snobol
'snobol4' is the name of the interpreter on the ucs systems. Typically the interpreter takes the first set of input as your actual program then it takes the programs input then it gives the output result.

Statements in snobol
- assignment
- pattern matching
- replacement
- end

Assignment statement
var = *val* (the space is intended)
v = 'dog'
v = 5
v = “10”
x = 5 * -z + '10.6'

String concatenation:
blah = 'spit' 'bol'
nullassignmentvar =

Special variables:
OUTPUT = *(whatever is assigned to OUTPUT will be written as output)*
myLine = INPUT

Pattern matching statement
subject pattern
trade = 'program'
trade 'gram'

or

part = 'gram'
trade part

Replacement statement
subject pattern = *object* replaces pattern in subject with object, first occurance
word = 'gird'
word 'i' = 'ou'

hand = “AC4DAHKDKS”
rank = 4
suit = 'D'
hand rank suit =
(replaces first occurrence of '4D' with empty string)

Special characters (these are placed in the first column, all regular program input should start in the second column):
 * denotes the beginning of a comment
 + denotes the concatenation of a card
 &anchor = 0 denotes that string replacement should replace all occurrences of the match
 any other character is a **label**
 &trim = 1 trims trailing blanks from input

Labels affect the control of the program (goto sections).

- A label begins as a word that starts in the first column
- control is sent to a label when a colon is given near the end of the line (see examples below)
- END is a special label that effects the end of program

label1 statement : s(label2) f(label3)
this has the effect of giving a label1 to the statement line, and upon the success or failure of the statement the program either goes to label2 or label3

label4 statement : (label5)
this is a hard goto that occurs with or without success

Builtin functions for pattern matching (http://drofmij.awardspace.com/snobol/)

- The "LEN(number)" function will match the first n characters in a string. Using the "." operator, a substring can be extracted from a string and assigned to another variable.
- The "BREAK()" function matches all the characters up to, but not including, the character(s) specified in the function call. More than one character can be specified.
- The "SPAN(string)" function matches an uninterupted sequence of one or more of the characters in "string" irregardless of order.

someString LEN(4) . aVariable

myVar = 'jack be nimble'
myVar BREAK('n')

In the following example, "SPAN('kcaj b')" matches the first 6 characters of myVar "jack b" as they are contained within the span string.
myVar = 'jack be nimble'
myVar SPAN('bkcaj ')

Arrays
COUNT = **ARRAY**('3:9',0)
Define an array with valid subscripts that must be in the range 3 through 9; all others will cause the array reference to fail. Array elements are initialized to zero instead of the normal default, which is the null string.

The following statement is a way of increments array element values.
COUNT<SIZE(WORD)> = COUNT<SIZE(WORD)> + 1

Padding output
OUTPUT = **LPAD**(I,5) LPAD(COUNT<I>,20)
Pads the output with spaces

Patterns

- Alternation
- Concatenation

Alternation
P1 | P2

Concatenation
P1 P2

Example patterns
KEYWORD = 'Computer' | “Program”

KEYWORD = KEYWORD | 'Algorithm'
Matches 'computer' or 'program' or 'algorithm'

BASE = 'binary' | 'decimal' | 'hex'
SCALE = 'fixed' | 'float'
ATTRIBUTE = SCALE BASE

ATTRIBUTE = 'fixed' | 'float' 'decimal'
Matches 'fixed' or 'floatdecimal'

ATTRIBUTE = ('fixed' | 'float') 'decimal'
Matches 'fixeddecimal' or 'floatdecimal'

WORDPAT = BREAK(&LCASE &UCASE) SPAN(&LCASE &UCASE "'-") . WORD
searches for a word, which consists of uppercase and lowercase letters, apostrophes and hyphens.

Pattern value assignments
- conditional
- immediate

Conditional value assignment
'.'

BASE = ('hex' | 'decimal') . B1
If this pattern is involved in a successful pattern matching statement then the portion of the subject string that was matched will be saved in B1.

A.and.B = (A B) . OUTPUT

Conditional value assignment example
BR = (('B' | 'R') . FIRST ('E' | 'EA') . SECOND ('D' | 'DS') . THIRD) . BRVAL
'BEATS' BR
OUTPUT = FIRST
Null string is the output.

To see partial matches:

Immediate value assignment
'$'
Immediate value assignment example
BR = (('B' | 'R') $ FIRST ('E' | 'EA') $ SECOND ('D' | 'DS') $ THIRD) . BRVAL
'BEATS' BR
OUTPUT = FIRST SECOND
FIRST = 'B'
SECOND = 'E'
SECOND = 'EA'
Output is 'BEA'

How matching works… sort of
'subject string'
cursor is initialized to be right before the first 's' of 'subject string'.

((a | b) (c | d)) | (e | f))
Wants to match (a or b) and (c or d) or (e or f). Very similar to an automaton.

Pattern functions
SPAN(str) – matches longest consecutive string of characters from 'str' in the subject string.

SPAN('aeiou')
match any sequence of consecutive vowels.

BREAK(str) – matches longest sequence of characters in the subject that does not contain any characters in 'str'.

BREAK(' ')
Matches the first word.

ANY(str) – matches any single character from the argument that appears in subject str.

NOTANY(str) – matches a single character from the argument that does not appear in the subject str.

TAB(n) – matches the *n* next characters from the current cursor position.

RTAB(n) – matches everything except the last *n* characters.

POS(n) – only succeeds if the cursor is in position *n*.

RPOS(n) - only succeeds if the cursor is right of position *n*.

LEN(n) – will match as many characters as *n* in the subject string

SIZE(str) – find the size of the string

(User defined) Functions
```
 DEFINE('DELETE(STR, CHAR)', 'D1')
*       Program code ....
D1      STR CHAR =              :S(D1)
        DELETE = STR            :(RETURN)
```
Function deletes all occurrences of 'char' from 'str'. 'D1' is a label where the function can be found.
```
                                :(FRETURN)
```
Returns a failure signal.
```
                                :(NRETURN)
```
Will return the variable name instead of the value.

Pattern function examples
'dog' 'd' any('aeiou') 'g'
Like a match once function.

NOTANY('aeiou')
Match anything but a vowel.

TAB(0) pattern RTAB(0)
This will always match the whole line even if pattern fails.

POS(0) pattern RPOS(0)
Will make sure that pattern IS the subject string.

(User defined) Function examples
DEFINE('MAXNO(P,N)')
Snobol assumes that the function name is the label.

```
MAXNO       N = GT(N,0) N – 1              :F(RETURN)
            MAXNO = null | P MAXNO         :(MAXNO)
```

Will match n times of pattern 'P' or less than n.

Example

```
*       The following reverses a string
*
*       has a locally defined variable CH, ONECH
        DEFINE('REVERSE(STR)CH,ONECH',R1)
        .
        .
        .
R1      ONECH = LEN(1) . CH
*       matches any character string of length 1
R2      STR ONECH =                                       :F(RETURN)
        REVERSE = CH REVERSE                              :(R2)
```

Example

```
*       all N length combinations of STR
        DEFINE('COMB(STR, N, HEAD)CH')
        .
        .
        .
*       output the value of HEAD is N == 0
COMB    OUTPUT = EQ(N,0) HEAD                             :S(RETURN)
C2      STR LE(N,SIZE(STR)) LEN(1) . CH =                 :F(RETURN)
        COMB(STR, N – 1, HEAD CH)                         :(C2)
```

Structured data types
ARRAY(...) TABLE(...) DATA(...)

Arrays

```
*       creates a 10 element array indexed from 1 to 10 of null strings
        VECTOR = ARRAY(10)

*       array with 11 elements indexed from -5 to 5
        LINE = ARRAY('-5:5')

*       3 by 3 array with all elements initialized to X
        BOARD = ARRAY('3,3', 'X')

*       all elements in A2 are A1
        A1 = ARRAY(5)
        A2 = ARRAY(5,A1)
```

To access arrays

```
        A1<2>
        BOARD<1,3>
```

*   this is how you would index an A1 element from A2

```
        ITEM(A2<1>,1)
```

Tables

By default the size is 10, tables are dynamic and have an increment size of 10

```
        T = TABLE()
        T<1> = 'Mark'
        T<'1'> = 'Radle'
        T<T<'1'>>     is actually T<'Radle'>
```

Data

```
        DATA('COMPLEX(R,I)')
        .
        .
        .
        C = COMPLEX(1.5, 2.0)
        R(C) = 2
        I(C) = 1.5
        .
        .
        A = R(C)
        B = I(C)
```

*   similar to a linked list

```
        DATA('TEXT(LINE, N, NEXT)')
        .
        .
        .
        I = 1
        HEAD = TEXT(INPUT, I)                                  :F(EMPTY)
        CURRENT = HEAD
LOOP
        I = I + 1
        NEXT(CURRENT) = TEXT(INPUT,I)                          :F(END)
        CURRENT = NEXT(CURRENT)                                :(LOOP)
EMPTY
        OUTPUT = 'NO INPUT'
END
```