Compiler Construction

Lecture 8: CPS & defunctionalization

Jeremy Yallop

jeremy.yallop@cl.cam.ac.uk

Lent 2023

Continuation-**P**assing **S**tyle

Programs in **continuation-passing style** have some useful properties:

Evaluation order is explicit

Every call is a tail call

$$\mathsf{f}\ (\mathsf{g}\ \mathsf{x}) \quad \rightsquigarrow \quad \mathsf{g}\ \mathsf{x}\ (\mathsf{fun}\ \mathsf{y} \to \mathsf{f}\ \mathsf{y}\ \mathsf{k})$$

Every intermediate result is named

Every *continuation* is reified

CPS conversion of `fib`


```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m-1) + fib (m-2)
```

let-bind function calls

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else let a = fib (m-1) in
      let b = fib (m-2) in
      a+b
```

CPS convert

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m-1) (fun a →
      fib_cps (m-2) (fun b →
      k (a+b)))
```

CPS conversion of `fib`: details

1. Add a continuation parameter `k` to each function
2. Apply `k` to values returned by the function
3. Replace each application `let` binding with a continuation argument

```
let rec fib m =
  if m = 0 then 1
  else if m = 1 then 1
  else let a = fib (m-1) in
       let b = fib (m-2) in
       a+b
```

CPS convert →

```
let rec fib_cps m k =
  if m = 0 then k 1
  else if m = 1 then k 1
  else fib_cps (m-1) (fun a →
       fib_cps (m-2) (fun b →
       k (a+b)))
```

CPS conversion of `fib`: details

1. Add a continuation parameter `k` to each function
2. Apply `k` to values returned by the function
3. Replace each application `let` binding with a continuation argument

```
let rec fib m =
  if m = 0 then 1
  else if m = 1 then 1
  else let a = fib (m-1) in
       let b = fib (m-2) in
       a+b
```

CPS convert →

```
let rec fib_cps m k =
  if m = 0 then k 1
  else if m = 1 then k 1
  else fib_cps (m-1) (fun a →
       fib_cps (m-2) (fun b →
       k (a+b)))
```

CPS conversion of `fib`: details

1. Add a continuation parameter `k` to each function
2. Apply `k` to values returned by the function
3. Replace each application `let` binding with a continuation argument

```
let rec fib m =
  if m = 0 then 1
  else if m = 1 then 1
  else let a = fib (m-1) in
       let b = fib (m-2) in
       a+b
```

CPS convert →

```
let rec fib_cps m k =
  if m = 0 then k 1
  else if m = 1 then k 1
  else fib_cps (m-1) (fun a →
       fib_cps (m-2) (fun b →
       k (a+b)))
```

CPS conversion of `fib`: details

1. Add a continuation parameter `k` to each function
2. Apply `k` to values returned by the function
3. Replace each application `let` binding with a continuation argument

```
let rec fib m =
  if m = 0 then 1
  else if m = 1 then 1
  else let a = fib (m-1) in
       let b = fib (m-2) in
       a+b
```

CPS convert →

```
let rec fib_cps m k =
  if m = 0 then k 1
  else if m = 1 then k 1
  else fib_cps (m-1) (fun a →
       fib_cps (m-2) (fun b →
       k (a+b)))
```

CPS conversion of `fib`: details

1. Add a continuation parameter `k` to each function
2. Apply `k` to values returned by the function
3. Replace each application `let` binding with a continuation argument

```
let rec fib m =
  if m = 0 then 1
  else if m = 1 then 1
  else let a = fib (m-1) in
       let b = fib (m-2) in
       a+b
```

CPS convert →

```
let rec fib_cps m k =
  if m = 0 then k 1
  else if m = 1 then k 1
  else fib_cps (m-1) (fun a →
       fib_cps (m-2) (fun b →
       k (a+b)))
```

`fib_cps` has the type `int → (int → int) → int`.

To recover a function of type `int →int`, pass the identity continuation `fun x →x`:

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))

let fib_1 x = fib_cps x (fun x → x)
```

Now `fib_1` can be used like `fib`:

```
List.map fib_1 [0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10]
⇝ [1; 1; 2; 3; 5; 8; 13; 21; 34; 55; 89]
```

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all `n` $\leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m–1) + fib (m–2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m–1) (fun a →
      fib_cps (m–2) (fun b →
      k (a+b)))
```

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all $m \geq 0$,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all $n \leq m$, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

fib_cps (m+1) k

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all $m \geq 0$,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all $n \leq m$, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

fib_cps (m+1) k

≡ (expand `fib_cps`) ...

if m+1 = 1 then k 1 else fib_cps ((m+1)−1) (fun a →fib_cps ((m+1)−2) (fun b →k (a+b)))

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all `n` $\leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

```
if m+1 = 1 then k 1 else fib_cps ((m+1)−1) (fun a →fib_cps ((m+1)−2) (fun b →k (a+b)))
```

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all $m \geq 0$,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all $n \leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

if m+1 = 1 then k 1 else fib_cps ((m+1)−1) (fun a →fib_cps ((m+1)−2) (fun b →k (a+b)))

≡ (arithmetic) . . .

if m+1 = 1 then k 1 else fib_cps m (fun a →fib_cps (m−1) (fun b →k (a+b)))

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int → int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all `n` $\leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

```
if m+1 = 1 then k 1 else fib_cps m (fun a →fib_cps (m−1) (fun b →k (a+b)))
```

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all `n` $\leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

if m+1 = 1 then k 1 else fib_cps m (fun a →fib_cps (m−1) (fun b →k (a+b)))

≡ (inductive assumption for $m-1$ and k = (fun b →k (a+b))) ...

if m+1 = 1 then k 1 else fib_cps m (fun a →(fun b → k (a+b)) (fib (m−1)))

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all `n` $\leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

```
if m+1 = 1 then k 1 else fib_cps m (fun a →(fun b → k (a+b)) (fib (m−1)))
```

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all `n` $\leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

if m+1 = 1 then k 1 else fib_cps m (fun a →(fun b → k (a+b)) (fib (m−1)))

≡ (inductive assumption for *m* and `k = (fun a →(fun b →k (a+b)) (fib (m-1)))`) ...

if m+1 = 1 then k 1 else (fun a →(fun b → k (a+b)) (fib (m−1))) (fib m)

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all $m \geq 0$,
  for all `k : int → int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all $n \leq m$, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

```
if m+1 = 1 then k 1 else (fun a → (fun b → k (a+b)) (fib (m−1))) (fib m)
```

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all `n` $\leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

if m+1 = 1 then k 1 else (fun a →(fun b → k (a+b)) (fib (m−1))) (fib m)

≡ (beta reduction ×2) ...

if m+1 = 1 then k 1 else k (fib m + fib (m−1))

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all `n` $\leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

```
if m+1 = 1 then k 1 else k (fib m + fib (m−1))
```

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all `m` ≥ 0,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all `n` $\leq$ `m`, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

if m+1 = 1 then k 1 else k (fib m + fib (m−1))

≡ (if e1 then k e2 else k e3 ≡ k (if e1 then e2 else e3)) ...

k (if m+1 = 1 then 1 else fib m + fib (m−1))

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all $m \geq 0$,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all $n \leq m$, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m−1) + fib (m−2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m−1) (fun a →
      fib_cps (m−2) (fun b →
      k (a+b)))
```

k (if m+1 = 1 then 1 else fib m + fib (m−1))

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all $m \geq 0$,
  for all $k : \text{int} \to \text{int}$,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all $n \leq m$, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
  if m = 0 then 1
  else if m = 1 then 1
  else fib (m-1) + fib (m-2)
```

```
let rec fib_cps m k =
  if m = 0 then k 1
  else if m = 1 then k 1
  else fib_cps (m-1) (fun a →
       fib_cps (m-2) (fun b →
       k (a+b)))
```

k (if m+1 = 1 then 1 else fib m + fib (m−1))

≡ (definition of `fib`) ...

k (fib (m+1))

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all $m \geq 0$,
  for all `k : int →int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all $n \leq m$, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
 if m = 0 then 1
 else if m = 1 then 1
 else fib (m–1) + fib (m–2)
```

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m–1) (fun a →
      fib_cps (m–2) (fun b →
      k (a+b)))
```

k (fib (m+1))

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Correctness of CPS conversion for `fib`

Claim

For all $m \geq 0$,
  for all `k : int → int`,
    `fib_cps m k = k (fib m)`.

Proof

By strong induction on `m`.

Base case (`m = 0`): `fib_cps 0 k = k 1 = k (fib 0)`.

Inductive step:
Assume for all $n \leq m$, `k (fib n) = fib_cps n k`.
We want to show: `fib_cps (m+1) k = k (fib (m+1))`.

```
let rec fib m =
  if m = 0 then 1
  else if m = 1 then 1
  else fib (m-1) + fib (m-2)
```

```
let rec fib_cps m k =
  if m = 0 then k 1
  else if m = 1 then k 1
  else fib_cps (m-1) (fun a →
       fib_cps (m-2) (fun b →
       k (a+b)))
```

`k (fib (m+1))`

QED

NB: We approximate OCaml functions by mathematical functions, ignoring side effects etc.

Defunctionalization

Defunctionalized programs have some useful properties:

No higher-order functions

All values are data

All control-flow is first order

Every function is named

Defunctionalization: example

1. Add a constructor to `fn` for each `fun`
2. Replace each `fun` with its constructor
3. Add a case to `apply` for each `fun`
4. Replace each application `p x` with `apply p x`

```
let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter (fun x → x < 3) l
  @ filter (fun x → x > y) l
```

```
type fn = Lt_three
        | Gt of int

let apply fn x =
  match fn, x with
  | Lt_three, x → x < 3
  | Gt y, x → x > y

let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if apply p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter Lt_three l
  @ filter (Gt y) l
```

Defunctionalization: example

1. Add a constructor to `fn` for each `fun`
2. Replace each `fun` with its constructor
3. Add a case to `apply` for each `fun`
4. Replace each application `p x` with `apply p x`

```
let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter (fun x → x < 3) l
  @ filter (fun x → x > y) l
```

```
type fn = Lt_three
        | Gt of int

let apply fn x =
  match fn, x with
  | Lt_three, x → x < 3
  | Gt y, x → x > y

let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if apply p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter Lt_three l
  @ filter (Gt y) l
```

Defunctionalization: example

1. Add a constructor to `fn` for each `fun`
2. Replace each `fun` with its constructor
3. Add a case to `apply` for each `fun`
4. Replace each application `p x` with `apply p x`

```
let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter (fun x → x < 3) l
  @ filter (fun x → x > y) l
```

```
type fn = Lt_three
        | Gt of int

let apply fn x =
  match fn, x with
  | Lt_three, x → x < 3
  | Gt y, x → x > y

let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if apply p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter Lt_three l
  @ filter (Gt y) l
```

Defunctionalization: example

1. Add a constructor to `fn` for each `fun`
2. Replace each `fun` with its constructor
3. Add a case to `apply` for each `fun`
4. Replace each application `p x` with `apply p x`

```
let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter (fun x → x < 3) l
  @ filter (fun x → x > y) l
```

```
type fn = Lt_three
        | Gt of int

let apply fn x =
  match fn, x with
  | Lt_three, x → x < 3
  | Gt y, x → x > y

let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if apply p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter Lt_three l
  @ filter (Gt y) l
```

Defunctionalization: example

1. Add a constructor to `fn` for each `fun`
2. Replace each `fun` with its constructor
3. Add a case to `apply` for each `fun`
4. Replace each application `p x` with `apply p x`

```
let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter (fun x → x < 3) l
  @ filter (fun x → x > y) l
```

```
type fn = Lt_three
        | Gt of int

let apply fn x =
  match fn, x with
  | Lt_three, x → x < 3
  | Gt y, x → x > y

let rec filter p l =
  match l with
  | [] → []
  | x :: xs →
      if apply p x
      then x :: filter p xs
      else filter p xs

let f l y =
    filter Lt_three l
  @ filter (Gt y) l
```

Combining CPS & defunctionalization

```
let rec fib_cps m k =
 if m = 0 then k 1
 else if m = 1 then k 1
 else fib_cps (m-1) (fun a → (* K1 *)
      fib_cps (m-2) (fun b → (* K2 *)
      k (a+b)))

let fib_1 x = fib_cps x (fun x → x) (* ID *)
```

To defunctionalize `fib_cps`, define a constructor for each `fun`:

```
type cont = ID | K1 of int * cont | K2 of int * cont
```

Constructor arguments are free variables, and we treat `k2` as free in `k1`:

```
let k2 = fun a b → k (a+b)
let k1 = fun a → fib_cps (m-2) (k2 a)
```

Now define an apply function of type `cont →int →int`

```
type cont = ID | K1 of int * cont | K2 of int * cont

let rec apply_cont k v = match k, v with
| ID, a → a
| K1 (m, k), a → fib_cps_defun (m−2) (K2 (a, k))
| K2 (a, k), b → apply_cont k (a+b)
```

and call `apply_cont` at every application of a continuation:

```
and fib_cps_defun m k =
  if m = 0 then apply_cont k 1
  else if m = 1 then apply_cont k 1
  else fib_cps_defun (m −1) (K1 (m, k))

let fib_2 m = fib_cps_defun m ID
```

Claim

Let `<c>` (type `cont`) represent a continuation `c : int → int` constructed by `fib_cps`. Then

```
apply_cont <c> m   =   c m
          and
fib_cps n c        =   fib_cps_defun n <c>
```

(**Proof** left as an exercise)

Observation: continuations have list (stack) structure


```
type int_list =
    NIL
  | CONS of int * int_list
```

```
type cont =
    ID                    (* 'Nil' *)
  | K1 of int * cont      (* 'Cons' *)
  | K2 of int * cont      (* 'Cons' *)
```

Idea: replace `cont` with standard lists:

```
type tag = SUB2 of int   (* K1: k (a+b) *)
         | PLUS of int   (* K2: fib_cps (m-2) (k2 a) *)

type tag_list_cont = tag list
```

fib_cps_defun revisited, using lists for continuations


```
type tag = SUB2 of int | PLUS of int
type tag_list_cont = tag list

let rec apply_tag_list_cont k v = match k, v with
  | [], a → a
  | SUB2 m :: k, a → fib_cps_defun_tags (m−2) (PLUS a :: k)
  | PLUS a :: k, b → apply_tag_list_cont k (a+b)

and fib_cps_defun_tags m k =
  if m = 0 then apply_tag_list_cont k 1
  else if m = 1 then apply_tag_list_cont k 1
  else fib_cps_defun_tags (m−1) (SUB2 m :: k)

let fib_3 m = fib_cps_defun_tags m []
```

Mutual recursion

Mutual recursion $\rightsquigarrow$ single recursion

Mutual recursion can be eliminated using **indexing**.

Given a set of mutually-recursive functions:

```
let rec is_even n = n = 0 || is_odd (n - 1)
and is_odd n = n <> 0 && is_even (n - 1)
```

define an index datatype with one constructor for each function:

```
type eo = Even | Odd
```

and define a function that maps an index argument to a corresponding body:

```
let rec is f n =
  match f with
  | Even → n = 0 || is Odd (n - 1)
  | Odd → n <> 0 && is Even (n - 1)
```

Mutual recursion $\rightsquigarrow$ single recursion for `fib`


```
type state_type =
| FIB (* for right-hand-sides starting with fib_ *)
| APP (* for right-hand-sides starting with apply_ *)

type state = (state_type * int * tag_list_cont) → int

(* eval acts as either apply_tag_list_cont or fib_cps_defun_tags *)
let rec eval = function
| FIB, 0, k → eval (APP, 1, k)
| FIB, 1, k → eval (APP, 1, k)
| FIB, m, k → eval (FIB, m-1, SUB2 m :: k)
| APP, a, SUB2 m :: k → eval (FIB, m-2, PLUS a :: k)
| APP, b, PLUS a :: k → eval (APP, a+b, k)
| APP, a, [] → a

let fib_4 m = eval (FIB, m, [])
```

Eliminate tail recursion to obtain *The Fibonacci Machine*


```
(* step : state → state *)
let step = function
| FIB, 0, k → (APP, 1, k)
| FIB, 1, k → (APP, 1, k)
| FIB, m, k → (FIB, m−1, SUB2 m :: k)
| APP, a, SUB2 m :: k → (FIB, m−2, PLUS a :: k)
| APP, b, PLUS a :: k → (APP, a+b, k)
| _ → failwith "step : runtime error!"

let rec driver state = function (* clearly tail recursive! *)
| APP, a, [] → a
| state → driver (step state)

(* fib_5 : int → int *)
let fib_5 m = driver (FIB, m, [])
```

(This version makes the tail-recursive structure very explicit.)

Tracing of fib_5 6


```
1   FIB   6   []
```

Tracing of fib_5 6


```
1   FIB   6   []
2   FIB   5   [SUB2 6]
```

Tracing of fib_5 6


```
1   FIB   6   []
2   FIB   5   [SUB2 6]
3   FIB   4   [SUB2 6,SUB2 5]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
1   FIB   6   []
2   FIB   5   [SUB2 6]
3   FIB   4   [SUB2 6,SUB2 5]
4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
1   FIB   6   []
2   FIB   5   [SUB2 6]
3   FIB   4   [SUB2 6,SUB2 5]
4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
1   FIB   6   []
2   FIB   5   [SUB2 6]
3   FIB   4   [SUB2 6,SUB2 5]
4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
```

Tracing of fib_5 6


```
1   FIB   6   []
2   FIB   5   [SUB2 6]
3   FIB   4   [SUB2 6,SUB2 5]
4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
1   FIB   6   []
2   FIB   5   [SUB2 6]
3   FIB   4   [SUB2 6,SUB2 5]
4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
```

Tracing of fib_5 6


```
1   FIB   6   []
2   FIB   5   [SUB2 6]
3   FIB   4   [SUB2 6,SUB2 5]
4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
20   APP   5   [SUB2 6,SUB2 5]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
20   APP   5   [SUB2 6,SUB2 5]
21   FIB   3   [SUB2 6,PLUS 5]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
20   APP   5   [SUB2 6,SUB2 5]
21   FIB   3   [SUB2 6,PLUS 5]
22   FIB   2   [SUB2 6,PLUS 5,SUB2 3]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
20   APP   5   [SUB2 6,SUB2 5]
21   FIB   3   [SUB2 6,PLUS 5]
22   FIB   2   [SUB2 6,PLUS 5,SUB2 3]
23   FIB   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
20   APP   5   [SUB2 6,SUB2 5]
21   FIB   3   [SUB2 6,PLUS 5]
22   FIB   2   [SUB2 6,PLUS 5,SUB2 3]
23   FIB   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24   APP   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
20   APP   5   [SUB2 6,SUB2 5]
21   FIB   3   [SUB2 6,PLUS 5]
22   FIB   2   [SUB2 6,PLUS 5,SUB2 3]
23   FIB   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24   APP   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25   FIB   0   [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
20   APP   5   [SUB2 6,SUB2 5]
21   FIB   3   [SUB2 6,PLUS 5]
22   FIB   2   [SUB2 6,PLUS 5,SUB2 3]
23   FIB   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24   APP   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25   FIB   0   [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26   APP   1   [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
20   APP   5   [SUB2 6,SUB2 5]
21   FIB   3   [SUB2 6,PLUS 5]
22   FIB   2   [SUB2 6,PLUS 5,SUB2 3]
23   FIB   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24   APP   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25   FIB   0   [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26   APP   1   [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27   APP   2   [SUB2 6,PLUS 5,SUB2 3]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1   FIB   6   []
 2   FIB   5   [SUB2 6]
 3   FIB   4   [SUB2 6,SUB2 5]
 4   FIB   3   [SUB2 6,SUB2 5,SUB2 4]
 5   FIB   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6   FIB   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8   FIB   0   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9   APP   1   [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10   APP   2   [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11   FIB   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12   APP   1   [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13   APP   3   [SUB2 6,SUB2 5,SUB2 4]
14   FIB   2   [SUB2 6,SUB2 5,PLUS 3]
15   FIB   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16   APP   1   [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17   FIB   0   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18   APP   1   [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19   APP   2   [SUB2 6,SUB2 5,PLUS 3]
20   APP   5   [SUB2 6,SUB2 5]
21   FIB   3   [SUB2 6,PLUS 5]
22   FIB   2   [SUB2 6,PLUS 5,SUB2 3]
23   FIB   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24   APP   1   [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25   FIB   0   [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26   APP   1   [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27   APP   2   [SUB2 6,PLUS 5,SUB2 3]
28   FIB   1   [SUB2 6,PLUS 5,PLUS 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
```

Tracing of fib_5 6

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []                                                    26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
 2  FIB  5  [SUB2 6]                                              27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
 3  FIB  4  [SUB2 6,SUB2 5]                                       28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]                                29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]                         30  APP  3  [SUB2 6,PLUS 5]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]                  31  APP  8  [SUB2 6]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]                  32  FIB  4  [PLUS 8]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]                  33  FIB  3  [PLUS 8,SUB2 4]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
```

Tracing of fib_5 6


```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
```

Tracing of fib_5 6


```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
```

Tracing of fib_5 6


```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
42  APP  3  [PLUS 8,SUB2 4]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
42  APP  3  [PLUS 8,SUB2 4]
43  FIB  2  [PLUS 8,PLUS 3]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
42  APP  3  [PLUS 8,SUB2 4]
43  FIB  2  [PLUS 8,PLUS 3]
44  FIB  1  [PLUS 8,PLUS 3,SUB2 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
42  APP  3  [PLUS 8,SUB2 4]
43  FIB  2  [PLUS 8,PLUS 3]
44  FIB  1  [PLUS 8,PLUS 3,SUB2 2]
45  APP  1  [PLUS 8,PLUS 3,SUB2 2]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
42  APP  3  [PLUS 8,SUB2 4]
43  FIB  2  [PLUS 8,PLUS 3]
44  FIB  1  [PLUS 8,PLUS 3,SUB2 2]
45  APP  1  [PLUS 8,PLUS 3,SUB2 2]
46  FIB  0  [PLUS 8,PLUS 3,PLUS 1]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
42  APP  3  [PLUS 8,SUB2 4]
43  FIB  2  [PLUS 8,PLUS 3]
44  FIB  1  [PLUS 8,PLUS 3,SUB2 2]
45  APP  1  [PLUS 8,PLUS 3,SUB2 2]
46  FIB  0  [PLUS 8,PLUS 3,PLUS 1]
47  APP  1  [PLUS 8,PLUS 3,PLUS 1]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
42  APP  3  [PLUS 8,SUB2 4]
43  FIB  2  [PLUS 8,PLUS 3]
44  FIB  1  [PLUS 8,PLUS 3,SUB2 2]
45  APP  1  [PLUS 8,PLUS 3,SUB2 2]
46  FIB  0  [PLUS 8,PLUS 3,PLUS 1]
47  APP  1  [PLUS 8,PLUS 3,PLUS 1]
48  APP  2  [PLUS 8,PLUS 3]
```

Tracing of fib_5 6


```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
42  APP  3  [PLUS 8,SUB2 4]
43  FIB  2  [PLUS 8,PLUS 3]
44  FIB  1  [PLUS 8,PLUS 3,SUB2 2]
45  APP  1  [PLUS 8,PLUS 3,SUB2 2]
46  FIB  0  [PLUS 8,PLUS 3,PLUS 1]
47  APP  1  [PLUS 8,PLUS 3,PLUS 1]
48  APP  2  [PLUS 8,PLUS 3]
49  APP  5  [PLUS 8]
```

Tracing of fib_5 6

CPS

D17n

CPS + D17n

Mutual recursion

```
 1  FIB  6  []
 2  FIB  5  [SUB2 6]
 3  FIB  4  [SUB2 6,SUB2 5]
 4  FIB  3  [SUB2 6,SUB2 5,SUB2 4]
 5  FIB  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
 6  FIB  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 7  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,SUB2 2]
 8  FIB  0  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
 9  APP  1  [SUB2 6,SUB2 5,SUB2 4,SUB2 3,PLUS 1]
10  APP  2  [SUB2 6,SUB2 5,SUB2 4,SUB2 3]
11  FIB  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
12  APP  1  [SUB2 6,SUB2 5,SUB2 4,PLUS 2]
13  APP  3  [SUB2 6,SUB2 5,SUB2 4]
14  FIB  2  [SUB2 6,SUB2 5,PLUS 3]
15  FIB  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
16  APP  1  [SUB2 6,SUB2 5,PLUS 3,SUB2 2]
17  FIB  0  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
18  APP  1  [SUB2 6,SUB2 5,PLUS 3,PLUS 1]
19  APP  2  [SUB2 6,SUB2 5,PLUS 3]
20  APP  5  [SUB2 6,SUB2 5]
21  FIB  3  [SUB2 6,PLUS 5]
22  FIB  2  [SUB2 6,PLUS 5,SUB2 3]
23  FIB  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
24  APP  1  [SUB2 6,PLUS 5,SUB2 3,SUB2 2]
25  FIB  0  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
26  APP  1  [SUB2 6,PLUS 5,SUB2 3,PLUS 1]
27  APP  2  [SUB2 6,PLUS 5,SUB2 3]
28  FIB  1  [SUB2 6,PLUS 5,PLUS 2]
29  APP  1  [SUB2 6,PLUS 5,PLUS 2]
30  APP  3  [SUB2 6,PLUS 5]
31  APP  8  [SUB2 6]
32  FIB  4  [PLUS 8]
33  FIB  3  [PLUS 8,SUB2 4]
34  FIB  2  [PLUS 8,SUB2 4,SUB2 3]
35  FIB  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
36  APP  1  [PLUS 8,SUB2 4,SUB2 3,SUB2 2]
37  FIB  0  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
38  APP  1  [PLUS 8,SUB2 4,SUB2 3,PLUS 1]
39  APP  2  [PLUS 8,SUB2 4,SUB2 3]
40  FIB  1  [PLUS 8,SUB2 4,PLUS 2]
41  APP  1  [PLUS 8,SUB2 4,PLUS 2]
42  APP  3  [PLUS 8,SUB2 4]
43  FIB  2  [PLUS 8,PLUS 3]
44  FIB  1  [PLUS 8,PLUS 3,SUB2 2]
45  APP  1  [PLUS 8,PLUS 3,SUB2 2]
46  FIB  0  [PLUS 8,PLUS 3,PLUS 1]
47  APP  1  [PLUS 8,PLUS 3,PLUS 1]
48  APP  2  [PLUS 8,PLUS 3]
49  APP  5  [PLUS 8]
50  APP  13 []
```

We turned the recursive cps into a function that uses no OCaml stack space

The transformed cps function carries its own stack as an extra argument

We transformed cps incrementally, with each step easily proved correct

Next time: application to **interpreter 0**