1. True or False. HAL/S NAME variables act like pointer variables in other programming languages. They are declared just like normal variables except that the keyword NAME must precede all attributes of the variable. Ans. __________

2. True or False. A NAME variable can only “point at” variables with exactly the same attributes, except that NAME INTEGER variables can point at SCALARs and NAME SCALAR variables can point at INTEGERS since HAL/S supports implicit type conversions between INTEGER and SCALAR. Ans. __________

3. True or False. A NAME variable occupies 1 halfword of memory and can address data residing in AP-101 sectors 0 plus one other sector (usually sector 1). If you wish to point at data residing in arbitrary sectors then you need to create a NAME REMOTE variable. Ans. __________

4. True or False. A NAME REMOTE variable occupies 2 halfwords of memory and it is created by appending the keyword REMOTE to a normal NAME declaration, e.g.,

   DECLARE VPTR NAME VECTOR DOUBLE REMOTE;     Ans. __________

5. True or False. Either of the following two forms will initialize a NAME variable to a null (zero) value:

   DECLARE SPTR NAME SCALAR INITIAL(NULL);     Ans. __________
   DECLARE SPTR NAME SCALAR INITIAL(NAME(NULL));

6. True or False. Either of the following two forms will assign a NAME variable to point to a null (zero) value:

   NAME(SPTR) = NULL;     Ans. __________
   NAME(SPTR) = NAME(NULL);

7. True or False. There are no illegal 16-bit addresses on the AP-101 so a NAME variable initialized to NULL (zero) is still a valid pointer (although probably not what you want). Ans. __________

8. True or False. The following code is valid:

   DECLARE T, N1 NAME, N2 NAME INITIAL(NAME(T));
   T = 2;
   NAME(N1) = NAME(N2);     Ans. __________
9. Which of the following are “dereferenced” usages of a NAME variable?

Ans. __________

DECLARE MATRIX, MAT INITIAL(3#(1,1,0)), VMAT NAME INITIAL(NAME(MAT));

a) NAME(VMAT) = NAME(MAT);
b) VMAT$(1,1) = 2;
c) CALL PROC1(NAME(VMAT));
d) CALL PROC2(VMAT);
e) VMAT = MAT;

10. True or False. A COMPOOL is normally allocated in sector 0 or 1 via Linkage Editor Concards. If it is nonetheless allocated in some other sector, then the COMPOOL must be included with the REMOTE attribute.

Ans. __________

D INCLUDE TEMPLATE CPOOL REMOTE

11. True or False. If a COMPOOL has been defined to be REMOTE, then only NAME REMOTE name variables can be used to “point to” its data.

Ans. __________

12. True or False. Data declared within a PROGRAM or separately compiled PROCEDURE or FUNCTION is allocated within a CSECT whose name begins with the characters “#D”.

Ans. __________

13. True or False. PROGRAM/PROCEDURE/FUNCTION data is normally allocated in sector 0 or 1, Variables contained within this data area can be accessed with standard 16-bit NAME variables.

Ans. __________

14. True or False. If either of the two following compiler directives is used, then the PROGRAM/PROCEDURE/FUNCTION data “may” be placed in some sector other than 0 or 1.

Ans. __________

D DATA_REMOTE
DATAREMOTE

15. True or False. If #D data is allocated remotely then only NAME REMOTE type name variables can be used internally to the PROGRAM/PROCEDURE/FUNCTION to point to such data.

Ans. __________
16. True or False. The following is a correct example of a non-dereferenced use of a NAME variable:

```
PROG: PROGRAM;
    DECLARE MATRIX, MAT, NMAT NAME;
    MAT = MATRIX(1,2,3,4,5,6,7,8,9);
    NAME(NMAT) = NAME(MAT);
    ...
    CALL PROC1(NAME(NMAT));
    ...
PROC1: PROCEDURE(PMAT);
    DECLARE PMAT NAME MATRIX;
    ...
CLOSE PROC1;
```

Ans. __________

17. True or False. The following is a correct example of a non-dereferenced use of a NAME REMOTE variable:

```
DATA_REMOTE
PROG: PROGRAM;
    DECLARE MATRIX, MAT, NMAT NAME REMOTE;
    MAT = MATRIX(1,2,3,4,5,6,7,8,9);
    NAME(NMAT) = NAME(MAT);
    CALL PROC1(NAME(NMAT));
PROC1: PROCEDURE(PMAT);
    DECLARE PMAT NAME MATRIX;
    ...
CLOSE PROC1;
```

Ans. __________

18. True or False. In the following example a dereferenced name variable is passed to PROC1.

```
DATA_REMOTE
PROG: PROGRAM;
    DECLARE MATRIX, MAT, NMAT NAME REMOTE;
    MAT = MATRIX(1,2,3,4,5,6,7,8,9);
    NAME(NMAT) = NAME(MAT);
    CALL PROC1(NMAT);
PROC1: PROCEDURE(PMAT);
    DECLARE PMAT MATRIX;
    ...
CLOSE PROC1;
```

Ans. __________
19. True or False. The following INCLUDE directive is legal and it causes only 3 symbols to be included from the COMPOOL CPOOL1 rather than all symbols. Ans. __________

D INCLUDE SDF CPOOL1 REMOTE: STRUCTURE Q, Q1, Q2;

20. True or False. NAME variables cannot be assigned REMOTE data and NAME REMOTE variables cannot be assigned to non-REMOTE data: Ans. __________

DECLARE DPTR1 NAME MATRIX INITIAL(NULL);
DECLARE R PTR1 NAME MATRIX REMOTE INITIAL(NULL);
NAME(DPTR1) = NAME(RPTR1); <illegal?>
NAME(RPTR1) = NAME(DPTR1); <illegal?>

21. True or False. Although NAME variables can normally only point at data with IDENTICAL attributes, two %MACROs exist that allow this protection to be bypassed. These %MACROs are %NAMECOPY and %NAMEADD. Ans. __________

22. True or False. %NAMEADD is a safer (simpler and less risky) %macro than is %NAMECOPY. Ans. __________

23. True or False. The syntax of %NAMECOPY is %NAMECOPY( SPTR1,SPTR2). Both SPTR1 and SPTR2 must be NAME variables of STRUCTURE type, but may have been defined with different structure templates. Ans. __________

24. True or False. In a %NAMECOPY either both operands must be non-remote, or both must be REMOTE. REMOTE and non-remote cannot be mixed. Ans. __________

25. True or False. In a %NAMECOPY the second argument can either be a NAME structure variable or just a structure variable. The first argument must always be a NAME structure, however. Ans. __________

26. True or False. The major usage of %NAMECOPY is to allow the “reuse” of a data or buffer area by being able to describe it with different layouts. Ans. __________

27. True or False. Unless a %NAMECOPY is used correctly, it is possible to access data “beyond” the actual bounds of the buffer area, i.e., there are no compile-time or run-time checks to ensure data integrity. Ans. __________
28. True or False. At run-time, both of the following NAME variables (NVARR and NS) will contain identical AP-101 addresses.

Ans. _________

```
DECLARE VARR ARRAY(10) SCALAR INITIAL(10#1);;
DECLARE NVARR NAME ARRAY(10) SCALAR;
DECLARE NS NAME SCALAR;
NAME(NVARR) = NAME(VARR);
NAME(NS) = NAME(VARR$1);
```

29. True or False. In the following case, NQ will contain the address of a “fictitious” zeroth-copy (0th) of the multicopy structure Q.

Ans. _________

```
STRUCTURE Q:
  1  A,  2  S1,  2  S2,  1  B,  2  IVAL INTEGER;
DECLARE Q Q-STRUCTURE(10);
DECLARE NQ NAME Q-STRUCTURE(10);
NAME(NQ) = NAME(Q);
```

30. True or False. HAL/S makes use of a fictitious 0th-element for all aggregate data variables, i.e., arrays, multicopy structures, vectors and matrices. Since HAL/S subscripting always starts from 1 this makes it unnecessary for the generated code to always have to subtract 1 before applying a subscript.

Ans. _________

31. True or False. Whether the address of an aggregate data item is contained in a NAME (or NAME REMOTE), or is being passed by “reference” to a PROCEDURE or FUNCTION, the 0th-element address value is always employed.

Ans. _________

32. True or False. If both arguments of a %NAMECOPY refer to multi-copy structures, the compiler will generate appropriate code to ensure that the address of the target NAME structure is correctly biased so that the 1st copy begins right at the beginning of the data area.

Ans. _________

```
STRUCTURE Q1:
  1  BUFF1 ARRAY(4) SCALAR DOUBLE,
  2  BUFF2 MATRIX;
DECLARE Q1 Q1-STRUCTURE(10);
DECLARE NQ1 NAME Q1-STRUCTURE(10) INITIAL(NAME(Q1));
STRUCTURE Q2:
  1  BUFFX ARRAY(17) SCALAR DOUBLE;
DECLARE NQ2 NAME Q2-STRUCTURE(5);
%NAMECOPY(NQ2,Q1);
%NAMECOPY(NQ2,NQ1);
```
33. True or False. %NAMEADD is a 3-argument %macro that should be used with great caution since the compiler DOES NOT automatically adjust addresses to reflect 0th-element addressing. Ans. __________

34. True or False. Unlike %NAMECOPY, %NAMEADD permits mixing REMOTE and non-REMOTE variables. Ans. __________

35. True or False. The form %NAMEADD(NV,V,2); is equivalent to the following (and quite illegal) HAL/S statement. NAME(NV) = NAME(V) + 2; Ans. __________

36. True or False. The expression NAME(VAR) always produces the address of a fictitious 0th-element in the event that VAR has components, e.g., is a multicopy structure, array, vector, or matrix. Ans. __________

37. True or False. In a %NAMEADD it is the programmer’s responsibility to ensure that the increment (or decrement) added to a NAME variable is correctly calculated so that the address stuffed into the receiving NAME variable correctly points to its 0th-element in case it also is an aggregate data item. Ans. __________

38. True or False. The following %NAMEADD will correctly reuse the buffer SBUF. DECLARE SBUF ARRAY(100) SCALAR DOUBLE; DECLARE PBUF NAME ARRAY(400) INTEGER; %NAMEADD(PBUF,SBUF,3); Ans. __________

39. True or False. A negative literal integer may be used in a %NAMEADD as long as the resultant address doesn’t cross an AP-101 sector boundary. %NAMEADD(NV1,NV2,-32); Ans. __________

40. True or False. It may be a problem for the Linkage Editor if a HAL/S aggregate variable lies in one sector, but its 0th-element lies near the end of the preceding sector. Ans. __________