Chapter 4

End of Chapter Exercises

1. *In what circumstances would you use the `IF` and `WHILE` constructs? How are they different?*

We use the `IF` when we want to choose whether or not to perform a single action (or sequence of actions). We use the `WHILE` when we want to repeatedly perform an action or sequence of actions. Both involve conditionally carrying out an action, but they differ in that the `WHILE` will repeatedly carry out its action block whereas the IFwill only carry out its action block once.

2. *Examine the following algorithm fragment:*

```
put on hat ;
IF (weather is sunny)
   put on sunglasses ;
ENDIF
put on shoes ;
```

What items of clothing will be put on a) when it is raining and b) when it is sunny?

When it is raining the hat and shoes are worn. When it is not raining (which could be sunny) then the hat, shoes, and sunglasses are worn. Sunglasses are worn conditionally dependent on the `IF` construct.

3. *Examine the following algorithm fragment:*

```
Go to shop ;
buy milk ;
IF (today is Saturday)
   buy weekly newspaper ;
   buy peanuts ;
ENDIF
buy bread ;
```

What items will be purchased a) on Thursday, b) on Saturday?

Answer: a) milk and bread, b) milk, newspaper, peanuts, bread You might have thought that peanuts were bought on Thursday too, but this is because the buy peanuts action is wrongly indented. The thing which really matters is that the buy peanuts action comes inside the `ENDIF` keyword. All the actions between the `IF` and `ENDIF` keywords are carried out when the condition is true.

4. *Consider the following algorithm fragment:*

```
1.  Eat breakfast ;
2.  Wash breakfast dishes ;
3.  WHILE (not finished breakfast) ;
       3.1 Read a story in morning paper ;
       3.2 Take next bite of breakfast ;
    ENDWHILE
```

How many stories in the morning paper will the person get to read?

No stories will be read as the `WHILE` loop is conditional upon breakfast not being finished but we can see from steps 1 and 2 that breakfast has been eaten before the `WHILE` loop action is reached.

5. *In the Stocksfield Diner customers can choose whether or not to have their hamburgers plain or with any combination of cheese, lettuce and tomatoes. Write three `IF` constructs that add as required by the customer.*

Adding cheese, lettuce and tomatoes:

```
Get cheese, lettuce, and tomatoes requirements ;
IF (cheese required)
   Add cheese ;
ENDIF
IF (lettuce required)
   Add lettuce ;
ENDIF
IF (tomatoes required)
   Add tomatoes ;
ENDIF
```

6. *Use the HTTLAP strategy to design an algorithm to represent filling a bath. The taps should be turned off altogether when the bath is full. When the bath is half full check the water temperature and if it is not warm enough adjust it by turning the cold tap down a quarter turn.*

```
1.  Turn on hot and cold taps ;
2.  Wait until half full ;
3.  IF (temperature too cool)
       3.1 Turn cold tap 0.25 turns counter-clockwise ;
    ENDIF
4.  Wait until bath full ;
5.  Turn off both taps ;
```

7. *The Department of Horticulture at the University of Stocksfield is testing a new fertilizer. They are interested in how long the fertilizer may be kept before its effectiveness is lost. They have done some calculations that indicate the fertilizer loses 6% of its potency every month. Using HTTLAP design an algorithm that finds how many months it takes before the fertilizer has lost more than 50% of its effective power after which is must be thrown away. The first four months of potency figures would look like this:*

Month 1, potency 100%
Month 2, potency 94%
Month 3, potency 88.36%
Month 4, potency 83.0584%

The algorithm should produce summary messages for each month as above stopping after the potency has gone below 50%. Note, you are removing 6% of the fertilizer's remaining potency each month, not its original potency. That is why after month 3 the potency is 88.36% and not 88%.

```
1.  Start potency at 100% ;
2.  WHILE (potency greater than or equal to 50%)
       2.1.  Display Month & potency message ;
       2.2.  Reduce potency by 6% ;
       2.3.  Move to next month ;
    ENDWHILE
```

8. *For the following scenarios write down a basic sequence of actions (no selections or iterations needed) and then try to partition the actions to see if the sequence fits the Initialization, Processing, and Finalization algorithm pattern:*
a) Getting up and ready in the morning
b) Cooking a meal
c) Writing an essay or school report

Getting up in the morning:

```
1.  Wake up ;
2.  Turn off alarm ;
3.  Get out of bed ;
4.  Take a shower etc  ;
5.  Put on underwear ;
6.  Put on socks ;
7.  Put on trousers ;
8.  Put on shirt ;
9.  Put on shoes ;
10. Make breakfast ;
11. Eat breakfast ;
12. Clear away and wash up breakfast things ;
```

Tasks 1-3 initialization; tasks 4-11 processing; task 12 finalization.

Cooking a meal

```
1.  Prepare ingredients ;
2.  Get pans etc ready ;
3.  Heat oven ;
4.  Mix & chop ingredients ;
5.  Put meal in oven ;
6.  Wait until cooked ;
7.  Serve meal ;
8.  Eat meal ;
9.  Wash dishes ;
```

Tasks 1-3 initialization; tasks 4-8 processing; task 9 finalization.

Writing an essay/school report

```
1.  Decide on essay title ;
2.  Go to library and get relevant books/articles ;
3.  Read the source material and make some notes ;
4.  Sketch outline of essay ;
5.  Draft the full essay ;
6.  Tidy up the format ;
7.  Proof read ;
8.  Print out ;
9.  Hand it in ;
```

Tasks 1-3 initialization; tasks 4-5 processing; tasks 6-9 finalization .

9. *Mow a lawn. Start with the simplest form of the problem and then introduce some of the real issues that you would face when mowing the lawn such as, height of cut, what happens when the grass box gets full?, the role of the weather, and so on.*

The simple problem:

```
1.  Get out lawnmower ;
2.  Add fuel as necessary ;
3.  Start mower ;
4.  Mow lawn ;
5.  Switch off mower ;
6.  Empty grass box ;
7.  Clean mower and put away ;
```

Dealing with additional problems

What other decisions do I make when mowing the lawn?

* I need to adjust the cut height depending on the length of the grass and the weather. If the grass is very long I will need to choose a high setting and go over the lawn once and then lower the setting and go over again: this avoids jamming the machine.
* If the weather has been very hot and dry (drought conditions) then I don't want to cut the grass as short as normal in order to prevent it getting scorched.
* Sometimes I can go all the way round without having to stop and empty the grass box, but if the grass has been growing well or I don't get it every week I usually end up having to empty the grass box several times.

How to factor all these issues into the algorithm?

```
1.  Get out lawnmower ;
2.  Set cutting height to 'short' ;
3.  Add fuel as necessary ;
4.  IF (grass length long)
       4.1. Set cutting height to 'high' ;
    ENDIF
5.  WHILE (not finished)
   5.1  Start mower ;
   5.2. WHILE (grass box not full and we haven't finished the
lawn)
          5.2.1. Mow grass ;
        ENDWHILE
   5.3. Switch off mower ;
   5.4. Empty grass box ;
ENDWHILE
6.  Clean mower and put away ;
```

If this is a situation where I need to go over the lawn twice then the simple solution is to just repeat the algorithm. Otherwise I could add another `WHILE` loop between statements 1 & 2 and test to see if another pass is required at the bottom of the loop:

```
1.  Get out lawnmower ;
2.  WHILE (not finished)
       2.1. Set cutting height to 'short' ;
       2.2. Add fuel as necessary ;
       2.3. IF (grass length long)
```

```
              2.3.1. Set cutting height to 'high' ;
           ENDIF
   2.4 WHILE (not finished)
         2.4.1.  Start mower ;
         2.4.2.  WHILE (grass box not full and we haven't
finished the lawn)
                    2.4.2.1.  Mow grass ;
                 ENDWHILE
         2.4.3.  Switch off mower ;
         2.4.4. Empty grass box ;
      ENDWHILE
   ENDWHILE
3.  Clean mower and put away ;
```

10. *You are a judge in the new hit reality TV show Earth's Next Top Professor, in which a number of university professors all compete in a series of weekly tasks to be crowned Earth's Top Professor and win a job teaching in the Department of Applied Studies at the world renowned University of Stocksfield. In order to decide which professor must leave the show each week you score each candidate according to four characteristics:*

- *Teaching ability, from 1–10*
- *Sense of humor, from 1–10*
- *Subject knowledge, from 1–10*
- *Good looks, from 1–10*

Using the HTTLAP strategy design an algorithm that requests a judge for the name of one of the candidates followed by the score the judge is awarding for each of the three characteristics. There is a constraint: the total points value must not exceed 20 points. If the judge awards more than 20 points then a score of 5 is assigned to each characteristic.

Now extend your solution so that the judge is asked for scores for each of the remaining professors.

Now extend your solution so that each of the three judges on the show is asked for their scores for each of the remaining professors.

First solution

```
1.  Ask judge for professor's name ;
2.  Get teaching ability score ;
3.  Get humor score ;
4.  Get knowledge score ;
5.  Get good looks score ;
6.  IF (teaching+humor+knowledge+looks is greater than 20)
```

```
        6.1.  Display 'scores too high' message ;
        6.2.  Set teaching ability score to 5 ;
        6.3.  Set humor score to 5 ;
        6.4.  Set knowledge score to 5 ;
        6.5.  Set good looks score to 5 ;
    ENDIF
```

Second solution

```
1.  WHILE (professors to judge)
        1.1.  Ask judge for professor's name ;
        1.2.  Get teaching ability score ;
        1.3.  Get humor score ;
        1.4.  Get knowledge score ;
        1.5.  Get good looks score ;
        1.6.  IF (total score is greater than 20)
                  1.6.1.  Display 'cores too high' ;
                  1.6.2.  Set teaching ability score to 5 ;
                  1.6.3.  Set humor score to 5 ;
                  1.6.4.  Set knowledge score to 5 ;
                  1.6.5.  Set good looks score to 5 ;
              ENDIF
    ENDWHILE
```

Third solution

```
1.  WHILE (judges to give scores)
        1.1.  WHILE (professors to judge)
                  1.1.1.  Ask judge for professor's name ;
                  1.1.2.  Get teaching ability score ;
                  1.1.3.  Get humor score ;
                  1.1.4.  Get knowledge score ;
                  1.1.5.  Get good looks score ;
                  1.1.6.  IF (total score is greater than 20)
                              1.1.6.1.  Display 'scores too high'
;
                              1.1.6.2.  Set teaching ability score
to 5 ;
                              1.1.6.3.  Set humor score to 5 ;
                              1.1.6.4.  Set knowledge score to 5 ;
                              1.1.6.5.  Set good looks score to 5
;
                          ENDIF
              ENDWHILE
    ENDWHILE
```

11. *Make a single cheese and onion omelette to feed up to three people. The number of people will vary each time. A one-person portion needs two eggs, a pinch of salt and pepper, a small amount of milk (say 3fl. oz/100ml), 1/4 of a small onion, 2 oz (about 60g) of cheese, and a pat of butter (or margarine if you prefer). The milk, eggs, onion, cheese, and butter need to be mixed prior to cooking in a large frying pan. Make sure the pan is not too hot when you add the mixture. Cook until the surface of the omelette is just starting to firm up. Serve and garnish with parsley and grated parmesan cheese if desired.*

```
Find out how many people to feed (up to 3) ;
Add (2
Add (pinch
Add (pinch
Add (100ml
Add (0.25
Add (60 gm
Add (knob
Put pan on heat ;
Mix ingredients in bowl ;
Add mixture to pan ;
WHILE (surface not firming up)
   Cook omelette ;
ENDWHILE ;
Get garnish requirements ;
IF (parsley required)
   Sprinkle parsley ;
ENDIF
IF (parmesan required)
   Sprinkle parmesan ;
ENDIF
Divide omelette by no. people ;
WHILE (portions served no. equal to no. required)
   Serve portion on plate ;
   Add 1 to number portions served ;
ENDWHILE
```

12. *Decorate your bedroom with new wallpaper.*

```
1.  Measure walls ;
2.  Calculate how many rolls of paper to buy ;
3.  Choose and buy wallpaper ;
```

```
4.  Measure paste and water and mix together ;
5.  Set out papering table ;
6.  Cut pieces of paper to fit walls ;
7.  WHILE (not finished)
   7.1.  Apply paste to piece of paper ;
       7.2.  Hang piece of paper ;
ENDWHILE
8.  Clean up ;
```

13. *What difficulties would you have if the requirements of Exercise 5 were changed so that a customer cannot have a plain hamburger but has a choice of either cheese or lettuce and tomatoes – that is, hamburger with cheese or hamburger with lettuce andtomatoes? What facility does the IFconstruct seem to be lacking?*

```
Get cheese requirements ;
IF (cheese required)
   Add cheese ;
ENDIF
IF (cheese NOT required)
   Add lettuce ;
   Add tomatoes ;
ENDIF
```

The IF appears to be missing the ability to automatically add lettuce and tomatoes when cheese is not required.

Projects

StockSnackz Vending Machine

We dealt with our vending machine dispensing snacks throughout the day by a single highly abstract action “Dispense snacks.” Now we have the constructs available to deal with optional and repeated actions we can extend the solution. Using pseudo-code notation, add any iterations and selections necessary to your solution from Chapter 3 to show many individual items being dispensed. You might want to start with the subproblem of dispensing the correct snack that corresponds to the button that was pressed. After that, move on to solving the problem of allowing this to happen repeatedly.

Pushing Button 1 dispenses a milk chocolate bar, Button 2 a muesli bar, Button 3 a pack of cheese puffs, Button 4 an apple, Button 5 a pack of popcorn, while pushing Button 6 displays on the machine’s small screen a summary of how many of each item have been dispensed. Pushing the Buttons 0, 7, 8, or 9 has no effect.

Dispense a single snack:

```
IF (button 1 pressed)
   Dispense milk chocolate ;
ENDIF
IF (button 2 pressed)
   Dispense muesli bar ;
ENDIF
IF (button 3 pressed)
   Dispense cheese puffs ;
ENDIF
IF (button 4 pressed)
   Dispense apple ;
ENDIF
IF (button 5 pressed)
   Dispense popcorn ;
ENDIF
IF (button 6 pressed)
   Print sales summary ;
ENDIF
```

Do this repeatedly:

```
1.  Install machine ;
2.  Turn on power ;
3.  Fill machine ;
4.  WHILE (not the end of the day)
       4.1.  IF (button 1 pressed)
                4.1.1.  Dispense milk chocolate ;
             ENDIF
       4.2.  IF (button 2 pressed)
                4.2.1.  Dispense muesli bar ;
             ENDIF
       4.3.  IF (button 3 pressed)
                4.3.1.  Dispense cheese puffs ;
             ENDIF
       4.4.  IF (button 4 pressed)
                4.4.1.  Dispense apple ;
             ENDIF
       4.5.  IF (button 5 pressed)
                4.5.1.  Dispense popcorn ;
             ENDIF
       4.6.  IF (button 6 pressed)
                4.6.1.  Print sales summary ;
             ENDIF
```

```
    ENDWHILE
```

Stocksfield Fire Service: Hazchem Signs

Using pseudo-code notation, add any selections and iterations necessary to your solution from Chapter 3 to the problem of decoding the hazchem Emergency Action Code.

```
// First character
IF character is 1
   Use coarse spray ;
ENDIF
IF character is 2
   Use fine spray ;
ENDIF
IF character is 3
   Use foam ;
ENDIF
IF character is 4
   Use dry agent ;
ENDIF
// Second character
IF character is P
   Use LTS ;
   Dilute spillage ;
   Risk of explosion ;
ENDIF
...
IF character is Z
   Use BA & Fire kit ;
   Contain spillage ;
ENDIF
// Third character
IF character is E
   Public hazard ;
ENDIF
```

Puzzle World: Roman Numerals and Chronograms

Go back to the problem of translating Roman numbers into decimal that you looked at in Chapter 3. You should be aware that just as English words can be misspelled, so can Roman numeral strings be malformed. For example, the Roman numeral string MCMC is malformed as it contradicts the syntax of the system. Before we can translate a Roman number into decimal we must first determine whether the Roman number is *valid. If it is, then we can translate it into decimal. The rules for forming valid Roman numeral strings are:*

1. *Smaller numerals follow larger numerals (see Rule 3 below). Summing the values of the numerals gives the value of the number.*
2. *The numerals I, X, C, and M (1, 10, 100, 1000 – all powers of 10) may be repeated up to three times in a row. No other numerals may be repeated.*
3. *Sometimes, a smaller numeral may precede a larger one (as in IV). These cases form compound numerals which are evaluated by subtracting the value of the smaller numeral from the larger one. To form a compound numeral all the following conditions must be met:*
    a) *The smaller numeral must be a power of ten (1, 10, 100, 1000).*
    b) *The smaller numeral must be either one-fifth or one-tenth the value of the larger one.*
    c) *The smaller numeral must either be the first numeral in the number, or follow a numeral of at least ten times its value.*
    d) *If the compound numeral is followed by another numeral, that numeral must be smaller than the one that comes first in the compound numeral (i.e., you can have XCI but not IXI).*

Work through the HTTLAP strategy to draft an outline solution to the problem of validating a Roman number.

We have more information now than before. Rule #1 we had already inferred in our workings out in the earlier chapters. Rule #2 formalises something I think we were already getting to. Rule # really clears up the compound numbers IV, XC, XL etc.

The key thing then is, as we said in earlier chapters, the sub-sequences in a roman number descend in value from left to right. Hence 1999 is MCMXCIX -- M (1000) + CM (900) + XC (90) + IX (9). It would not be IXXCCMM as this is in ascending order of sub-sequences.

So, how to validate a number. I would start with a general solution for rule #1:

```
1.  Look at first sub-sequence in the number ;
2.  WHILE (sub-sequences to process)
       2.1.  IF current sub-sequence less than next one in the
number
                2.1.2.  Display 'This number is invalid' ;
             ENDIF
       2.2.  Look at next sub-sequence ;
    ENDWHILE
```

That is, if we get through a whole roman number without a 'This number is invalid' message then it must be valid. Of course, it is incomplete as is does not deal with the compound numbers such as IX where a smaller number does precede a larger one. Therefore, I need to expand upon statement 2.1. What do we know about compound numbers? 3(a)-3(d) above tell us what to look for. In a nutshell, if the smaller number is a power of 10 (i.e., not 5) and it equals 0.1 or 0.2 the value of the larger one and it is the first numeral in the number or it is at least 10 times smaller than the numeral that precedes and the number that follows the compound number is smaller than the first numeral in the compound number then we're ok! Phew.

We could break this down into its components but that requires knowledge of nesting or compound conditions and/or nested constructs which we have not covered yet. Chapter 5 will show us the pseudo-code needed to deal with this problem.

Pangrams: Holoalphabetic Sentences

Using iteration and selection constructs (`WHILE` and `IF`) update the sequence of actions you produced in Chapter 3 for determining whether a sentence is a pangram

```
1.  Start at 'A' ;
2.  WHILE (letters left in the alphabet)
       2.1. Write down current letter of the alphabet ;
       2.2. Move to next letter ;
    ENDWHILE
3.  Starting at the first letter of the sentence
4.  WHILE (letters left in the sentence)
       4.1. Cross off letter on the paper that matches current
    letter ;
       4.2. Move to next letter ;
    ENDWHILE
5.  Start at 'A' ;
6.  WHILE (letters left in the alphabet)
       6.1. IF (letter not crossed out)
               6.1.1. Add 1 to number of letters not crossed out ;
            ENDIF
       6.2. Move to next letter ;
    ENDWHILE
7.  IF (number of letters not crossed out equals zero)
       7.1. Display 'Sentence is a pangram' ;
    ENDIF
8.  IF (number of letters not crossed out greater than zero)
       7.1. Display 'Sentence is NOT a pangram' ;
    ENDIF
```

Online Bookstore: ISBNs

The check digit in an ISBN is calculated by a Modulus 11technique using the weights 10 to 2. This means that each of the first nine digits is multiplied by a number in a sequence from 10 to 2. If you add these products together and then add the value of the check digit, this total sum should be divisible by 11 without leaving a remainder. If it does give a remainder then the check digit does not match the rest of the number and we know the ISBN has been copied down incorrectly. The check digit is calculated in the following manner:

Multiply the first nine digits by 10, 9, 8, 7, . . ., 2 respectively and add theresults. Divide this sum by 11 and take the remainder. Finally, subtract this remainder from 11 to give the check digit. If the value is 10 the check digit becomes "X." For example, we can validate the ISBN 0-14-012499-3 as
follows:

Check digit

$$= \frac{(0 \times 10) + (1 \times 9) + (4 \times 8) + (0 \times 7) + (1 \times 6) + (2 \times 5) + (4 \times 4) + (9 \times 3) + (9 \times 2)}{11}$$

$= 118 \div 11$

$= 10$, remainder 8

So, the check digit = 3 (i.e., 11-8)

Using HTTLAP, write down the basic sequence of actions necessary to calculate the check digit for any ten-digit ISBN (assume the ISBN will be in a raw format without hyphens). Once you have identified the sequence, look to see if you can identify any repeated actions which could be better expressed using an iteration.

```
1.  Multiply first digit by 10 and add result to total ;
2.  Multiply second digit by 9 and add result to total ;
3.  Multiply third digit by 8 and add result to total ;
4.  Multiply fourth digit by 7 and add result to total ;
5.  Multiply fifth digit by 6 and add result to total ;
6.  Multiply sixth digit by 5 and add result to total ;
7.  Multiply seventh digit by 4 and add result to total ;
8.  Multiply eighth digit by 3 and add result to total ;
9.  Multiply ninth digit by 2 and add result to total ;
10. Divide total by 11 and take the remainder ;
11. IF (11 - remainder equals check digit)
       11.1. Display 'ISBN valid' ;
    ENDIF
```

Statements 1 through 9 look like they could go in an iteration. We need to keep track of which digit we're looking at as the numberit is multiplied by depends on its position. We will see how to do this fully in the next chapter but for now, we could do something like this:

```
1. Start at first digit ;
2. WHILE (not at end of ISBN)
      2.1. Multiply current digit by 11 - current position and add
result to total ;
      2.2. Move to next digit ;
   ENDWHILE
2. Divide total by 11 and take the remainder ;
3. IF (11 - remainder equals check digit)
       3.1. Display 'ISBN valid' ;
   ENDIF
```