succ,
66 rw add_zero,
67 refl,
68 rw add_succ,
69 rw add_succ,
70 rw add_succ,
71 rw add_succ,
```

71:12: goal

```
case mynat.zero
b c : mynat
⊢ zero + b + c = zero + (b + c)
```

```
induction a with d hd,
rw zero_add,
rw zero_add,
refl,
induction b with d hd,
rw zero_add,
rw add_zero,
refl,
rw add_succ,
induction c with d hd,
rw add_zero,
rw add_zero,
rw add_succ,
refl,
rw add_succ,
rw add_succ,
rw add_succ,
induction d with d hd,
rw add_zero,
rw add_zero,
rw add_succ,
refl,
rw add_succ,
rw add_succ,
rw add_succ,
induction d with d hd,
rw add_zero,
rw add_zero,
rw add_succ,
refl,
rw add_succ,
rw add_succ,
rw add_succ,
induction d with d hd,
rw add_zero,
rw add_zero,
rw add_succ,
refl,
rw add_succ,
rw add_succ,
rw add_succ,
```

81:12: goal

```
case mynat.succ
d d d : mynat,
hd :
((((d + d + d = d + (d + d) → succ d + d + d = succ d + (d + d)) →
d + succ d + d = d + (succ d + d) → succ (succ d + d) + d = succ d +
(succ d + d)) →
(d + d + succ d = d + (d + succ d) → succ d + d + succ d = succ d +
(d + succ d)) →
d + succ d + succ d = d + (succ d + succ d) → succ (succ (succ d +
d) + d) = succ (succ d + (succ d + d))) →
((d + d + succ d = d + (d + succ d) → succ d + d + succ d = succ d +
(d + succ d)) →
d + succ d + succ d = d + (succ d + succ d) → succ (succ d + d) +
succ d = succ d + (succ d + succ d)) →
(d + d + succ (succ d) = d + (d + succ (succ d)) → succ d + d + succ
(succ d) = succ d + (d + succ (succ d))) →
d + succ d + succ (succ d) = d + (succ d + succ (succ d)) →
succ (succ (succ (succ d + d) + d)) = succ (succ (succ d + (succ d +
d)))) →
(((d + d + succ d = d + (d + succ d) → succ d + d + succ d = succ d
+ (d + succ d)) →
d + succ d + succ d = d + (succ d + succ d) → succ (succ d + d) +
succ d = succ d + (succ d + succ d)) →
(d + d + succ (succ d) = d + (d + succ (succ d)) → succ d + d + succ
(succ d) = succ d + (d + succ (succ d))) →
d + succ d + succ (succ d) = d + (succ d + succ (succ d)) →
succ (succ (succ d + d) + succ d) = succ (succ d + (succ d + succ
d))) →
((d + d + succ (succ d) = d + (d + succ (succ d)) → succ d + d +
succ (succ d) = succ d + (d + succ (succ d))) →
d + succ d + succ (succ d) = d + (succ d + succ (succ d)) →
succ (succ d + d) + succ (succ d) = succ d + (succ d + succ (succ
d))) →
(d + d + succ (succ (succ d)) = d + (d + succ (succ (succ d))) →
succ d + d + succ (succ (succ d)) = succ d + (d + succ (succ (succ
d)))) →
d + succ d + succ (succ (succ d)) = d + (succ d + succ (succ (succ
d))) →
succ (succ (succ (succ (succ d + d) + d))) = succ (succ (succ (succ
d + (succ d + d)))),
hd :
(((d + d + succ d = d + (d + succ d) → succ d + d + succ d = succ d
+ (d + succ d)) →
d + succ d + succ d = d + (succ d + succ d) → succ (succ d + d) +
succ d = succ d + (succ d + succ d)) →
(d + d + succ (succ d) = d + (d + succ (succ d)) → succ d + d + succ
(succ d) = succ d + (d + succ (succ d))) →
d + succ d + succ (succ d) = d + (succ d + succ (succ d)) →
succ (succ (succ d + d) + succ d) = succ (succ d + (succ d + succ
d))) →
((d + d + succ (succ d) = d + (d + succ (succ d)) → succ d + d +
succ (succ d) = succ d + (d + succ (succ d))) →
d + succ d + succ (succ d) = d + (succ d + succ (succ d)) →
```

```
succ (succ d + d) + succ (succ d) = succ d + (succ d + succ (succ
d))) →
(d + d + succ (succ (succ d)) = d + (d + succ (succ (succ d))) →
succ d + d + succ (succ (succ d)) = succ d + (d + succ (succ (succ
d)))) →
d + succ d + succ (succ (succ d)) = d + (succ d + succ (succ (succ
d))) →
succ (succ (succ (succ d + d) + succ d)) = succ (succ (succ d +
(succ d + succ d))),
hd :
((d + d + succ (succ d) = d + (d + succ (succ d)) → succ d + d +
succ (succ d) = succ d + (d + succ (succ d))) →
d + succ d + succ (succ d) = d + (succ d + succ (succ d)) →
succ (succ d + d) + succ (succ d) = succ d + (succ d + succ (succ
d))) →
(d + d + succ (succ (succ d)) = d + (d + succ (succ (succ d))) →
succ d + d + succ (succ (succ d)) = succ d + (d + succ (succ (succ
d)))) →
d + succ d + succ (succ (succ d)) = d + (succ d + succ (succ (succ
d))) →
succ (succ (succ d + d) + succ (succ d)) = succ (succ d + (succ d +
succ (succ d))),
hd :
(d + d + succ (succ (succ d)) = d + (d + succ (succ (succ d))) →
succ d + d + succ (succ (succ d)) = succ d + (d + succ (succ (succ
d)))) →
d + succ d + succ (succ (succ d)) = d + (succ d + succ (succ (succ
d))) →
succ (succ d + d) + succ (succ (succ d)) = succ d + (succ d + succ
(succ (succ d))),
hd :
d + d + succ (succ (succ (succ d))) = d + (d + succ (succ (succ
(succ d)))) →
succ d + d + succ (succ (succ (succ d))) = succ d + (d + succ (succ
(succ (succ d)))),
hd : d + succ d + succ (succ (succ (succ d))) = d + (succ d + succ
(succ (succ (succ d))))
⊢ succ (succ (succ (succ (succ (succ d + d) + d)))) = succ (succ
(succ (succ (succ d + (succ d + d)))))
```

Addition World

Level 3: `succ_add`

Oh no! On the way to `add_comm`, a wild `succ_add` appears. `succ_add` is the proof that `succ(a) + b = succ(a + b)` for `a` and `b` in your natural number type. We need to prove this now, because we will need to use this result in our proof that `a + b = b + a` in the next level.

NB: think about why computer scientists called this result `succ_add` . There is a logic to all the names.

Note that if you want to be more precise about exactly where you want to rewrite something like `add_succ` (the proof you already have), you can do things like `rw add_succ (succ a)` or `rw add_succ (succ a) d`, telling Lean explicitly what to use for the input variables for the function `add_succ`. Indeed, `add_succ` is a function -- it takes as input two variables `a` and `b` and outputs a proof that `a + succ(b) = succ(a + b)`. The tactic `rw add_succ` just says to Lean "guess what the variables are".

Lemma

For all natural numbers a, b, we have

$$\mathrm{succ}(a) + b = \mathrm{succ}(a + b).$$

```
lemma succ_add (a b : mynat) : succ a + b = succ (a + b) :=
```

Proof :

```
begin
  induction b with n hd,
  rw add_zero,
  rw add_zero,
  refl,
  rw add_succ,
  rw hd,
  rw add_succ,
  refl,

end
```

Addition World

Level 4: `add_comm` (boss level)

[boss battle music]

Look in Theorem statements -> Addition world to see the proofs you have. These should be enough.

Lemma

On the set of natural numbers, addition is commutative. In other words, for all natural numbers a and b, we have

$$a + b = b + a.$$

```
lemma add_comm (a b : mynat) : a + b = b + a :=
```

Proof :

```
begin
  induction b with n hd,
  rw add_zero,
  rw zero_add,
  refl,
  rw add_succ,
  rw hd,
  rw succ_add,
  refl,

end
```

Addition World

Level 5: `succ_eq_add_one`

I've just added `one_eq_succ_zero` (a proof of `1 = succ(0)`) to your list of theorems; this is true by definition of 1, but we didn't need it until now

Levels 5 and 6 are the two last levels in Addition World. Level 5 involves the number 1. When you see a 1 in your goal, you can write `rw one_eq_succ_zero` to get back to something which only mentions `0`. This is a good move because 0 is easier for us to manipulate than 1 right now, because we have some theorems about 0 (`zero_add`, `add_zero`), but, other than `1 = succ(0)`, no theorems at all which mention 1. Let's prove one now.

Theorem

For any natural number n, we have

$$\mathrm{succ}(n) = n + 1.$$

```
theorem succ_eq_add_one (n : mynat) : succ n = n + 1 :=
```

Proof :

```
begin
  induction n with n hd,
  rw zero_add,
  rw one_eq_succ_zero,
  refl,
```

Proof :

```
begin
rw succ_add,
rw hd,
refl,

end
```

Level 6: `add_right_comm`

Lean sometimes writes `a + b + c`. What does it mean? The convention is that if there are no brackets displayed in an addition formula, the brackets are around the left most `+` (Lean's addition is "left associative"). So the goal in this level is `(a + b) + c = (a + c) + b`. This isn't quite `add_assoc` or `add_comm`, it's something you'll have to prove by putting these two theorems together.

If you hadn't picked up on this already, `rw add_assoc` will change `(x + y) + z` to `x + (y + z)`, but to change it back you will need `rw ← add_assoc`. Get the left arrow by typing `\l` then the space bar (note that this is L for left, not a number 1). Similarly, if `h : a = b` then `rw h` will change `a`'s to `b`'s and `rw ← h` will change `b`'s to `a`'s.

Also, you can be (and will need to be, in this level) more precise about where to rewrite theorems. `rw add_comm,` will just find the first `? + ?` it sees and swap it around. You can target more specific additions like this: `rw add_comm a` will swap around additions of the form `a + ?`, and `rw add_comm a b,` will only swap additions of the form `a + b`.

Where next?

There are thirteen more levels about addition after this one, but before you can attempt them you need to learn some more tactics. So after this level you have a choice -- either move on to Multiplication World (which you can solve with the tactics you know) or try Function World (and learn some new ones). After solving this level, click "Main Menu" in the top left to take you back to the overworld, and make your choice. Other things, perhaps of interest to some players, are mentioned below the lemma.

Lemma

For all natural numbers a, b and c, we have

$$a + b + c = a + c + b.$$

```
lemma add_right_comm (a b c : mynat) : a + b + c = a + c + b :=
```

Proof :

```
begin
induction a with n hd,
rw zero_add,
rw zero_add,
induction b with n hd,
end
```

```
begin
rw zero_add,
rw add_zero,
refl,
rw add_succ,
```

```
begin
rw succ_add,
rw hd,
refl,
rw succ_add,
```

```
begin
rw succ_add,
rw hd,
rw succ_add,
rw succ_add,
```

```
refl,
```