1.

a. //S1 S2 T2 T1 S3 S3.1 --true()
//S1 S2 T1 T2 S3 S3.1 --unification error
//S3 S3.1 S1 S2 T2 T1 --thread suspended
//S2 T2 S1 T1 S3 S3.1 -- unfication error

//unification error occurs whenever B is being reassigned to a different bool value

b. //finite 1 gives T2 and Y their values
//finite 2 gives T2 and Y their values
//finite 3 gives T2 and Y their values
//finite 4 gives T1 its value
//finite 5 gives Unbound for all 3
//finite 6 gives T2 its value
//finite 7 gives T2 and T1 their value
//finite 8 gives T1 its value
//finite 9-Infinity gives all Unbound

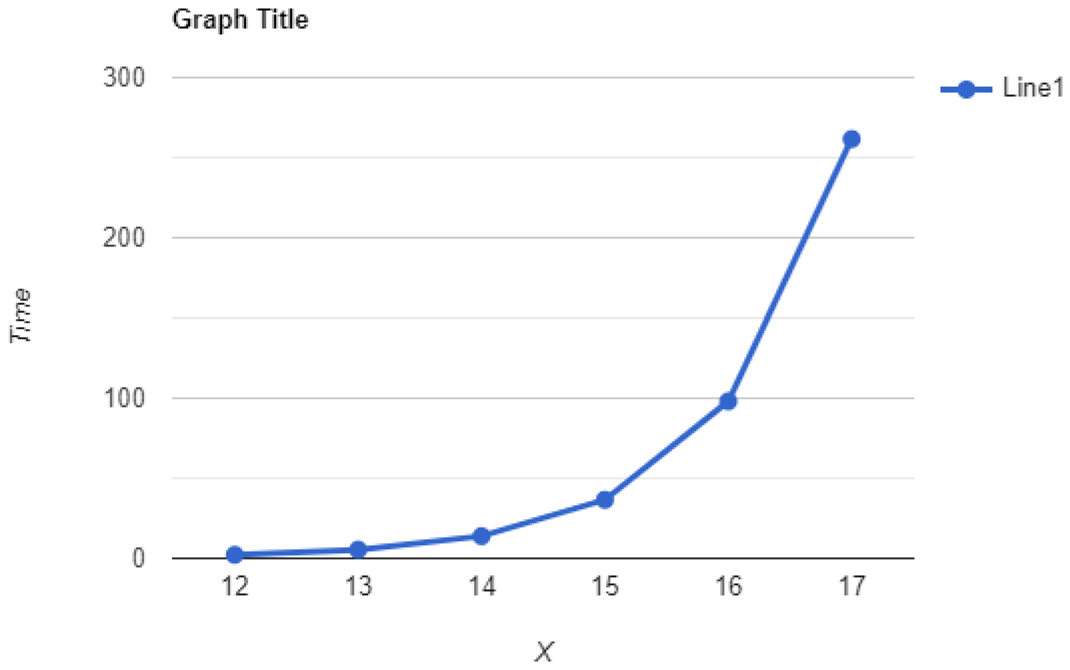
//Having a quantum of infinity is only allowing for the skip browses to be printed
//in the main thread before the computations of the other threads are actually being executed
//having a low quantum, like finite 1, can allow for the main thread to let the other threads do their computations

c.

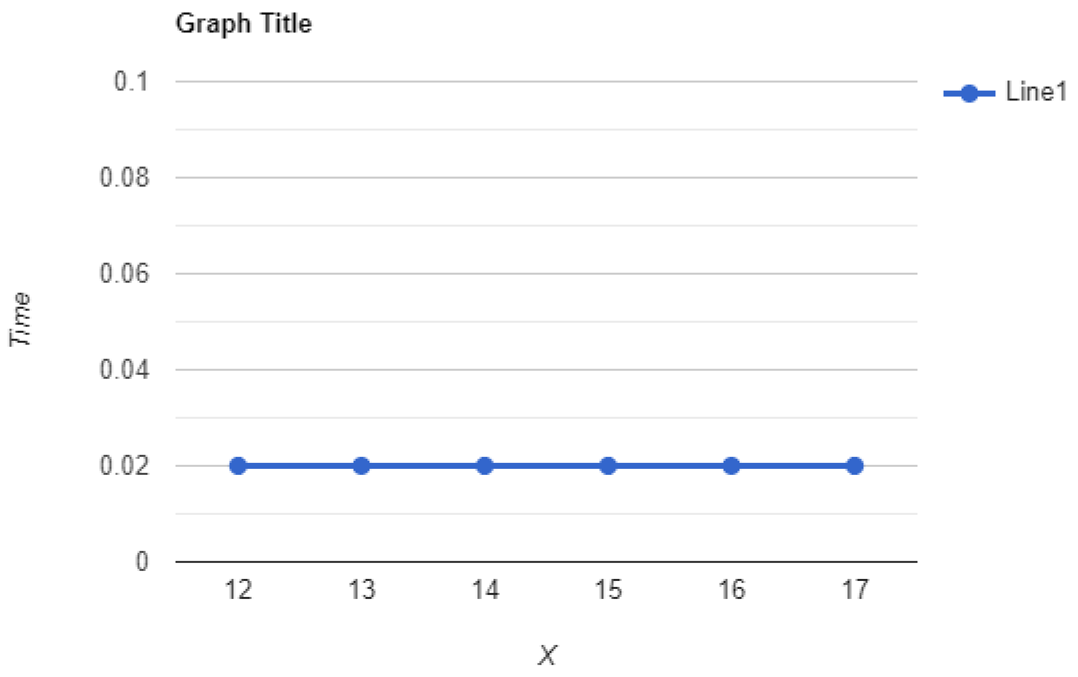
```
local Z in
  Z = 3
  thread local X in
    X = 1
    skip Browse X
    skip Browse X
            skip Basic
    skip Browse X
    skip Browse X
            skip Basic
    skip Browse X
    end
  end
  thread local Y in
    Y = 2
    skip Browse Y
    skip Basic
            skip Browse Y
    skip 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
```

d. //Quantum of 3 is not causing a suspension because the browse statement is executed after the thread is completely executed
//minimum quantum is 5

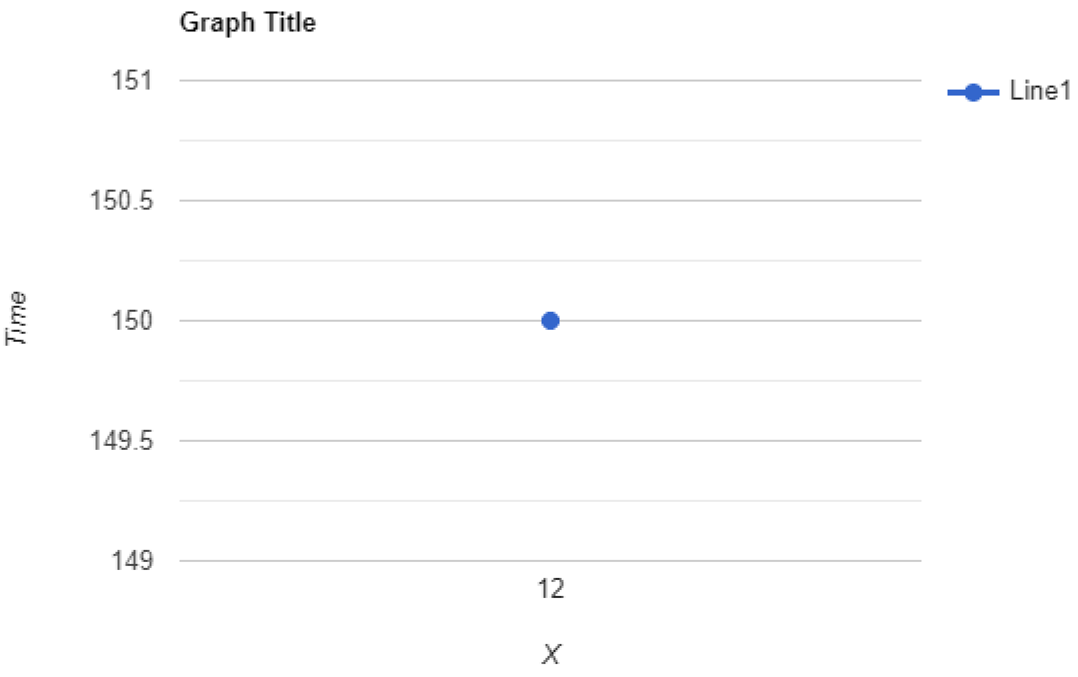
e. **Fib1:**



Fib2:


Graph Title
Line1
Time
0.1
0.08
0.06
0.04
0.02
0
12
13
14
15
16
17
X

FibThread:


Graph Title
Line1
Time
151
150.5
150
149.5
149
12
X

5a) Fib1 has an exponential time complexity, Fib2 has a linear and incrementive time complexity, fibThread has an insane time complexity that also looks exponential for both iterations provided.

```
//1 = 0
//2 = .1   5 suspended
//3 = .41  24 suspended
//4 = .76  48 suspended
//5 = 1.53 91 suspended
//6 = 2.61
//7 = 4.70
//8 = 7.69

//12 = 65.44
```

The suspended threads should give an insight to how many threads were being made with our quantum being finite 1, I do not see much of a pattern emerging at all, maybe 2N?.

2.

a.

```
local Producer OddFilter Filter N L P F in
  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

 OddFilter = proc {$ P Out}
    Filter = fun {$ O1 T1}
    case O1 of nil then T1
      [] '|'(1:H 2:T) then S in
        if ((H mod 2) == 1) then
          S = {Filter T T1}
          S
        else
          S = {Filter T T1} (H|S)
        end
      end
    end
    Out = {Filter P nil}
  end
// Example Testing
N = 0
```

```
L = 100
{Producer N L P}
{OddFilter P F}
skip Browse F
end
```

b.

```
local Generate Sum D in
  fun {Generate N Limit}
            if (N<Limit) then
            (N|{Generate (N+1) Limit})
            else nil end
      end
      fun {Sum Xs A}
            case Xs
                  of (X|Xr) then {Sum Xr(A+X)} else case Xs of nil then A else nil
end
            end
      end
      local Xs S in
             Xs = {Generate 0 5} //Producer thread
             S =  {Sum Xs 0} // Consumer thread
             skip Browse S
      end
end
```

c.

```
local Generate Sum in

  fun {Generate N Limit}
  if (N<Limit) then
  (N|{Generate (N+2) Limit})
  else nil end
end
fun {Sum Xs A}
  case Xs
    of (X|Xr) then {Sum Xr(A+X)} else case Xs of nil then A else nil end
  end
end
local Xs S in
  thread Xs = {Generate 0 101} end//Producer thread
  thread S =  {Sum Xs 0} skip Browse S end// Consumer thread

end
```

end