```
1
2  let rec tail_sum n acc =
3    print_endline ("acc: " ^ (string_of_int acc));
4  if n = 0 then acc else tail_sum (n - 1) (acc + n)
5
6  let _ = tail_sum 1000000 1
7
```

Run

Stop 

Step 

Run ▶ Stop ■ Step ↷

```
1
2  let rec tail_sum n acc =
3    print_endline ("acc: " ^ (string_of_int acc));
4  if n = 0 then acc else tail_sum (n - 1) (acc + n)
5
6  let _ = tail_sum 1000000 1
7
```

HOW TO BUILD THIS MECHANISM?

```
1   const pythagoras = (x, y) {
 1    return add(square(x), square(y))
 2  }
 3
 4  // =========Transform to CPS=========
 5
 6  const pythagoras = (x, y, k) => {
 7    return square(x, x2 => {
 8      return square(y, y2 =>{
 9        return add(x2, y2, k)
10      })
11    })
12  }
```

CONTINUATION PASSING STYLE

JS GENERATORS

```
 4  const pythagoras = (x, y) {
 3    return add(square(x), square(y))
 2  }
 1
5   // =======Transform using generators=======
 1
 2  const pythagoras = function* (x, y) {
 3    const _t1 = yield* square(x)
 4    const _t2 = yield* square(y)
 5
 6    return yield* add(_t1, _t2)
 7  }
```