

Tracing Pseudo-code

Construct a trace table for the execution of `di2b2(41)` and explain what would happen if the binary test at [2] were to be replaced with $n \geq 0$.

```
[0] bits = di2b2 (n)
[1] bits ← ∅
[2] while n > 0
    [2.1] q ← floor(n/2)
    [2.2] r ← n - 2×q
    [2.3] bits ← r, bits
    [2.4] n ← q
```

Line	n	bits	q	r	n > 0
[0]	41	-	-	-	-
[1]	41	∅	-	-	-
[2]	41	∅	-	-	1
[2.1]	41	∅	20	-	1
[2.2]	41	∅	20	1	1
[2.3]	41	1	20	1	1
[2.4]	20	1	20	1	1
[2]	20	1	20	1	1
[2.1]	20	1	10	1	1
[2.2]	20	1	10	0	1
[2.3]	20	0,1	10	0	1
[2.4]	10	0,1	10	0	1
[2]	10	0,1	10	0	1
etc.					

The trace table has been cut short since the loop repeats for n going through 5, 2, 1 and 0 but note that if the test for the while loop is $n \geq 0$ then the loop will never terminate since q and consequently n will be computed to be 0 in every subsequent pass.

Writing Pseudo Code for the Data Analysis functions:

```
[0] s = sum(L)
    [0.1]NB. L is a list of numbers
    [0.2]NB. s is the sum of the numbers in L
[1] s ← 0
[2] while length(L) > 0
    [2.1] s ← s + first(L)
    [2.2] L ← behead(L)
```

```
[0] m = mean(L)
    [0.1]NB. L is a list of numbers
    [0.2]NB. m the mean of the numbers in L
[1] m ← sum(L)/length(L)
```

```
[0] y = dfm(L)
    [0.1]NB. L is a list of numbers
    [0.2]NB. y: deviations of L from the mean
[1] y ← L - mean(L)
```

```
[0] y = msd(L)
    [0.1]NB. L is a list of numbers
    [0.2]NB. y is average deviation from mean
[1] y ← mean(dfm(L))
```

```
[0] s = sd(L)
    [0.1]NB. L is a list of numbers
    [0.2]NB. s is the standard deviation
[1] y ← sqrt(msd(L))
```

Notice that functions usually call other functions, *i.e.* they are *composite* functions (or functions of functions).

Integers in a computer

With 12-bits for storing integers find:

- (a) the largest and smallest integers that can be represented?

The largest positive number is 011111111111 or 2047. 100000000000 is the most negative number and represents -2048 .

- (b) The computer representations for

$$(i) \quad -1 \quad \equiv \quad 111111111111$$

$$(ii) \quad 27 \quad \equiv \quad 00000011011$$

$$1's \text{ complement is } 11111100100$$

$$-27 \quad \equiv \quad 11111100101$$

$$(iii) \quad 200 \quad \equiv \quad 000011001000$$

$$1's \text{ complement is } 111100110111$$

$$-200 \quad \equiv \quad 111100111000$$

$$37 \quad \equiv \quad 000000100101$$

- (c) To recover the negative integers we can either reverse the process or take the two's complement. Both lead to the same result.

$$(i) \quad 111101011001 - 1 = 111101011000$$

$$1's \text{ complement is } 000010100111$$

$$128 + 32 + 7 = 167$$

$$111101011001 \text{ is } -167$$

$$(ii) \quad 110110100011 - 1 = 110110100010$$

$$1's \text{ complement is } 001001011101$$

$$512 + 64 + 16 + 13 = 605$$

$$110110100011 \text{ is } -605$$

$$(ii) \quad 000011111110 = 000011111111 - 1$$

$$= (000100000000 - 1) - 1$$

$$= 254$$
