

SPECIAL RULES

1. In the rules which define this programming language, the term "data item" is used to denote a variable, a constant, or a numeric literal.
2. Subfields of variables may be referenced by use of parentheses following the variable. Two data items within parentheses define beginning and ending character positions of the subfield. Examples: X(1-5) and Y(N-M).
3. Numeric literal may be used in instructions in place of a variable or a constant.
4. Alphanumeric variables receive data left justified with trailing spaces supplied if necessary. Numeric variables receive data right justified with leading zeros supplied if necessary. Alphanumeric data will be truncated on the right and numeric data will be truncated on the left; if the receiving variable is too small to contain the data. The content of variables prior to program execution may not be assumed.
5. A condition may take any of the following forms:

| | |
|-----------------------------|-----------------------------|
| data item GREATER data item | data item EQUAL data item |
| data item LESS data item | data item UNEQUAL data item |

Numeric data items are compared algebraically, alphanumeric data items are compared according to the following sequence: the space character is lowest, then the alphabet then the numbers in ascending sequence.
6. A label may appear to the left of any program instruction.

3 OPERAND

| INSTRUCTION | DESCRIPTION OF INSTRUCTION |
|--|---|
| t ALPHA(n) | Defines a variable which is n alphanumeric characters long and is referenced by the name t. |
| t NUMBER(n) | Defines a variable which is n numeric characters long and is reference by the name t. |
| t CONSTANT v | Defines a constant which is referenced by the name t. The value of the constant is v, where v may be a number or v may be a string of alphanumeric characters enclosed by quotation marks. (Examples: NUM CONSTANT 123 and NY CONSTANT 'NEW YORK') |
| x IS y operator z | Operators are PLUS, MINUS, TIMES and DIVIDED BY (remainder lost). The result of the operation utilizing data items y and z is placed in x. |
| GO s | Program control is transferred to the statement labeled s. |
| LOOP WHILE condition: statement(s) ENDLOOP | If the condition is true, execute all statements until ENDLOOP, then return to LOOP and test the condition again, if the condition is false, control is transferred to the statement following ENDLOOP. Control may be transferred to any statement following ENDLOOP or before LOOP WHILE by a GO instruction. |
| IF condition: statement(s) ENDIF | If the condition is true, execute all statements until ENDIF; if the condition is false, control is transferred to the statement following ENDIF. Control may be transferred to any statement following ENDIF or before IF by a GO instruction. |
| READ r from f ENDFILE GO s | One record is read from file f and place into r. If end of file is reached then control is transferred to the statement labeled s. |
| WRITE r to f | The contents of r are written to file f. |
| STOP | The program goes to normal termination. |
| ABORT | The program goes to abnormal termination. |

GENERAL INSTRUCTIONS FOR 2 OPERAND PSEUDO LANGUAGE

| <u>Instruction</u> | <u>Meaning</u> |
|---|---|
| X LOAD Y | Load the contents of Location Y into Location X |
| IF (conditional) ...statements... END IF | Execute the statements after and before the END IF only if the conditional is true |
| LOOP WHILE (conditional) ...statements... END LOOP | Execute the statements after the conditional and before the ENDLOOP until the condition is no longer true |
| IS or EQUAL | Value to right of "equal" is moved to location to the left |
| ADD A TO B | Value of A added to Value of B, stored in location B. |
| SUBTRACT A FROM B | Value of A subtracted from Value of B, stored in location B. |
| MULTIPLY A BY B | Value of A multiplied by Value of B, stored in B. |
| DIVIDE A BY B | Value of A divided by Value of B, remainder lost. |
| GO TO | Transfer control to statement indicated. |

INSTRUCTIONS FOR THE PSEUDO LANGUAGE (ONE OPERAND)

An instruction may consist of 3 parts: a label, an operand, and an address field, e.g. INP IN CARD. INP is the label; IN is the operation code; CARD is the address field. All work is done in the accumulator which automatically will accommodate any size field. When a field is stored, in a data field, it is right justified. If there is not enough room it truncates.

able to Accumulator
 $V \rightarrow A$
 $A \rightarrow V$

Label
xxx

| | | | |
|----------------|---------------|---------------------------|--|
| | LOAD | x | Load all characters at address x |
| | UNLOAD | x(1-5) | Unload into characters 1-5 |
| | COMPARE | x | Compares the accumulator against address x. If the accumulator is higher, it sets the HI flag. If it is lower, is sets the LOW flag. If equal, it unsets both. |
| | TEST | x | If true, reads next instruction else skip next instruction. |
| <i>operand</i> | Equate | <i>address field</i> x | Sets up a label equal to a constant |
| | HOP | x | Jump to x |
| | INCREASE | x | Increase accumulator by contents of address x |
| | DECREASE | x | Decrease accumulator by contents of address x |
| | MULTIPLY | x | Multiply accumulator by contents of address x |
| | DIVIDE | x | Divide accumulator by contents of address x |
| | IN | x | Read from hardware name indicated |
| | OUT | x | Output on hardware name indicated |
| | SPECIAL NAMES | x | TAPE, CARD, CONSOLE, ENDT, ENDC, HI, LOW |

ENDT AND ENDC are flags that are set at end of tape and end of cards respectively.

Hi and Low are set when a compare instruction receives a high or low condition.

| | | | |
|-----|------|---|--|
| | STOP | | END RUN |
| XXX | NAME | x | Defines storage area of size "x" example: RECIN NAME 80 |

PSEUDO CODE PROBLEM # 1

LC

HOW TO REVIEW

Follow your paper checking off instructions

PROBLEMS THAT LEAD TO ERRORS:

- Carelessness including losing your place, flipping pg over
- Misreading syntax instruction; misreading problem instruction
- Misreading question
- Misreading answer
- Jumping to wrong conclusion
- Truncation(alpha,numeric); Rounding(1.5 may be 1 so $1.5 + 2$); remainders($3/2$ may be 1)
- Accumulator size vs variable size
- One more reiteration in error
- Equals versus Greater than
- Tired
- Messiness
- Run out of time
- Tediousness
- Murphy's law

PR

TYPES OF PSEUDO

- One Operand vs Two Operand vs Three Operand (COBOL - like)
- Read and Solve vs Missing Statements vs Errors →
For later two, select plausible choice from answers and pl and work thru
- A few reiterations versus too many reiterations - *↳ go through first + second look for pattern*
Establish pattern; verify; do last reiteration

try to determine when the loop will dump out + attempt to solve the problem

PSEUDO CODE

- What is it?
- Made up of - instructions, operands, constants, variables
- Follows syntax instructions
- Why is it used?
- Benefits?

WHEN TO DO PSEUDO CODE PROBLEMS ON TEST?

HOW TO ATTACK PSEUDO CODE!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

- Unstaple booklet
- Use as much paper as you need; one side at a time; number pages
- BE VERY NEAT
- - Must Read Syntax Instructions that tell you pseudo code rules!!!
- Must Understand Instructions!!!
- Set up scrap paper page
 - Constant - Picture Size- across top
 - Variable - Picture Size- column for each
 - What is asked for - column for each
(read question w/o answer)
- Read Problem - # lines on text booklet
- Use separate line on your scrap paper page to correspond to each instruction (#) in problem.
- Do not cross-off values (messy, leads to errors, impossible to p/u if lost, impossible to alter, impossible to review).
- Draw solid line to indicate end of loop.
- Draw dotted line to indicate where you might jump out of loop to EOJ.
- Circle loops
- Bracket IFS
- Try to establish pattern (always verify and do last reiteration)