1.

```
local B in
 thread          // S1
   B=true        // T1
 end
 thread          // S2
   B=false       // T2
 end
 if B then       // S3
   skip Browse B   // S3.1
 end
end
```

S1 T1 S2 T2 S3 ---- displays nothing

// thread1 ran first, and B = true, so B is bound to true. and then thread2 ran after, but B is already bound to ture, so B can't be bound to false. so, unification error happens.

S1 T1 S2 S3 S3.1 T2 ----- displays true

// thread1 ran first, and B = true, so B is bound to true. and then thread2 ran after, but B is already bound to ture, so B can't be bound to false. so, unification error happens.

S1 S2 T1 T2 S3 ---- displays nothing

// thread1 ran first, and B = true, so B is bound to true. and then thread2 ran after, but B is already bound to ture, so B can't be bound to false. so, unification error happens.

S1 S2 T2 S3 T1 ---- displays nothing

// thread2 ran first, and B = false, so B is bound to false. and then thread1 ran after, but B is already bound to false, so B can't be bound to true. so, unification error happens.

S2 S1 T1 S3 S3.1 T2 ----- displays true

// thread1 ran first, and B = true, so B is bound to true. and then thread2 ran after, but B is already bound to ture, so B can't be bound to false. so, unification error happens.

S1 S2 T1 S3 S3.1 T2 ----- displays true

// thread1 ran first, and B = true, so B is bound to true. and then thread2 ran after, but B is already bound to ture, so B can't be bound to false. so, unification error happens.

S2 S1 T2 S3 T1 ---- displays nothing

// thread2 ran first, and B = false, so B is bound to false. and then thread1 ran after, but B is already bound to false, so B can't be bound to true. so, unification error happens.

S1 S2 T2 T1 S3 S3.1 ----- displays true

// thread2 ran first, and B = false, so B is bound to false. and then thread1 ran after, but B is already bound to false, so B can't be bound to true. so, unification error happens.

S2 T2 S1 T1 S3 S3.1 ----- displays true

// thread2 ran first, and B = false, so B is bound to false. and then thread1 ran after, but B is already bound to false, so B can't be bound to true. so, unification error happens.

S2 S1 T2 T1 S3 S3.1 ----- displays true

// thread2 ran first, and B = false, so B is bound to false. and then thread1 ran after, but B is already bound to false, so B can't be bound to true. so, unification error happens.

S2 S1 T1 T2 S3 S3.1 ----- displays true

// thread1 ran first, and B = true, so B is bound to true. and then thread2 ran after, but B is already bound to ture, so B can't be bound to false. so, unification error happens.

S2 T2 S1 S3 T1 ----- displays nothing

// thread2 ran first, and B = false, so B is bound to false. and then thread1 ran after, but B is already bound to false, so B can't be bound to true. so, unification error happens.

2.

```
local X Y T in
  thread Y = X end
  X = 3
  skip Browse Y
end

local T1 T2 in
 T2 = thread 3 end
 T1 = thread (4+3) end
 skip Browse T2
 skip Browse T1
end
```

// In infinity case, the output is Y: unbound, T2: unbound, T1: unbound. The reason is those threads are executed after Browse.

// In finite 1 case, the output is Y:3 T2:3 T1: unbound, the thread runs after " X = 3" so we have Y = X = 3. Then for the "T1 T2" one,

// thread of T2 runs first and then "skip Browse T2”, "skip Browse T1", finally, thread of T1. So, we have T2 = 3, T1 = unbound.

3.

```
local Z in
  Z = 3
  thread local X in
    X = 1
    skip Browse X
    skip Browse X
    skip Basic
```

```
   skip Browse X
   skip Browse X
   skip Basic
   skip Browse X
   end
  end
  thread local Y in
   Y = 2
   skip Browse Y
   skip Basic
   skip Browse Y
   skip Browse Y
   skip Browse Y
   skip Basic
   skip Browse Y
   end
  end
  skip Browse Z
  skip Browse Z
  skip Browse Z
  skip Basic
  skip Browse Z
  skip Browse Z
 end
```

4. With quantum of 3, B is bounded to true, but with quantum of 5, B become unbound. The reason why is because When it is large enough, thread takes a lower priority, causing the if statement to be executed first, so B becomes unbound.

5.

(a)

function	Maximum X	Time (s)	Memory (bytes)
fib1_sugar	15	37.96	13,435,508,912
fib2_sugar	1800	57.24	20,367,039,712
fib1_thread	13	25.71	3,371,539,608

For fib1_sugar, the

maximum X within one minute is 15. and

for fib2_sugar, the maximum X is 1800, for fib1_thread, the

maximum X is 13.

Why is that result is because, fib1_sugar and fib1_thread

are using recursion without tail call, and there are lots of repetitive calculations

as the progress going through. fib2_sugar is using recursion with tail call and accumulator.

so, there is no repetitive calculations in fib2_sugar. Thus fib2_sugar is more efficient than fib1_sugar and fib1_thread

(b)

```
num_of_threads n
  | n == 0 = 0
  | n == 1 = 0
  | n == 2 = 1
  | otherwise =  num_of_threads(n-2)+num_of_threads(n-1)+2
```

Part 2

1.

```
OddFilter = proc {$ P Out}
   case P
      of nil then Out = nil
      [] '|'(1:X 2: Xr) then
      if ({Mod X 2} == 0) then T in
         Out = (X|T)
         {OddFilter Xr T}
      else
         {OddFilter Xr Out}
      end
   end
end
```

2.

```
   Consumer = fun {$ P} in
      case P
         of nil then 0
         [] '|'(1:X 2: Xr) then
         (X + {Consumer Xr})
      end
   end
```

3.

```
local Producer OddFilter Consumer N L P F S in
   thread
   Producer = proc {$ N Limit Out}
   if (N<Limit) then T N1 in
      Out = (N|T)
      N1 = (N + 1)
      {Producer N1 Limit T}
   else Out = nil
   end
   end
   end
   thread
   OddFilter = proc {$ P Out}
      case P
         of nil then Out = nil
         [] '|'(1:X 2: Xr) then
         if ({Mod X 2} == 0) then T in
            Out = (X|T)
            {OddFilter Xr T}
         else
            {OddFilter Xr Out}
         end
      end
   end
   end
   thread
```

```
   Consumer = fun {$ P} in
      case P
         of nil then 0
         [] '|'(1:X 2:Xr) then
         (X + {Consumer Xr})
      end
   end
   end

   // Example Testing
   N = 0
   L = 101
   {Producer N L P} // [0 1 2 ... 100]
   {OddFilter P F} // [0 2 4 .. 100]
   {Consumer P S}
   skip Browse F
   skip Browse S
end
```