Q PROGRAMMING LANGUAGE

Kdb+ comes with its built-in programming language that is known as **q**. It incorporates a superset of standard SQL which is extended for time-series analysis and offers many advantages over the standard version. Anyone familiar with SQL can learn **q** in a matter of days and be able to quickly write her own ad-hoc queries.

Starting the “q” Environment

To start using kdb+, you need to start the **q** session. There are three ways to start a **q** session −

- Simply type “c:/q/w32/q.exe” on your run terminal.
- Start the MS-DOS command terminal and type **q**.
- Copy the **q.exe** file onto “C:\Windows\System32” and on the run terminal, just type “q”.

Here we are assuming that you are working on a Windows platform.

Data Types

The following table provides a list of supported data types −

Name	Example	Char	Type	Size
boolean	1b	b	1	1
byte	0xff	x	4	1
short	23h	h	5	2
int	23i	i	6	4
long	23j	j	7	8
real	2.3e	e	8	4
float	2.3f	f	9	8
char	“a”	c	10	1
varchar	`ab	s	11	*
month	2003.03m	m	13	4
date	2015.03.17T18:01:40.134	z	15	8
minute	08:31	u	17	4
second	08:31:53	v	18	4
time	18:03:18.521	t	19	4
enum	`u$`b, where u:`a`b	*	20	4

Atom and List Formation

Atoms are single entities, e.g., a single number, a character or a symbol. In the above table *ofdifferentdatatypes*, all supported data types are atoms. A list is a sequence of atoms or other types including lists.

Passing an atom of any type to the monadic *i. e. singleargumentfunction* type function will return a

negative value, i.e., **–n**, whereas passing a simple list of those atoms to the type function will return a positive value **n**.

Example 1 – Atom and List Formation

```
/ Note that the comments begin with a slash “ / ” and cause the parser
/ to ignore everything up to the end of the line.

x: `mohan              / `mohan is a symbol, assigned to a variable x
type x                 / let’s check the type of x
-11h                   / -ve sign, because it’s single element.

y: (`abc;`bca;`cab)    / list of three symbols, y is the variable name.

type y
11h                    / +ve sign, as it contain list of atoms (symbol).

y1: (`abc`bca`cab)     / another way of writing y, please note NO semicolon

y2: (`$”symbols may have interior blanks”)   / string to symbol conversion
y[0]                   / return `abc
y 0                    / same as y[0], also returns `abc
y 0 2                  / returns `abc`cab, same as does y[0 2]

z: (`abc; 10 20 30; (`a`b); 9.9 8.8 7.7)      / List of different types,
z 2 0                  / returns (`a`b; `abc),
z[2;0]                 / return `a. first element of z[2]

x: “Hello World!”      / list of character, a string
x 4 0                  / returns “oH” i.e. 4th and 0th(first)
element
```