CSE324 Quiz #3 Name: ---------**KEY**---------

1) The Declaration section of a HHL will be read by the compiler in static typed HLLs and by the CPU in case of dynamically typed HLLs.
NEVER by the CPU!

2) The ***implicit*** name declaration feature in FORTRAN can easily lead to ***aliasing*** problem which is considered a **security loophole** in the language.
There is no relation between the two features!

3) FORTRAN is one of the pioneers HLLs that introduced "True" built-in Abstract Data Types. T
Sure! All operations on Integers/Real are true built-in ADTs!

4) The major design goal of the early FORTRAN was to minimize the involvement of the operating system at run time, for better execution speed. T
Minimize dynamic memory allocation, i.e., min the OS memory mgmt. involvement.

5) In early FORTRAN**, overworking** the INTEGER type led to **security loophole**. T

6) ***Static*** type checking is more **secure** than ***dynamic*** type checking.
NOP, more efficient maybe, but not more secure!

7) A language with a **secure** typing system is **not** always **secure.** T
There are some other none typing system related issues that might cause a security loophole,

8) In general, a HLL might gain in the direction of **power** when losing in its typing system **efficiency**. T
Sure! Dynamic typing is more polymorphic/powerful.

9) A HLL with very **rich** set of types will always have a **secure** typing system.

NOP! You still need a strong type checking system!

12) In general, a HLL ***by-reference*** parameter passing mechanism is **more efficient** than ***by-value***.
NOP, more efficient only when passing a long aggregate data, not scalars.

13) When the semantics of HLL statements are very different, their corresponding syntaxes must be totally different, otherwise there a potential of **security loophole**. T
Sure! It is very easy to move from one syntax to another for a totally different semantics.

14) In a HLL with ***global*** name declaration, the use of ***by-reference*** parameter passing mechanism will lead to a **security loophole** in the language. T
True! Aliasing more than one name on the same memory location is dangerous, yet powerfull!

15) A language is ***insecure*** when a any typing ***error*** escapes both lines of defense: the compiler and run time. T