```
-- input parties
principal A
principal B
-- compute parties
principal C
principal D
-- output parties
principal E
principal F

def cmp : ℤ{isec:A,B} →{inp:A,B} 𝔹{yao:C,D}
def cmp = λ xy →
  let x : ℤ{yao:C,D}
  let x = share{yao:C,D} xy.A
  let y : ℤ{yao:A,B}
  let y = share{yao:C,D} xy.B
  let r : 𝔹{yao:C,D}
  let r = x ≤ y
  in r

def cmp-mpc : 𝟙 →{inp:A,B;rev:E,F} 𝔹{ssec:E,F}
def cmp-mpc = λ • →
  let xy : ℤ{isec:A,B}
  let xy = {par:A,B} read ℤ "e1-input.txt"
  let r : 𝔹{yao:C,D}
  let r = cmp xy
  let o : 𝔹{ssec:E,F}
  let o = reveal{E,F} r
  in o

def one-liner : 𝟙 →{inp:A,B;rev:E,F} 𝔹{ssec:E,F}
def one-liner = λ • →
  let xy = {par:A,B} read ℤ "e1-input.txt"
  in reveal{E,F} (share{yao:C,D} xy.A) ≤ (share{yao:C,D} xy.B)

def main : 𝔹{ssec:A,B} × 𝔹{ssec:A,B}
def main = cmp-mpc • , one-liner •
```