# **SPITBOL/SNOBOL4 HOMEWORK**

**A bit of background**

* Snobol has a bunch of dialects, relatives, and implementations[[1]](#footnote-1) (this was Pretty Normal for when it was created – languages tended to be VERY tied to specific hardware and OS combos for a long time)
* We’re sticking to “plain vanilla” Snobol4[[2]](#footnote-2) (as much as possible)

(for the tidy-inclined, “.sno” is the usual extension for Snobol)

**And here we go**

Once you’re logged in, we’re going to check some things out….

And, since time is tight, we’re going to make life a bit easier by starting with some existing code (at first).

***First Exercise***

* Take bottles.sno and modify it so that you’re taking 2 bottles at a time, not one.
* Take a screenshot of it running with the change.

Code:

\* Bottles of Beer on the Wall

\* From Peter-Arno Coppen,

\*

&fullscan = 1

Beer = " of beer"

Wall = " on the wall"

Take = "Take two down and pass them around"

NL = CHAR(10)

NL = ". "

Bottle = \*?((B1 = (EQ(N1,2) "Just two bottles",

+ N1 EQ(REMDR(N1,2),0) " bottles"))

+ (B2 = (EQ(N2,2) "Just two bottles",

+ (EQ(N2,0) "No more",(N1 - 2))

+ " bottles")))

Sing = \*?(OUTPUT = B1 beer Wall NL B1 Beer NL

+ Take NL B2 Beer Wall NL)

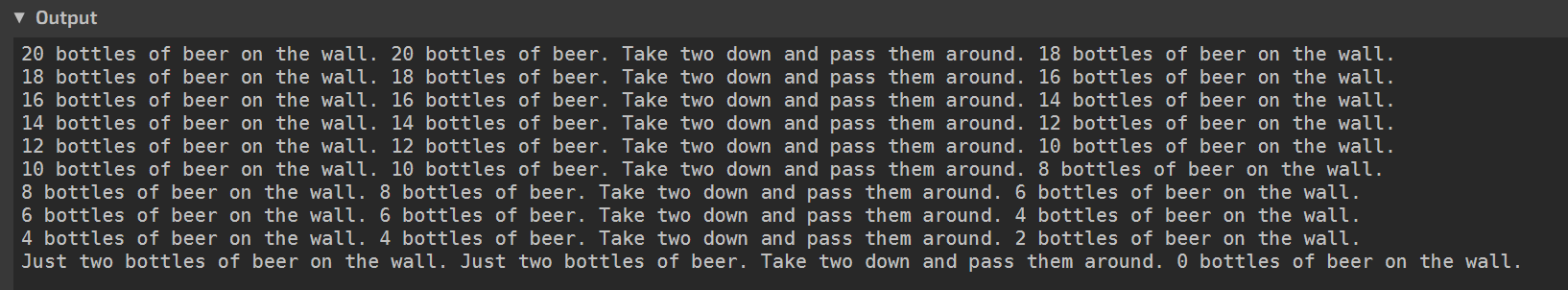
Supply = DUPL("B",20)

Count = ("B" \*count | "")

Remove = @N2 "B" @N1

Supply ? FENCE Count Remove Bottle Sing FAIL

END



***Second Exercise***

* Take stringcomp.sno and add comments to it (explaining what the operators are checking).
* Add at least 3 more “things” to it (each thing should use at least 2 of the string tools like differ or LNE, etc.).
* One of your three things should use some tool from the manuals/docs (see footnotes) that is NOT in the examples – you can also try <http://berstis.com/s4ref/pats3b.htm> for ideas.

Code:

s1 = 'mnopqrs'

s2 = 'mnopqrs'

s3 = 'mnopqr'

s4 = 'nop'

s5 = 'nOp'

OUTPUT = 'Case sensitive comparisons:'

\* LEQ tests for lexical/letter equality of two strings

OUTPUT = LEQ(s1, s2) s1 ' and ' s2 ' are equal (LEQ).'

\* IDENT tests for equality between strings, objects & numbers

OUTPUT = IDENT(s1, s2) s1 ' and ' s2 ' are equal (IDENT).'

OUTPUT =

\* LNE checks if the arguments are different strings

OUTPUT = LNE(s1, s3) s1 ' and ' s3 ' are not equal (LNE).'

\* LEQ tests for equality of the strings, and ~ negates the output

OUTPUT = ~LEQ(s1, s3) s1 ' and ' s3 ' are not equal (~LEQ).'

\* DIFFER returns true if the strings (can also check objects or numbers) are unequal

OUTPUT = DIFFER(s1, s3) s1 ' and ' s3 ' are not equal (DIFFER).'

\* LGE checks if the first string is greater than or equal to the second string

OUTPUT =

OUTPUT = LGE(s1, s3) s1 ' is greater than or equal to ' s3 ' (LGE).'

\* LLE checks if the first string is less than or equal to the second

OUTPUT = LLE(s3, s1) s3 ' is less than or equal to ' s1 ' (LLE).'

OUTPUT =

\* LGT checks if the first string is greater than the second

OUTPUT = LGT(s4, s1) s4 ' is greater than ' s1 ' (LGT).'

\* LLT checks if the first string is less than the second

OUTPUT = LLT(s1, s4) s1 ' is less than ' s4 ' (LLT).'

OUTPUT =

OUTPUT = "Case insensitive comparison:"

\* LEQ checks for equality between the string s4 and the string produced by REPLACE

\* REPLACE takes the first argument as a string & replaces any characters that are in the character list of the second argument with the corresponding character in the second list(3rd argument)

OUTPUT = LEQ(s4, REPLACE(s5, &UCASE, &LCASE)) s4 ' and ' s5 ' are equal.'

OUTPUT =

\* EQ checks for numerical equality (the '1234' is being interpreted as a number b/c that's the type needed here)

OUTPUT = 'String and numeric conversions and comparisons:'

OUTPUT = EQ('1234', 1234) '"1234" and 1234 are equal (coerce to integer).'

\* LEQ checks for lexical (letter) equality as above, but 1234 is being interpreted as a string in this case

OUTPUT = LEQ('1234', 1234) '"1234" and 1234 are equal (coerce to string).'

OUTPUT =

\* GT checks if the first number is greater than the second

OUTPUT = GT('1234', 1233) '"1234" is greater than 1233 (numeric comparison).'

\* LT checks if the first number is less than the second

OUTPUT = LT('1233', 1234) '"1233" is less than 1234 (numeric comparison).'

\*DATE is kind of self-explanatory...

OUTPUT = DATE()

\* ATAN is the arc tangent of the argument

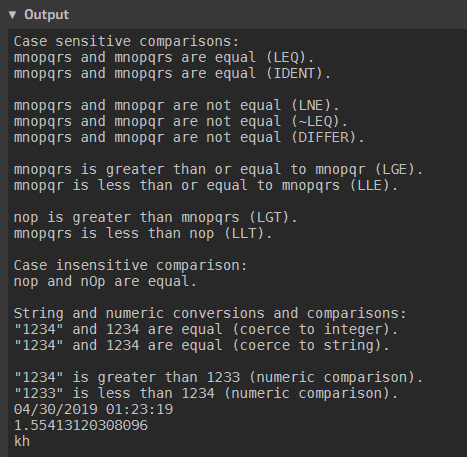
OUTPUT = ATAN(60)

\*CHAR gives the character with the ASCII code from the argument

OUTPUT = CHAR(107) CHAR(104)

END

Output:



***Turning it in***

I would like a blackboard upload of your code and the screenshots. To speed up grading, please put the screenshots in a word processing (OpenOffice/Word or PDF; Word is preferred!) document; you may also put your code in there if you want, or you can upload it as text file(s).

1. <http://www.snobol4.org/> [↑](#footnote-ref-1)
2. <http://www.snobol4.org/docs/burks/manual/contents.htm> is one reference. <http://www.snobol4.org/docs/books.html> is another. [↑](#footnote-ref-2)