CSE324 Quiz #**8** Name:_____**KEY**_____

True or False: (all questions are about **Ada**), one point for very True/False question.

1) The Ada's ***rendezvous*** protocol involves one-way **messaging** mechanisms: *client- server*, to exchange messages. **T**

2) The ***rendezvous*** mechanism is to be used between *clients* and *servers* tasks. **T**

3) It is **not** always the case that a ***call*** to a *server*'s ***entry***, by some *client* task, must be **accepted** by the server, before any service is granted. **F**

4) When handling any none-locally defined ***exception***, the system will keep on the stack all activation records that did not define the corresponding exception handling code (EHC). **F (Pops all not keep)**

5) A task's ***entry*** is not the same (semantically) as a regular Ada procedure call. **T**

6) In some cases, more than one server will guard the same critical shared memory among a number of communicating tasks, for more efficiency (faster service). **F**

7) For better **efficiency**, a server task might maintain **two or more** entries for the same service to guarantee the security of guarding the shared memory. **F**
One critical code per every shared area, otherwise chaotic buffer state!

8) There is no explicit "***call***" statement to invoke a ***task***. **T**

9) The use of the ***rendezvous*** protocol is to support/facilitate the security of communicating tasks, procedures, and functions that share/access a common memory buffer. **F (only tasks)**

10) Java and C++ have exception handling mechanisms. **T (Yes, "try" & "catch")**

11) A CONSTRAINT_ERROR will be raised when a client attempt to execute an empty "entry" of a server. **F**
(What?!?! The statement doesn't make sense at all)

12) The ***select*** clause will terminate when all clients terminate tasks terminate. **T**

13) The ***exception*** handling mechanism will follow the ***dynamic*** chain to find the place of any raised name of a ***none-local*** imbedded Exception Handler Routine in the code. **T**

14) The exception handlers might be declared locally, in the ***callee's*** codes and/or globally at the calling sequence/hierarchy, i.e., caller, caller of the caller, and so on. **T**

15) ***Exceptions*** are *synchronous* in nature; they occur at a fixed specific time like the operating system's ***traps*** mechanism. **F (synchronous, like OS S/W traps)**

16) Ada parameter passing mechanism is a again toward *security* with some loss of *efficiency*. **F**
Actually, the compiler not the user will do it in the most efficient way!

17) An Ada "***in***" parameter is not suppose to be changed in the callee at all. **T**

18) An Ada "***out***" parameter must change inside the callee. **F (not must!)**

19) An Ada "***in-out***" is used to send and return values in/out to/of the callee, respectively. **T**

20) Ada parameter passing mechanism is more ***abstract*** the peers in Algol and Pascal. **T**

Extra Credit: (**gain 3**points for correct answer, or **lose 2** points for wrong answer)

21) Ada tasking facility guarantees mutual exclusive access to critical codes, yet it is **not** really **very fair** to clients! **T**

22) The ***select*** clause makes the serving of clients asynchronous, alleviating the client service "***starvation***" problem. **T**

23) It is more efficient for a HLL program not to wait for the *exceptional* event to occur, just preemptively do something about it. **T**