#CLOJURECODES..<n> AGAINST AN ANTI-NSA NETWORK WHICH CLAIMED TO ACCORDING TO MEDIA SOURCES BE FUNNELING MONEY AND INVESTMENT TRIES FROM A && B BANK AMONG OTHERS TO TRY AND USE GUN WEAPONS TO GO AGAINST ANTI-TERROR OPERATIVES FROM NSA. THESE MONIES ACCORDING TO MEDIA SOURCES WERE FUNNELED FROM NICARAGUA AND WENT THROUGH CHICAGO AND NEW YORK EARLIER THIS YEAR. THE ANTI-NSA NETWORK DID NOT SHOW UP FOR COURT BUT FLED TO NON-EXTRADITION COUNTRIES.

<nt.clojure:lib:::Solver.Multiply.Bug.Fix.clojure.cl-CL::lib::cool.lib.cl.n>

user -> (toString [Infixer::314a] ) :: a && b \_h + t == a not= {((toString))}}} }}

#CLOJURECOMMENT: THIS defrecord RETAINS THE KEY VALUE PAIRS FOR

A && B PAIRS KEYS REDUCE. THIS ALLOWS THE ABPair seq TO MAP VECTOR.

defrecord (:h Pair -> Pair.objective.a && b:: [h t]) n-1::a| a \*\*b Pair

{[ h t ]}..map vector { 354:2, redondo:8 } keys reduce { [ h t ]} Pair

(:t Pair2 -> /a ?b && a ?? <n> assoc {( disassoc: Pair [ a && b].Pairsecurity:t-5%mod::: ?? a if not= redondo then Pair

Otherwise a ? b == assoc::keys:reobjective.Pair [ h t ]

map vector remap::: assoc {( disassoc: not=Pair + Pair == a && b

deftype ((-h Pair.vector.objective:::defsecurity const a && b { [ keyword/76assign.vector#$mod3 ] } filter (kw/#\*assign\*) {[ h t ] }}

let (assoc(keys::reobjectivePair.mod3)))

toString[ h t ] kw:sym:::/2u34 retoString[reobjective.Pair] filter h

assoc: defrecord ::: kw: sym

disassoc ::telephony(Suite2u34$#assoc::keys-Repeat))))

Pair.obj.val==+disassoc:::toString [ h t ]:::filter map keys:::$mod3

obj.connection::deftype:::defretoString[time and date].Pair filter nil = 2

defMacro.Object.type:::

seq ( vector map:: {[(((3)))]}}} n seq ((permN:::n-10..n)n+4$mod3-3+E))) reseq(kw:permutatedseq:::n for if key-in == vector {n} then assoc:::keys merge to sym::: kw:::resequence the defrecordobj.connection connection flush close sym:::2

defExpensiveT==deftype(toString[expenseNewexpensesaveexpense connection

let (Newexpense save time date sym) {n} vector: map

reduce map

{n}

flush.connection { close sym ] } n

close.connection {saveNewExpensesave connection permN:::n-10..n)n+4 kw:Upmap30..n..merge to sym::: kw:::resequence the Obj.record.close.flush.connection (Newexpense save time date sym) {n} vector: map reduce map-1 +2

flush.connection connection save {n}

close.connection connection.NewExpensesave.permN<n>?a&&b==2

Telephonycellulag2@reExpensesave connection reduce map::: vector: map

{n}

close resequence merge to Obj.record

user -> (Infixer:::infix map vector {{{3-4$..n}} } n

close.user->Infixer:::map

useful:::callenderexpensivevec:::{@emergcarambul. Connection reduce close flush connection useful callenderexpensivevec:::{{@emerg&\*}]a [ a&&b==2][.Obj.str.val==+.useful.Telephonycellulag2@save.&\*lag2.cl.nt.cl](mailto:.Obj.str.val==+.useful.Telephonycellulag2@save.&*lag2.cl.nt.cl) connection:::n

let (connection Obj.str.val-1==+2.close.flush connectsave.\*lag==save. {n}

( time date [3+1@save.cl==3)..n<n](mailto:3+1@save.cl==3)..n%3cn)> Obj.record==Infixer.map

[Temp==normaltemp2@+1@merge.Newtemp=1](mailto:Temp==normaltemp2@+1@merge.Newtemp=1) reduce a && c c=1+

let temp connection close flush Obj.record==Infixer.Temp(map: vector) map: vector

<n> {n} ..

<n kw:calendar#3>

<html:n>