Grading Key for NS-Diagram Assignment

**DB Doesn’t Belong**

The item in the “Statement Box” doesn’t belong in the box.

What goes in a “Statement Box”?

- An INVOKED PROCEDURE name (explained on a separate page)
- An ASSIGNMENT statement \( x=3, \ y=y+1 \)
- An INPUT operation \( \text{Read } x, \ y, \ z \quad x=\text{Cells}(1,1).\text{Value} \)
- An OUTPUT operation \( \text{Print } x,y,z \quad \text{Cells}(1,1).\text{Value}=x \)
- A CONTROL STRUCTURE (IF Construct, CASE Construct, LOOP Construct)
- NOTHING ELSE

What other kind of “areas” (boxes) are part of a NS Diagram?

- CONTROL BOX One of the “odd shaped” non-rectangular areas of the control structure.

What goes in a “Control Box”

- Control Information
  - “UNTIL \( x<6 \)”
  - “IF \( x\text{Tol}>\text{specTol} \)”
  - “FOR \( i=1 \) to 10 STEP 2”
- Control Labels
  - T/F
  - Start/End
  - Ret/Return
- NOTHING ELSE
IS Invalid Structure

The structure drawn is not one of the standard forms and therefore cannot be constructed from available coding tools (Computer Language Elements).

Area “8” is an invalid loop of some type.
Notice the word “END” is DB.
Notice “dA” is not calculated anywhere… and hence cannot be controlling.
Notice A1 and A2 are not calculated anywhere.
Notice “Determine Function Area with double the min number of iterations” is NOT a well selected name.
Another example:

\[ dA = B_7 - A_7 \text{ from previous} \]

\[ \Delta A = E_{11} - dA \]

\[ \text{If } \Delta A = (-) \rightarrow \text{ Recalculate (F)} \]
\[ \text{If } \Delta A = (+) \rightarrow \text{ End} \]

Notice the “box” at the bottom contains improper items (if you want two “if” statements they should appear as IF CONSTRUCTS.
Notice: What happens if you Recalculate (F)… where do you go next???
Notice: What does “END” mean????
Notice: There is no RETURN control
Another Example:

Notice the procedure “name” has changed… “Determine Solution” is very different than “Determine a Solution”. The top diagram “implies” some kind of looping (Recalculate n) but there is no loop construct.

Note: There is no NS-diagram for “Determine if n is acceptable”. However, the code buried in the “Determine a Solution” is probably what is thought of. How can this be done where placed (item2)?
Another Example:

Notice: No “END”
Notice “structure failure” … “Obtain Problem Data” (this data could be bad and how does the program handle it?) Now “Determine Solution”. What is a valid vs invalid “solution”? “Report Solution” if valid… and “Determine Another Solution” if invalid? Doesn’t the “another solution” get reported? Is there a “LOOP” here… This appears to be a very mal-formed loop. A person proceeding with this approach WILL FAIL TO SOLVE THE PROBLEM.
VAG Vague Procedure names…

Procedures are “invoked actions”, hence, ALL BOXES containing a “word or phrase” should be VERB-like not NOUN-like. In addition, these “invoked actions” should be carefully selected to remind you EXACTLY what action is being performed.

<table>
<thead>
<tr>
<th>Noun Like</th>
<th>Verb Like</th>
<th>Better Verb Like</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculations</td>
<td>Perform Calculations</td>
<td>CalcAreaAndTolerance</td>
</tr>
</tbody>
</table>
HE: Hidden Errors

Some errors you have to “live and learn” to become aware of. For example, consider the following:

\[
\begin{align*}
n_1 &= 10 \quad \text{(Starting 2 cases of)} \\
 n_2 &= 20 \quad \text{# of rectangles) } \\
 n_1 &= 2 \times n_1 \\
 n_2 &= 2 \times n_2 \\
 \text{FIND AREA } (n_1) \\
 \text{FIND AREA } (n_2) \\
 \text{Accuracy} &= \frac{\text{Area}(n_2) - \text{Area}(n_1)}{\text{Area}(n_2)} \\
 \text{UNTIL Accuracy } \leq \text{ Rel Error} \\
 \end{align*}
\]

Notice: we don’t have anyway (yet) to pass information to a procedure (subroutine). Hence, the Find Area (n1) and Find Area (n2) cannot be implemented. Notice: The calculated accuracy can be either positive or negative. ANY negative value will cause the program to terminate (regardless of the actual accuracy).
End Loop
Return to top of loop

Where is the Loop Construct
What Loop??
This loop is an “UNTIL” loop so the condition should be checked at the bottom not the top.
No values are returned… Only a function name returns a value!
### Unnecessary Use of Subscripts (Arrays)

- \[ \text{width} = \frac{b-a}{n-1} \]
- \[ h_j = a + (j-1) \cdot \text{width} + \frac{1}{2} \cdot \text{width} \]
- \[ A_j = \text{width} \times h_j \]
- \[ A_{act} = A_{act} + A_j \]
- \[ j = j + 1 \]

**return value of** \( A_{act} \)

See \( H_j, A_j \), etc.
(Check validity)

X_{initial} < X_{final}

- TRUE
- FALSE

Interval = X_{final} - X_{initial}

- TRUE
- FALSE

Limit error is positive

- TRUE
- FALSE

Print error message & exit

Limit relative error < 10^{-20}

- TRUE
- FALSE

Print error message & quit

X_{initial} > 0

- TRUE
- FALSE

Print error message & exit

Return
Notice the SIGNIFICANT ERROR(S) in the conceptualization. The main program (above) seems to make sense… if the data is valid, determine the solution otherwise report the error. However the previous page shows a very different handling of this. We see that the program reports the errors “while checking if valid” and either QUILTS or EXITS at that point… this is INCONSISTENT. What happens when you “exit” or “quit”? Does the program STOP? Do you “exit” the “data validation procedure”?