CSE324 Quiz #5 Name----------**KEY**----------

True or False: (questions are about FORTRAN & Algol, early versions , ONLY)

a) Algol is more powerful than FORTRAN. T
Algol has Dynamic arrays, by-name, rich typing system, etc.

b) Upon calling a procedure/function that does not define any local names, an activation record (AR) will not be pushed into the runtime stack, in such special case. F
NOP! An AR is created regardless, for other duties as explained in class!

c) "***Recursion***" and ***dynamic arrays*** in Algol are examples of gaining **power** for loss of **security**. F
NOP! power versus efficiency, not security, both mechanisms are secure in Algol.

d) Algol is a ***block-structured*** imperative High Level Language. T

e) Unlike FORTRAN, Algol designers NEVER trusted the users to do the right thing, hence the language is secure! F
NOP! The trusted users to do the correct by-name parameter passing!

f) Algol's ***dynamic arrays*** alleviated the problem of having a lack of ***pointers*** to facilitate the implementation of dynamic data structures, e.g., lists, stacks, and queues.
T The use of arrays, dynamic arrays, pointers in dynamic ADTs is explained in class.

g) Algol's ***typing system*** is ***insecure*** due to the use of ***by-name*** parameter passing. F
By-name is not related to the typing system!

h) The ***by-name*** parameter passing facility in Algol is an examples of gaining **power** and losing **security** and **efficiency**. T

i) In Algol, ***nested*** *block*s inside *procedure*s might be called from any place in the code, yet according to the contour diagram rules in statically scoped languages. F
Blocks are NEVER called, they are only **in-place** static abstraction.

j) In Algol, the ***type*** declaration section is ***elaborated*** at run time, for better ***efficiency***. F

Extra credits (if correct you **get +5 points** for each correct answer, otherwise you **lose -5 points**):

k) The ***in*security** of Algol's ***by-name*** will be solved if ***thunks*** are evaluated **only once,** at the caller environment, which won't affect the ***by-name***'s **power**. F
NOP! It will violate its semantics & integrity since it won't be powerful anymore!

l) In Algol, in some special scenarios, the AR of the ***caller*** as well as the ***definer*** of any invoked (called) procedure ***P*** (callee) are **NOT** always guaranteed to be pushed/exist in the run time stack when ***P*** is called, i.e., below ***P***'s AR. F
NO way, not in statically scoped and stack model of computation HLLs, like Algol!