Chapter 3

End of Chapter Exercises

1. *Nick's computer has three times the memory of Lynne's and Alf's computers put together. Shadi's PC has twice as much memory as Chris's. Nick's computer has one-and-a-half times the memory of Shadi's. Between them, Alf and Shadi's computers have as much memory as Lynne's plus twice the memory of Chris's. Shadi, Chris, Nick, Alf, and Lynne's PCs have 2,800 megabytes of memory between them. How much memory does each computer have?*

A difficult question until you realize algebra is the way to go. n = Nick, l = Lynne, a = Alf, s=Shadi, c=Chris

(1) $n = 3(l + 4)$

Shadi's PC has twice as much memory as Chris's:

(2) $s = 2c$

Nick's computer has one-and-a-half times the memory of Shadi's:

(3) $n = \frac{3}{2}s$

Between them, Alf and Shadi's computers have as much memory as Lynne's plus twice the memory of Chris's:

(4) $a + s = l + 2c$

Shadi, Chris, Nick, Alf, and Lynne's PCs have 2,800 megabytes of memory between them:

(5) $s + c + n + a + l = 2800$

We know from (1) that $l + a = \frac{n}{3}$ so we can rearrange (5) to give:

(6) $c + s + \frac{4}{3}n = 2800$

We also know from (2) that $c = \frac{s}{2}$ so we can substitute again to change (6) into:

(7) $\frac{3}{2}s + \frac{4}{3}n = 2800$

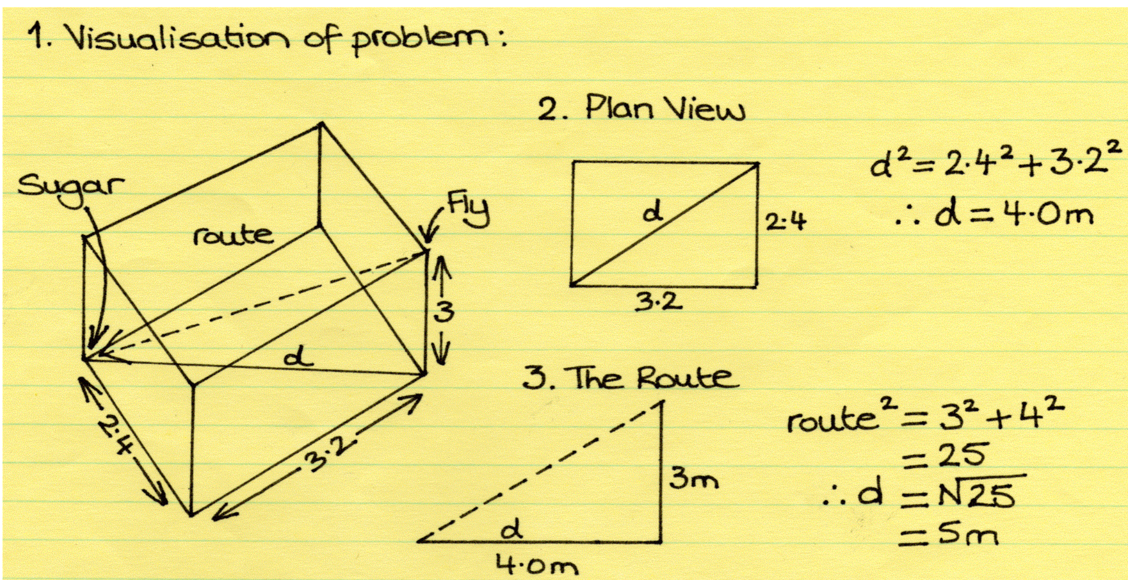
We also know from (3) that $n = \frac{3}{2}s$ so $\frac{4}{3}n = \frac{4}{3} \times \frac{3}{2}s = \frac{12}{6}s = \frac{4}{2}s$ which means we can substitute in (7) to give

(8) $\frac{7}{2}s = 2800$

Try doing this exercise with words alone!

2. *The fly-sugar problem. This exercise is a variation of the ant-sugar problem. A room measures 3.2m long, 2.4m wide and 3m high. A fly sits in one of the upper corners looking at a bowl of sugar in the lower corner diagonally opposite. Assuming the fly can, indeed, fly, calculate the shortest route to the sugar. This is easiest if you use a combination of algebra and diagrams as it is a problem in geometry.*

Here is my solution:



3. *Chessboards have alternate black and white squares, and the square in the top left corner is always white. Imagine you are making a chessboard but for some peculiar reason you have decided to paint each square in a random order. Without continually counting "white, black, white, black..." from the top left corner each time or looking at pictures of a chessboard each time, consider how else you could determine whether any given square should be black or white.*

Look at the row and column numbers of the white squares. R1C1 is white, R2C2 is white, R1C3 is white, R2C3 is black. What's the pattern?

A square whose row and column numbers are either both odd or both even is white.

4. *A farmer keeps sheep and chickens. In the farmyard there are 68 animals with a total of 270 legs. Assuming every chicken has exactly 2 legs and each sheep exactly 4 legs, how many chickens and sheep are there in the farmyard? The farmer changes the number of sheep and chickens such that there are now 75 animals but still 270 legs. How many of each animal does the farmer own now?*

I approached this the way a naval gunner might lock on to a target: fire a shot which we expect to fall short and measure where it lands. Now extend the range to one which we think will be long. Measure where that shot falls. Subtract the two distances and calculate the correct range. So, here's a way of homing in on the answer.

What do we know? We know there are 270 legs and two types of animal: chicken (2 legs) and sheep (4 legs) and that the number of chicken and sheep together is 68. What is the unknown? How many of each animal on the farm.

Let's simplify the problem and remove the sheep. If there are only chickens, then there must be 270 ÷ 2 of them (2 legs each) = 135 chickens. Clearly this is not close to the right answer as the farm only has 68 animals. So, we must have fewer chickens and more sheep. Ok, lets say there are only sheep. That would give us 270 ÷ 4 = 67.5 sheep. Now, while this is not the answer it is pretty close. We can't have half a sheep, but we can have 67 of them. Half a sheep is equal (in legs) to one chicken. So, 67 sheep × 4 legs = 268 and 1 chicken = 2 legs and 268 + 2 = 270 so there must be 67 sheep and only one lonely chicken. Hurrah!

Now we have to find the balance with 75 animals but the same number of legs. Clearly there must be fewer sheep and more chickens than before. We've added 7 to the animal count. For every sheep we remove we must add two chickens to keep the total number of legs the same. Also, removing 1 sheep and adding 2 chickens adds 1 animal overall to the farm. So, if we need 7 extra animals we can do this by removing 7 sheep and adding 14 chickens. Removing 7 sheep gives us 60 sheep = 240 legs. Adding 14 chickens gives us 15 in total = 30 legs. 240 + 30 = 270 so there must be 60 sheep and 15 not so lonely chickens.

I solved this by a combination of verbal reasoning (I talked the problem through with myself) and some basic arithmetic.

5. *When you pay cash for something, good cashiers give you your change using the fewest coins possible. Using the HTTLAP strategy, write an algorithm that works out the ideal change to give for any amount between 1 and 99 pence/cents. If you are working with euros or British pounds then the coins available to you are 1, 2, 5, 10, 20, and 50 (cents and pence). If working with U.S. dollars, then the available coins are 1, 5, 10, and 25 (I am ignoring the rare and unpopular half-dollar and one-dollar coins). Here are some examples:*

- *To give 67 pence in change requires 1×50p + 1×10p + 1×5p + 1×2p*
- *To give 43 euro cents in change requires 2×20c + 1×2c + 1×1c*
- *To give 63 U.S. cents in change requires 2×25¢ + 1×10¢ + 3×1¢*

What is the known? We know we have an amount which must be made up using the fewest coins possible. We know what denominations of coin are available. I am going to work with sterling because I'm in the UK. So, I have 50p coins, 20p coins, 10p coins, 5p coins, and 1p coins.

Lets start with a simple problem. How many coins are needed to give 50 pence change? Only 1 coin is needed, a 50p coin. Why? 50p is the largest denomination coin that is not larger than the amount of change needed. In fact, it is equal, so the task is easy. What about 70 pence change? What's the highest denomination that is not larger than the amount needed? It's 50p again. So, we need 1 50p piece. But we're not finished there's still 20p left over. What's thelargest denomination that will fit into 20 pence? Why, it's the 20p coin. So for 70 pence we need 1×50p + 1×20p. I see a pattern emerging. Find the largest denomination coin and find the left over. How about 40 pence? The largest coin we can use is the 20p. But we need more than 1 of them because 20 goes into 40 twice over. 2×20p = 40. Job done. Ok, let's extend this about. How about the following algorithm?

```
1.   Number of 50p coins needed = amount ÷ 50 ;
2.   Amount left over = remainder after dividing amount by 50 ;
3.   Number of 20p coins = left over ÷ 20 ;
4.   Amount left over = remainder after dividing left over by 20 ;
5.   Number of 10p coins = left over ÷ 10 ;
6.   Amount left over = remainder after dividing let over by 10 ;
7.   Number of 5p coins = amount ÷ 5 ;
8.   Amount left over = remainder after dividing left over by 50 ;
9.   Number of 2p coins = left over ÷ 2 ;
10.  Number of 1p coins = remainder after dividing left over by 2 ;
```

Let's try it out. How about 98p?

1. Number of 50p = 1 because 50 goes into 98 one time.
2. Left over = 48p
3. Number of 20p = 2 (48÷20 = 2)
4. Left over = 8p
5. Number of 10p = 0 (10 doesn't divide into 8)
6. Left over = 8p
7. Number of 5p = 1
8. Left over = 3p
9. Number of 2p = 1
10. Number of 1p =1

Answer = 1×50p + 2×20p + 0×10p + 1×5p + 1×2p + 1×1p = 6 coins.

6-14: These exercises also appeared in Chapter 2. All we need to do is rewrite our solutions using the pseudo-code notation. For example, here's the coffee making algorithm in pseudo-code:

```
1.  Measure water for one cup ;
2.  Pour water into coffee machine's reservoir ;
3.  Put filter paper into machine ;
4.  Measure coffee for one cup ;
5.  Put coffee into filter paper and close the door ;
6.  Plug machine into electricity outlet ;
7.  Switch on machine ;
8.  Wait for coffee to filter through ;
9.  Pour cup ;
10. Turn off machine ;
```

Notice how this algorithm looks very similar to the solution I came up with in Chapter 2 for converting decimal numbers to roman numerals?

Projects

StockSnackz Vending Machine

Take your existing solution from Chapter 2 and write it using pseudo-code. Does drawing a diagram of the vending machine and its principal components help?

```
1. Install the new machine ;
2. Turn on power ;
3. Load machine with snacks ;
4. Dispense snacks ;
5. Show dispensing report ;
```

Stocksfield Fire Service: Hazchem Signs

Take your existing solution from Chapter 2 and write it using pseudo-code.

```
1. Decode first character and give fire-fighting instructions
;
2. Decode second character and give precaution instructions ;
3. Decode third character and state whether public hazard
exists ;
```

Puzzle World: Roman Numerals and Chronograms

Take your existing solution from Chapter 2 and write it using pseudo-code.

Translating decimal to roman:

```
1.  Number of 'M's needed = number ÷ 1000 ;
2.  Leftover = remainder after dividing number by 1000 ;
3.  Number of 'D's needed = leftover ÷ 500 ;
4.  Leftover = remainder after dividing leftover by 500 ;
5.  Number of 'C's needed = leftover ÷ 100 ;
6.  Leftover = remainder after dividing leftover by 100 ;
7.  Number of 'L's needed = leftover ÷ 50 ;
8.  Leftover = remainder after dividing leftover by 50 ;
9.  Number of 'X's needed = leftover ÷ 10  ;
10. Leftover = remainder after dividing leftover by 10 ;
11. Number of 'V's needed = leftover ÷ 5 ;
12. Number of 'I's needed = remainder after dividing leftover
by 5 ;
```

Translating Roman to decimal:

This problem requires us to work through the roman number identifying sub-sequences of roman digits and converting them to decimal and then to add up the values of all the sub-sequences at the end. We do not have the vocabulary yet for dealing with repeated actions, so the outline algorithm will be quite short for the time being. If we take a roman number like MCMXCIX we might translate it like this:

```
1.  Work through the number identifying the sub-sequences ;
2.  Calculate the decimal value of each sub-sequence ;
3.  Sum all the values ;
```

So, for MCMXCIX we would identify the sub-sequences thus: M : CM : XC : IX
The decimal equivalents are 1000 : 900 : 90 9
The total value then is 1999.

We can see, therefore, that there is some sort of repeated action dealing with each of the sub-sequences. Identifying the sub-sequences is also an interesting problem in its own right.

Pangrams: Holoalphabetic Sentences

Take your existing solution from Chapter 2 and write it using pseudo-code.

This algorithm also implies some repeated actions. It also refers to decisions that must be made (see step 3). We will deal with these problems as our vocabulary is extended in later chapters.

```
1.  Write each letter of the alphabet on a piece of paper ;
2.  Starting at the first letter of the sentence, work
    through the sentence letter by letter. Strike through the
    corresponding letter on the piece of paper from step 1
    until the whole sentence has been processed ;
3.  Take the piece of paper from step 1. If there are any
    letters that haven't been crossed out, then the sentence
    is not a pangram ;
```

Online Bookstore: ISBNs

Take your existing solution from Chapter 2 and write it using pseudo-code.

```
1.  Determine the group code and write it out ;
2.  Write a hyphen ;
3.  Determine the publisher code and write it out ;
4.  Write a hyphen ;
5.  Determine the title code and write it out ;
6.  Write a hyphen ;
7.  Write out the check digit ;
```

There are some sub problems lurking in here. How do we determine the group code, the publisher code, and the title code? We will expand this aspect in later chapters.