CSE324 Quiz #2 Name: ------**KEY**----------

True or False(T/F): **Reds are True**

1) The ***imperative*** domain is all about "*What*" more than "*How*" to carry out the code execution, i.e., no *true* data abstraction. **(This is true for the *logic* paradigm!)**

2) For better efficiency, compilers carry out the *semantics* analysis first before *syntax* analysis, since statements might be semantically correct, yet syntactically invalid.
(syntax first before semantics CFG rules then CSG kicks in)

3) The compiler builds the code *Parse* Tree **just** after the *semantics* analysis phase. **Nop! after the Syntax analysis phase**

4) In the ***pure*** functional paradigm, there is no notion of intermediate memory side effect (i.e., change of **memory** state), while executing code; also *recursion* replaces *iteration*. **(yes, as defined in class)**

5) Definitely, the most useful language of all is Prolog! **(The application domain decide the usefulness of the use!)**

6) One of the major differences between the *Block-Structured* and *Object Oriented* paradigms is their view of data nature & manipulation. ***(sure! passive vs. alive)***

7) In general, HLLs are much more *readable* than Low Level Languages, yet they tend to be *slower* to execute.

8) Interpreted HLLs codes are *slower* to execute than compiled **HLL's codes**.

9) In the pure ***logic*** paradigm, there is a total separation between the ***logic*** and ***control*** of the executing code.

10) In ***pure* Object Oriented** paradigm, the ***abstraction*** of a data type is inherently *built in* the type itself (*object's* ***behavior&state*)**, hence the security of the **data** manipulation is not added later **(ad hoc)** through the coding.

11) All compiler generated machine codes (final & intermediate) are executable over the hardware given that they go first through the load&link phases before they **are run by the system CPU**. ***(True for final not intermediate!)***

12) For the Java HLL, it is enough to generate the intermediate compiled *byte code* to make the language portable over different hardware platforms. ***(Nop, you still need the Java VM interpreter)***

13) Since the language is close to the hardware, Assemblers' **output** machine code is directly executable over the hardware. ***(Nop, you still need go through the link/load phases)***

14) Compilers are, sometimes, **are** able to detect run time *dynamic semantics* **errors** at compile time.
(Nop, if so we can solve the unsolvable Halting problem!)

15) The *undefined symbol* error is detectable at the compiler *lexical* analysis phase, whereas **the error due to the** use of *undeclared name* (id) is detectable at the *semantics* analysis phase.