HAL/S PROGRAMMING FUNDAMENTALS
QUIZ 9

1. True or False. HAL/S will automatically convert BIT strings to INTEGER (and vice-versa) in assignment statements, comparisons, and parameter passing.
Ans. _____

2. True or False. Conversion of BIT strings to INTEGER will always result in unsigned (positive) integers. Ans. _____

3. True or False. The INTEGER and SCALAR conversion/shaping functions allow the construction of 1-, 2-, and 3- dimensional arrays of either single or double precision.
Ans. _____

4. True or False. The INTEGER and SCALAR conversion/shaping functions will accept arguments of INTEGER, SCALAR, VECTOR, and MATRIX data types. BIT and CHARACTER strings may be contained as well. Also, any of these data types can be arrayed. Ans. _____

5. True or False. The BIT conversion/shaping function always produces a 32-bit bit string (which is the maximum size for a bit string). Ans. _____

6. True or False. The BIT conversion/shaping function can be used to produce 1-, 2-, or 3-dimensional arrays of bit strings, but no precision qualifier can be specified since it is meaningless. Ans. _____

7. True or False. The BIT conversion/shaping function cannot accept VECTOR or MATRIX data types as input. Ans. _____

8. True or False. Although the BIT conversion/shaping function cannot have a precision qualifier in its subscript position, it can contain a component subscript which thereby allows bit strings of less than 32 bits to be produced. Ans. _____

9. True or False. The RADIX form of the BIT conversion/shaping function is used to convert CHARACTER strings into BIT strings. The RADIX may be @BIN, @DEC, @OCT, or @HEX and is supplied in the subscript position. Ans. _____

10. True or False. Like the BIT conversion/shaping function the CHARACTER conversion/shaping function cannot accept VECTOR or MATRIX data types.
Ans. _____

11. True or False. The only way that an array can appear in a BIT or CHARACTER conversion/shaping function is if the function is participating in an "arrayed" assignment or comparison operation. In this case each iteration through the array causes the conversion of the next element. Ans. _____

12. True or False. The CHARACTER conversion/shaping function produces a character string that is exactly long enough to hold the converted data item (up to a max of 255 characters of course). It may be component-subscripted, however, in the same manner as the BIT conversion/shaping function so that a smaller character string is produced. Ans. _____

13. True or False. When BIT strings are converted to CHARACTER via the CHARACTER conversion/shaping function the subscript position can contain a RADIX of @DEC, @HEX, @OCT, or @BIN to specify the desired form of the output. In this case no component subscripting can be performed. Ans. _____

14. True or False. If not overridden by the RIGID keyword, HAL/S attempts to sort declared data items in order to group unarrayed (simple) integers, scalars, bit strings, etc., ahead of aggregate data items like arrays, vectors, matrices, and multicopy structures. Within a sort grouping HAL/S tries to collect data items that have similar memory boundary requirements. Ans. _____

15. True or False. The RIGID keyword may be applied only to COMPOOL block header definitions, STRUCTURE templates, and Minor Structures. Ans. _____

16. True or False. If the RIGID keyword is applied to a structure template then the entire contents of the structure are rigid (allocated precisely in the declared order). If the RIGID keyword is applied to a minor structure, however, then only that substructure is RIGID. If the RIGID keyword is applied to a structure template and the NONRIGID keyword is applied to one of the minor structures, then that minor structure may have its elements sorted by the compiler. Ans. _____

17. True or False. HAL/S always places structures (and all minor structures) at specific memory boundaries (e.g., addresses divisible by 2). Ans. _____

18. True or False. Use of the RIGID keyword may result in "wasted" memory space since the compiler may have to skip over one or more halfwords ("alignment gaps") in order to align the next data item on a suitable memory boundary. Ans. _____

19. True or False. The HAL/S compiler shows memory maps (including alignment gaps) in its Phase 2 (code) listing. Ans. _____

20. True or False. In a non-RIGID structure template, HAL/S will tend to sort structure terminals for optimal memory packing – however, it always keeps data belonging to a given minor structure from mixing with the data of other minor structures. Ans. _____

21. True or False. A multi-copy structure (even if not RIGID) may have wasted space (alignment gaps) between each individual copy. Ans. _____

22. True or False. If the RIGID keyword is applied to a COMPOOL block header, then any structure templates contained within that COMPOOL will be treated as RIGID also, i.e., their terminals will be allocated in the declare order. Ans. _____

23. What are the number of AP-101 halfwords occupied by the following data items?

INTEGER	Ans. _____
INTEGER DOUBLE	Ans. _____
SCALAR	Ans. _____
SCALAR DOUBLE	Ans. _____
BOOLEAN	Ans. _____
BIT(7)	Ans. _____
BIT(16)	Ans. _____
BIT(17)	Ans. _____
BIT(32)	Ans. _____
VECTOR(3)	Ans. _____
CHARACTER(5)	Ans. _____

24. What is the value of BVAR$1 in the following case? Ans. _____

```
DECLARE  BVAR  BIT(12) INITIAL(BIN'0110010101110');
```

25. True or False. BIT strings are never initialized by the compiler. Although they are right-justified in a AP-101 halfword (or fullword), the compiler always "masks" when it loads the bit string into a register so that any unallocated bits are safely ignored. Ans. _____

26. True or False. NAME variables (pointers) always occupy a single AP-101 halfword no matter what kind of data item they point to. NAME REMOTE variables (long pointers) always occupy an AP-101 fullword (2 halfwords) – again, regardless of what kind of data item they point to. Ans. _____

27. True or False. EVENT variables act like BOOLEAN variables except that they have a special structure (defined in the HAL/S-FCOS ICD). They reside in a single halfword with the upper 15 bits reserved for FCOS use and the low bit containing a 0 or 1. Ans. _____

28. True or False. The opposite of the DENSE keyword is ALIGNED, and the opposite of the STATIC keyword is AUTOMATIC. Ans. _____

29. True or False. If the keyword DENSE is applied to a COMPOOL block header then all bit strings within the COMPOOL will be packed with shorter bit strings sharing a single halfword or fullword. Ans. _____

30. True or False. If the keyword DENSE is applied to a structure template then it affects the entire template unless it is overridden by the ALIGNED keyword appearing on a minor structure – or on a bit string terminal. Ans. _____

31. True or False. The DENSE keyword cannot be applied to a structure – only to the template. Furthermore, if it is applied to any data item other than a bit string it is simply ignored. And it is even ignored for bit strings unless they are defined within a structure template. Ans. _____

32. True or False. ALIGNED and STATIC are default attributes. It is therefore pointless to ever specify STATIC. The ALIGNED keyword can be useful, however, to locally "turn off" dense packing of bit strings within a DENSE structure template. Ans. _____

33. True or False. DENSE is only effective on the bit strings contained within a single minor structure (or the entire structure if it has no 'forks' or minor structures). Bit strings belonging to 2 different minor structures will never be "packed" with one another. Ans. _____

34. True or False. If a structure template (or minor structure) is declared RIGID then the benefit of DENSE bit string packing will likely be reduced. Ans. _____

35. True or False. DENSE bit strings may be utilized just like normal bit strings. They not only save memory but also generally provide speedier code. Ans. _____

36. Which of the following are FALSE statements about TEMPORARY variables?

a. TEMPORARY variables can be defined within any DO … END group except for a DO CASE.

b. In the form "DO FOR TEMPORARY I = …", the variable I will be equivalent to an INTEGER DOUBLE and will possibly be maintained solely within a register and have no allocated memory address.

c. The names of TEMPORARY data items must be unique within the DO…END group within which they are defined so that other TEMPORARY data items in other DO…END groups can have the same name. However, no TEMPORARY can have the same name as a normal declared data item that is visible by the name scoping rules.

d. TEMPORARY data items may have INITIAL or CONSTANT initialization but only if they are declared as AUTOMATIC.

e. EVENT variables and NAME variables cannot be defined as TEMPORARY.

Ans. _____

37. True or False. A structure variable can be defined as a TEMPORARY, but this requires that the template being referenced is contained within some previous DECLARE group. Ans. _____

38. True or False. TEMPORARY variables are advantageous in that they may reduce the overall data memory requirements of a code block. This is because such variables are allocated within a run-time stack and may actually overlay each other.

Ans. _____

39. True or False. Access to TEMPORARY variables is generally as fast as access to normal declared data items. Ans. _____

40. True or False. A stack "walk-back loop" is produced when a stack-resident variable (a procedure parameter or TEMPORARY variable) is "scoped-into" a nested procedure or function. Ans. _____

41. True or False. HAL/S LOCK groups, UPDATE blocks, and TASK blocks are not utilized by FSW. Ans. _____

42. True or False. To make a HAL/S PROCEDURE or FUNCTION "Exclusive" simply requires that the EXCLUSIVE keyword be appended to the PROCEDURE or FUNCTION block definition and that all declared data be marked as STATIC.

Ans. _____

43. True or False. EXCLUSIVE procedures and functions require calls to the FCOS at their entry and exit points so that FCOS can guarantee that only one process (HAL/S PROGRAM) has access to that resource. Ans. _____

44. True or False. In order to make a PROCEDURE or FUNCTION truly reentrant simply requires that the keyword REENTRANT be appended to its block definition. REENTRANT blocks have no interface with the FCOS. Ans. _____