

Chapter 1 Solutions and additional material for exercises.

(many of these exist in the old solutions pdf's)

Class activity p 6

1. Since it is a family company we would probably use interview and observation over questionnaires (since it is a small number of employees). Through these two techniques we could get a good idea of how the system works and where the data is stored. Existing documents would be worth study since they can tell us the structure of the files currently used to run the stock control application. We would need these details to design the record card or internet-based system. We might search the literature to see if this process has been undertaken by similar businesses already.
2. Comparing the systems:

System	1 manual	2 existing	3 internet
a) costs	This would be low cost as no extra computer equipment is needed.	This would be low cost as no extra computer equipment is needed.	High cost to buy a web server and implement a tailor-made application.
b) requirements	No extra connections although a second POS terminal for entering orders could be considered.	No extra connections although a second POS terminal for entering orders could be considered.	Needs at least a modem and dedicated phone line – preferable is a broadband connection.
c) time	Seems that little time would be required.	No time needed to implement.	Would take some time to develop and pilot.
d) impact on staff	At the end of the day, the data from the record cards needs to be entered into the stock database via the POS terminal. This is extra work.	None. Notice we haven't considered the customer who might be unhappy with this solution.	Staff need to learn to use new software.

When completing the feasibility report, this table could be presented as an “executive summary”. You would probably want to include a section on expected benefits to the customer or problems that might arise (for example, we haven't considered whether the hand-writing of telephoned orders could lead to mistakes and mis-understandings). Remember that not all technically feasible solutions are practical for all types of business and that a finally chosen solution could be a mix or compromise between some of those examined.

3. Once a new system is chosen and is up and running then it should be analysed to see how well it is working and whether further improvements are possible.

Exercise 1.1 p 9

1. a) A general applications package, a specific applications package or a tailor-made solution.
 b)

package	advantage	disadvantage
general applications package	Lowest cost, probably most reliable, training courses almost certainly available.	May not fit well with existing work flow.
specific applications package	Moderate cost, software is adapted to most needs of company.	Training could be expensive, may not have all desired features.
tailor-made software	Does everything the company wants, staff can have input to development; may be able to sell commercially and offer training later.	Expensive.

- c) “Big bang”, phased implementation or parallel running.
 d)

method	advantage	disadvantage
big bang/direct changeover	Changeover time is short, if the new system works as expected this is the simplest way to go.	Could be catastrophic for business if the new system fails and the old one discarded.
phased implementation	Catastrophic failure is unlikely; bits that don't work can revert to old system while problems are fixed.	Output of one phase of the system may not be compatible with input required to next (eg computer –produced text might have to be glued onto hand-drawn card)
Parallel running	Safest way, you can compare the output of old and new directly.	Twice as much work for staff.

- e) The analyst could provide samples of cards produced by a similar system elsewhere, (s)he could provide prototype screens to show how it would work when completed. The analyst might give some facts/projections on time spent on creative, design activities instead of routine work with the new system.

2. a)

	method	advantage	disadvantage
stock control	big bang/direct changeover	Staff don't have any extra work today, except training	If the system fails, staff might have to resort to manual control of stock.
	phased implementation	Could introduce for a small subset of goods, see if it works properly. Easier to manage transition this way.	More time is spent changing over, possible confusion as to which goods are being tracked by which system.
	parallel running	Can make sure new system gives same results as old one on stock levels.	Twice as much work to run both systems.
patient monitoring	big bang/direct changeover	Don't have to run both systems at the same time.	If this system fails, patients could be put in danger.
	phased implementation	Staff could monitor just one patient at first, using the new system, makes less work because they can actually go and look at a single patient easily.	Staff might get confused as to where they are supposed to look to find critical information on patients being monitored.
	parallel running	Safest way, if patient is in danger old system will still tell you.	Staff might get confused if systems give different outputs.

b)

Item	normal data	data at limits	extreme data	abnormal data
stock level	23 (just is normal – what more to say?)	0 (must test out of stock)	100000 (or suitably large number)	90.67 (stock is in whole numbers of items)
expiry date	12/03/06 (just is normal!)	29/02/2008 (leap year but valid)	30/02/2007 (invalid date)	31st June, 1998 (wrong type of entry)

c) Hardware needed, software needed, criteria for success, personnel involved, project schedule.

d)

GUI	CLI
<p>Can have large visual display showing patients critical readings, perhaps projected to wall. Could use a touch screen system for getting more detail (touch a patient ID to see more). Could be easier to use for novice nurses.</p> <p>Can have big flashing screen effect when patient is in danger.</p>	<p>Can also have a big text display, perhaps scrolling through each patient on regular basis. Harder to learn text commands.</p> <p>Might be harder to read data at a distance</p> <p>Likely that either system would be provided with a loud alarm and perhaps flashing light as well so choice of interface may not be critical.</p>

Exercise 1.2 p 15 - see exercise 3.1 of xcompch3sol.pdf

Exercise 1.3 p 16 — see exercise 3.2

Exercise 1.4 p 17 — see exercise 3.4

Exercise 1.5 p 19

- a. It depends a little bit what the researcher is doing with the images. Pretty clearly if he wants to edit them in some way then a GUI is a very useful tool. However, sometimes the preparation of images requires simple activities like resizing, re-sampling and changing format. These activities could be carried out using a batch process help in a text file and run from a CLI.
- b. When using a CLI the technician has to remember the commands to copy files; but if she uses the interface frequently he probably knows the syntax of these by heart, thus it will be quicker for her and use only a small amount of system resources on a low specification machine.
- c. Similar considerations apply to the student. In fact many GUI's and sophisticated IDE's will provide a graphical user interface for doing this. The popular ANT (<http://apache.org>) tool is a good example of compilation via CLI and it is useful to demonstrate this to students if teachers have the time. One version can be run in a Windows command line window.
- d. The task can be done easily in either interface but since this is a new user, a GUI is probably more appropriate.

Chapter 2 solutions

The code for many of these exercises is to be found in the chap2code.zip file, downloadable from <http://www.ibid.com.au>).

Exercise 2.1 — p 45

All the locations hold the value 13.

Exercise 2.2 — p 48

The result is a String: "My name isMichael Moore3"

The result is double: 4.6

The result is numeric (int, double, depends on type of length): (length + 23)

"Dhanasun says \"Hello\" to you\\me\\everyone"

Exercise 2.3 — p 50

```
x = 2.3; // parenthesis are not needed.
x = - 1;
x = 0; // parenthesis not needed as % comes first in the expression.
x = -20; // parenthesis not needed.
```

Exercise 2.4 — p 59

```
char tempType
double temp
double result
```

Exercise 2.5 — p 62

```
/**
 * The CarRental Program from book p 62
 *
 * @author Richard
 * @version 20041220
 */
public class CarRental
{

    public static void main(String[] args)
    {
        new CarRental();
    }
    /**
     * Constructor
     */
    public CarRental()
    {
        int distance = inputInt("Distance travelled? ");
        double cost;
        if (distance < 100)
        {
            cost = distance * 20.0;
        }
        else
        {
            cost = distance * 25.0;
        }
        output("The cost is: " + cost);
    }
    /**
     * IBIO methods, (c) International Baccalaureate 2003
     * Computer Science Subject Guide, Appendix 2.
     */
}
```

Exercise 2.6 — p 64

```
if (x == 9)
{
    if (y < 3)
    {
        if (c != 'x')
```

```

    {
        output("flip");
    }
    else
    {
        if (z >= 3)
        {
            output("flop");
        }
        else
        {
            output("fly");
        }
    }
}
}
}

```

output: "flop"

Exercise 2.7 — p 65

(There are a number of solutions, here we work it in the same order as the question)

```

if (quantity >= 120)
{
    discount = 10;
}
else if (quantity >= 50)
{
    discount = 5;
}
else if (quantity >= 15)
{
    discount = 1;
}
else
{
    discount = 0; // strictly speaking, this clause not needed!
}

```

Notice that there are problems if we don't use the else part in the chain, eg:

```

if (quantity >= 120)
{
    discount = 10;
}
if (quantity >= 50)
{
    discount = 5;
}
etc

```

(What will go wrong in this case?)

Exercise 2.8 — p 76

The assumption for this method must be that both times are on the same day, since the day is not specified anywhere).

Thus, the simplest solution would be to add 12 hours to the time if it is PM, then subtract. What? You still want me to write it for you?

```
public int timeDifference(Time t)
{
    int t1 = this.iHour;
    int t2 = t.iHour;

    if (this.indicator == 'p')
    {
        t1 = t1 + 12;
    }
    if (t.indicator == 'p')
    {
        t2 = t2 + 12;
    }
    return Math.abs(t1 - t2);
}
```

The `this` keyword is used to identify the data member for the current instance of the Class. An alternate approach would be to make a static method, in which case there are two parameters:

```
public static int timeDifference(Time tm1, Time tm2)
{
    int t1 = tm1.iHour;
    int t2 = tm2.iHour;

    if (t1.indicator == 'p')
    {
        t1 = t1 + 12;
    }
    if (t2.indicator == 'p')
    {
        t2 = t2 + 12;
    }
    return Math.abs(t1 - t2);
}
```

These methods belong in the Time Class.

Exercise 2.9 — p 79

One way to achieve this outcome is to use a boolean flag:

```
boolean gotTime = false;
// had to move this outside while block and initialize it
Time now = new Time();
do
{
    String sHour = input("Please input the hour (0-11): ");
    String indicator = input("Please indicate am or pm (a, p): ");
    try
    {
        now = new Time(sHour, indicator);
    }
}
```

```

        gotTime = true;
    }
    catch( TimeClassException e )
    {
        output("Something wrong " + e.getMessage());
    }
} while (!gotTime);
output("The time is " + now.toString());public boolean getTime()

```

Exercise 2.10 — p 83

The algorithms need another int variable to keep track of where the pointer, pos, is. Eg largePos. This should be initialized to zero and changed whenever a new largest/smallest is found:

```

smallest = listA[pos];
smallPos = pos;

```

Exercise 2.11 — p 86

An example:

```

/**
 * set the length
 *
 * @param int the video length in minutes
 * @exception if tape length < 0 or > 500
 */
public void setLength(int length) throws VideoTapeException
{
    if ( (length <0) && (length > 500) )
    {
        throw new VideoTapeException("Bad tape length");
    }
    else
    {
        this.length = length;
    }
}

```

Exercises 2.12 — p89, 2.13 — p 95 and 2.14 — p 103

These exercises are left for students to do – otherwise we will have written a whole SL or HL dossier, thus inviting plagiarism.

It should also be noted that the ideas presented in these exercises should be regarded as a starting point for further exploration – especially for capable students.

Exercise 2.16 — p 107

This is a useful exercise for developing an understanding of array algorithms and for an introduction to sorting. I have found it necessary to introduce simpler array operations before attempting this with some classes. Some students get confused with String comparisons and find it conceptually easier to deal with int arrays.

See <http://www.ib-computing.com/java/arrays/arrayOperations.html> for an example Applet and exercises.

Trace of the initial (incorrect) algorithm which works for the given data set:

i	j	j <= LAST P	names[i] >= MARKER	names[j] == MARKER	names					
					0	1	2	3	4	5
0	0	T	T	T	XXXX	XXXX	sayaka	jithan	XXXX	andrew
	1			T						
	2			F	sayaka		XXXX			
1	3	T	T	F		jithan		XXXX		
2	4	T	F							
3	5	T	T	F			andrew			XXXX
4	6	F								
		ends								

Observation: Just because an algorithm gives correct results on one run, don't assume it will be correct for all data input to it:

i	j	j <= LAST P	names[i] >= MARKER	names[j] == MARKER	names					
					0	1	2	3	4	5
0	0	T	F	T	XXXX	XXXX	phuong	XXXX	richard	XXXX
	1			T						
	2			F	phuong		XXXX			
1	3	T	F	T						
	4			F		richard			XXXX	
	5	T	F	T						
	6	overruns end of array								

There are many problems with the existing algorithm. Having traced through with a couple of data sets, students will probably focus on the incorrect terminator condition and add a check that J only advances if it is less than the array size, ie change the inner while loop to:

```
while ( (names[j] == MARKER) && (J < 5) )
```

However, the algorithm is more fundamentally flawed than this. The programmer has mixed up the pointers, the outer loop should not be checking the entry at i and incrementing i immediately since i is the pointer to the new array. It should only ever be necessary to check what values are at position j.

This situation is typical of programmers who try to solve the problem by fiddling with code at the computer. Students should learn to sit back and think the problem through with pencil and paper. In this case it might be useful to use pieces of paper with values written on them and two coloured pencils (or other pointers) to represent I and J as you walk through the process.

One of my favourite teaching aids is a large card marked with boxes for array locations, a box for a temporary (temp) store and coloured pointers labelled i, j, k, pos etc. It is quite effective when used with a large-scale trace table – on A3 paper say.

The following method is in the Chapter2Code examples (XandShuffle Applet):

```
private void shuffle(String[] names)
{
    int place = 0;
    int search = 0;
    // Check array for deleted entries
    while (search < MAX)
    {
        // current search name deleted?
        if (!names[search].equals("XXXX"))
```

```

    {
        // no, copy down array
        if (place != search)
        {
            names[place] = names[search];
            names[search] = "XXXX";
        }
        place = place + 1;
    }
    // advance to next position
    search = search + 1;
}
}

```

I've discussed this solution at length because I have found it a useful exercise with students since it is not too complicated an algorithm and should be accessible to almost everyone in the class. If your better ones can see the solution and get bored, ask them to illustrate the process at the console/as an applet for the other students.

At the end of the exercise we should have a good idea of what makes a complete data set for testing, examples:

0	1	2	3	4	5
XXXX	XXXX	sonic	richard	XXXX	mikk
XXXX	XXXX	XXXX	XXXX	XXXX	XXXX
yuwei	bich tram	jacky	fei fei	jessie	angel
minh	XXXX	XXXX	XXXX	XXXX	XXXX
shan	chervonne	XXXX	XXXX	XXXX	XXXX

Exercise 2.17 — p 110

1 Suggested test data would be:

- a. a null string
- b. a string with only one token
- c. a string that ends with a space
- d. a string that begins with a space
- e. strings where words are split by other characters, such as a comma

2 What happens (see Chapter2Code samples for proof)

- a. terminates properly, data remains empty
- b. works correctly, word is placed in data[0]
- c. works correctly
- d. data[0] contains the null string.
- e. these characters are included as part of the token

3 The extra words are ignored.

4 You could make a humungous (very large) while condition as in:

```

while ( (message.charAt(pos) != SPACE) &&
        (message.charAt(pos) != COMMA) &&
        (message.charAt(pos) != PERIOD) &&
        (pos < len) )

```

But you would be better off using some kind of function such as `ISPUNCTUATION` which tests for all the possibilities and returns a boolean value. Elegance would be improved if all possible punctuation marks were placed in an array and tested in a loop.

Java has two ways of tokenizing strings – the `java.util.StringTokenizer` Class and `String.split(String)` which carry out the same functions as this Class. However, the standard String algorithms will always repay study since they often appear in examination questions from time to time.

Exercise 2.18 — p 112

To use the `StringTokenizer`, import `java.util.*`;

```
else if (cmd == 'C')
{
    StringTokenizer t = new StringTokenizer(message, " ");
    output("There are: " + t.countTokens() + " words in the message");
}
else if (cmd == 'T')
{
    StringTokenizer t = new StringTokenizer(message, " ");
    message = "";
    while (t.hasMoreTokens())
    {
        String word = t.nextToken();
        String first = word.substring(0,1);
        String rest = word.substring(1);
        first = first.toUpperCase();
        message = message + first + rest + " ";
    }
    output(message);
}
else if (cmd == 'F')
{
    // Get String to search for, if none, do nothing
    String find = input("String to search for: ");
    if (!find.equals(""))
    {
        find = find.toLowerCase();
        String testMessage = message.toLowerCase();
        int index = testMessage.indexOf(find);
        if (index >=0)
        {
            while (index >= 0)
            {
                output("That String is found at position " + index + "\n");
                index = testMessage.indexOf(find, index + 1);
            }
        }
        else
        {
            output("String not found in: " + message);
        }
    }
}
else if (cmd == 'E')
{
    message = getMessage();
}
```

Exercise 2.19 — p 113 (labelled 2.15)

```
while ( (pos < SIZE) && (scores[pos] != -99) && (scores[pos] != -999));  
  
..  
  
if ( (scores[pos] == -99) || (scores[pos] == -999))  
{  
    reeturn pos;  
}  
else  
{  
    retun -1;  
}
```

Exercise 2.20 — p 120 (labelled 2.19)

```
top = SIZE - 1;  
while (top > 0)  
{  
    upper = 1;  
    while (upper <= top)  
    {  
        int lower = upper + 1;  
        if (data[upper] < data[lower])  
        {  
            //swap as before  
        }  
        upper =upper + 1;  
    }  
    top = top - 1;  
}
```

b)

```
top = SIZE - 1;  
while ( (top > 0) && (swapped) )  
{  
    upper = 1;  
    swapped = false;  
    while (upper <= top)  
    {  
        int lower = upper + 1;  
        if (data[upper] < data[lower])  
        {  
            //swap as before, except for:  
            swapped = true;  
        }  
        upper =upper + 1;  
    }  
    top = top - 1;  
}
```

Exercise 2.21 — p 125 (labelled 2.20)

One strategy is “exhaustive” search – ie search every cell in tern, either by col then row or by row then column.

The second is more efficient:

Check column 0 in each row
if the first value is less than the wanted value
 binary search the row
else
 move on to the next row

The dimension of c will have a big effect on this second algorithm. If it is very small then a linear search of each row will be much more efficient.

To sort the array by both row and column involves examining the first value in each row to see where the row belongs. Then the values in the two rows can be exchanged. It could be like a bubble sort of each first element but the swap gets a little longer (although not more complex, really).

```
public void sortMatrix(int[][] matrix)
{
    for(int top = (ROW - 1); top > 0; top--)
    {
        for(int upper = 1; upper < top; upper++)
        {
            int lower = upper - 1;
            // compare values at start of each row
            if (matrix[upper][0] < matrix[lower][0])
            {
                // swap row values from 0 to col
                for(int s = 0; s < COL; s++)
                {
                    int temp = matrix[upper][s];
                    matrix[upper][s] = matrix[lower][s];
                    matrix[lower][s] = temp;
                }
            }
        }
    }
}
```

Notice that this arrangement does not give rise to a more efficient approach to searching the array (at least, I haven't been able to determine one).

For very large array dimensions it might even be quicker to merge the rows into a linear array and binary search that. The time taken to merge only increases as a linear function of array total size. Might make an interesting big O investigation for an HL student.

Exercise 2.22 — p 125 (labelled 2.21)

- a) There are 12×31 elements or 372
- b) null
- c) It's big, clumsy and mostly empty. I can allocate birthdays to non-existent dates such as 31st February. What can I do if two people share a birthday? Two people with different birthdays might have the same name – I could get into a lot of trouble that way.
- d) $27 \times 372 = 10044$ bytes, about 10KB
- e) $372 - 366 = 6$ elements.
- f) There does not seem to be a great virtue in storing the data as a huge calendar, that's just transferring an existing structure into computer form without thought. The real question is, what do we want to do with the data? My guess is we want to search it by name or date. A better way is to store the data in a simple linear array ordered by either name or date.

Whichever search is most frequent could use a binary approach if the list gets large, the other data then has to be searched using linear search.

- g) Let's give the BirthdayData Class a toString() method:

```
public String toString()
{
    String gP;
    if (getGetsPresent())
    {
        gP = "yes";
    }
    else
    {
        gP = "no";
    }
    return getName() + ", " + getSex() + ", " + gP;
}
```

Now locate the data and return it's toString value:

```
public String getDetails(int day, int month)
{
    return BirthDays[day][month].toString();
}
```

see xcompch2sol.pdf

Exercise 3.1 — p 139 - same as 2.1

Exercise 3.2 — p 140 - same as 2.2

Exercise 3.3 — p 147 - same as 2.6

Exercise 3.4 — p 149 - same as 2.7

Exercise 3.5 — p 150 - same as 2.5

Exercise 3.7 — p 157 - same as 2.4 but note Q1 has been removed from the new book

Exercise 3.8 — p 165 - same as 2.8

Exercise 3.9 — p 166 - same as 2.13

Exercise 3.10 — p 168 - same as 2.14

Exercise 3.11 — p 170 - same as 3.7 in xcompch3sol.pdf

Exercise 3.12 — p 172

questions 1 – 4: see the glossary in the Computer Science Subject Guide

5 For you to do.

6 For you to do.

Exercise 3.13

1. Without compression 300 bits or 75 bytes of data. Without compression I made it 24 bytes.
2. Once you get over 15 blocks of the same colour you have to restart the count. Could overcome by using more bits in the run number. Notice that this would add to the size of the compressed file in the above example.
3. Good for images with large runs of the same colour (“blocky” images). Bad for images where adjacent pixels are often different – eg a painting.
4. Boring. Do I have to? even with the compressed image that’s 48 digits. Let’s just do the 3 bytes in the example: 120114 (you get the idea).

Exercise 3.14 (labelled 3.13 again) — same as 2.10 (1-11)

12 The hardware only operates at the bit level. Software is required to add such processes as error checking, flow control and higher level functions such as network address translation.

13, 14 – Depends on your school’s network.

15. For you.

16. Same as 2.11 1

17. 2.11 – 2

18. 3

- 19. 4
- 20. 5
- 21. 6
- 22. 7
- 23. 8
- 24. Students should be able to describe aspects such as: parity checks, Block Character Checks, flow control, transmission speed, information contained in a datagram and so on.
- 25. For you to do.

Exercise 3.15 (labelled 3.14) on p 185 same as 2.12

Exercise 3.16 (labelled 3.15) on p 187

- 1 same as exercise 2.16 q1
- 4 same as 2.16 q4
- 5 same as 2.16 q5

Exercise 3.17 (labelled 3.16) on p 187 same as 2.21

Exercise 3.18 (labelled 3.17) on p 189

- 4 Because the bytes that represent a character can equally well be interpreted as an unsigned integer value. A given number of bits can be interpreted in many different ways – which is why variables are typed in Java and C.
- 5 It doesn't compile; possibly we were trying to do something like this:

```
char x = 's';
x++; x++;
output(x);
```

In which case the output is 'u'. It raises some interesting issues re the + operator in Java (which is polymorphic), since:

```
char x = 's';
x = x + 2;
output(x);
```

Gives rise to a possible loss of precision error since the + operator converts x to a numeric value. We can use a cast:

```
char x = 's';
x = (char) (x + 2);
output(x);
```

NB: You have to be using the version of IBIO published in the guide, not the earlier version which was used in some of the (chap2code.zip) examples. To add this behaviour, extend the IBIO with these methods:

```
static void output(String info) { System.out.println(info); }
static void output(double info) { System.out.println(info); }
```

```

static void output(int info)    { System.out.println(info); }
// make sure these are included:
static void output(char info)  { System.out.println(info); }
static void output(long info)  { System.out.println(info); }
static void output(boolean info){ System.out.println(info); }

```

Remember that the IBIO methods and any related type conversion issues they raise will NOT be examined.

Exercise 3.19 (labelled 3.18) on p 192

- 1 1111011, 7B
- 2 45, 2D
- 3 431
- 4 110101111 (or 000110101111)

Exercise 3.20 (labelled 3.19) on p 192

- 1 4096
- 2 1, 16, 256, 1 5 and 9.
- 3 4096

Exercise 3.21 (labelled 3.20) on p 196

- 1 111, 111111, 11111111, 111111111. Yes.
- 2 7, 3F, FF, 1FF
- 3 B76
- 4 4F
- 5 21
- 6 2999

Exercise 3.22 (labelled 3.21) on p 198

- 1 110100, 111111, 111110
- 2 for example:

```

110100
001100
-----
0000000

```

Exercise 3.23 (labelled 3.22) on p 199 — same as 2.15

Exercise 3.24 (labelled 3.23) on p 200 — same as 2.18

Exercise 3.25 (labelled 3.24) on p 205 — same as 2.22

Exercise 3.26 (labelled 3.25) on p 208 — same as 2.23

Chapter 4

Exercie 4.1

1. (i) 111, (ii) 1010, (iii) 10000000
2. (i) 1, (ii) 10, (iii) 0, (iv) 1011
3. (i) 22, (ii) 1DC, (iii) 159, (iv) 25CD (easy to get wrong!!)

Exercise 4.2

- (i) Unsigned

32	16	8	4	2	1
0	0	0	0	0	0
1	1	1	1	1	1

Min = 0

Max = $32+16+8+4+2+1 = 63$

Signed would allow -31 to $+31$ using the MSB as sign bit

- (ii) Max = $2^{(6-1)} - 1 = 2*2*2*2*2 - 1 = 31$, likewise min = -31
(iii) With n bits, if all but the MSB set to 1 we have $2^n - 1$, if only MSB set to 1 we have 2^n .

Exercise 4.3 (OLD Exercise 4.6)

Exercise 4.4 (OLD Exercise 4.7)

Exercise 4.5 (OLD Exercise 4.8)

Exercise 4.6 (OLD Exercise 4.9)

Exercise 4.7 (OLD Exercise 4.10)

Exercise 4.8 (OLD Exercise 4.11)

Exercise 4.9 (number line)

Exercise 4.10 (OLD Exercise 4.13)

Exercise 4.11 (OLD Exercise 4.16)

Exercise 4.12

1. fixed point uses an implied decimal point position to represent real numbers
floating point uses a mantissa and exponent.
2. normalisation refers to adoption of a standard way to represent the mantissa ie 0.1 etc, where a 1 must always follow the decimal point. Using non-normal form would allow many different ways to represent the same number.

123.0 in decimal can be written
 $0.123 * 10^3$ or $1.23 * 10^2$ or $12.3 * 10^1$ etc

3. Not possible is it, there are not sufficient bits to represent the size of the mantissa!
4. $234.5 = 0.2345 * 10^3$ (in standard maths it would be $2.345 * 10^2$)
Mantissa = 0.2345, exponent is 3
5. $234.5 = 1110100.1 = 0.11101001 * 2^7$

Mantissa = 0.11101001
Exponent = 7

6. No, overflow of the mantissa.
7. The mantissa would need to be at least 9 bits and the exponent 4
8.
Largest Positive Magnitude = $0.11111111 * 2^{0111}$
Largest Min Magnitude = $1.00000000 * 2^{1000}$

6. Should attempt to detect unintended results in software by testing for less than zero. Can inspect higher order bit ie MSB to see if it is zero etc. Java raises no Exception error as such.

Exercise 4.14 is OLD exercise 4.19

4.15 is OLD 4.20

4.16 is OLD 4.21

4.17 is OLD 4.22

4.18 is OLD 4.23

4.19 is OLD 4.24

4.20 is OLD 4.25

4.21 is OLD 4.26

4.22 is OLD 4.27

4.23 is OLD 4.28

Chapter 5 Solutions

Exercise 5.1 same as before

Exercise 5.2 ditto

Exercise 5.3 ditto

Exercise 5.4: Student investigations, no answers provided.

Exercise 5.5: OLD exercise 5.4

Exercise 5.6: OLD exercise 5.5

Exercise 5.7: OLD exercise 5.6 — remove q. 7.

Exercise 5.8: OLD exercise 5.7

Exercise 5.9: OLD exercise 5.8

Exercise 5.10: OLD 5.9

Exercise 5.11: OLD 5.10

Exercise 5.12: OLD 5.11

Exercise 5.13: OLD 5.12

Exercise 5.14: OLD 5.13

Exercise 5.15: OLD 5.13 p. 264

Exercise 5.16: OLD 5.14 p. 267

Exercise 5.17: OLD 5.15 p. 272

Exercise 5.18: OLD 5.16

Exercise 5.19: OLD 5.17

Exercise 5.20: OLD 5.18

Exercise 5.21:

1. List is long and requires accessing/searching to add and delete

To add we can just append: linked list $O(1)$ at head, bTree OK but need to traverse to locate add point, Array OK if in boundary but need to search for end of list $O(N)$ or if index held $O(1)$.

To delete is $O(N)$ for the array and $O(N)$ to locate if using linear search, $O(\log N)$ if using binary search but the array needs to be resorted! Deletion from a binary tree is easy to code but requires recursion and depends on height and balance of the tree.

To search is fast for binary tree and binary search ie $O(\log N)$ and slow for array and linked list ie $O(N)$. But, binary search requires sorting. If adds and delete are constant and at a reasonable rate this will carry an overhead.

2. File is large in terms of records. At best it is approx. 4m records.

To efficiently access records for editing requires a direct access file, but random

organisation. Each record has a relative record position and the records would be fixed in length, hence each record has a record position and its start point in the byte stream can be calculated. An index could be used to fully or partially index the records. Requires overhead of updating the index, accessing the index and storing the index.

Hashing could be used but the record space would probably need to be larger to minimise collisions. But access would be quick.

Adding could be done by appending, but this means the file length is unbounded. Alternatively, deleted records could be flagged for deletion and the file packed to remove deleted records.

Another alternative would be to linearly search the file for a flagged deleted record and add into that spot $O(N)$!

3. The 100 records could be placed at the start of the file and sequential access used.

4. Look for macro improvements. For very large N small improvements don't make much improvement were as improving from $O(N^3)$ to $O(N)$ provides very large improvements.

5. 6. & 7. Solutions not provided.

Chapter 6

Exercise 6.1: OLD 6.1

Exercise 6.2: OLD 6.3

Exercise 6.3 and 6.4 p. 337 and 338 have got stuffed up!

Thus Exercise 6.4: OLD exercise 6.4

Exercise 6.5

1. Role: coordinate network and respond to requests from clients, provide security
2. A gateway allows connection to a separate network eg from LAN to a WAN
3. ISP: provides connection to the Internet
4. Firewall and proxy can provide security by intercepting requests and checking these before passing onto the actual server.
5. Ethernet Standard
LAN standard controls the physical and lower level software connections.
features include: LAN medium used, protocol used, error detection and collision handling (see: <http://www.lantronix.com/learning/networking.html>)
6. ISDN: international standard for transmitting voice, video and data over telephone lines, 64,000 bits per second.

7. ADSL: new technology allows very high speed broadband connecton over traditional telephone lines, allows phone to be used normally.
8. Fibre Optics: transmission of data via lighth signals sent over glass fibres. Very high speed, high cost, low error rate.
9. Wireless: uses radio frequency, good speed, good mobility, reasonable cost, no cabels, need transmitter base station to be present.

10. ISDN V s ADSL — ADSL if available can be a cost effective way to provide braod band connection. ISDN is not suitable for applications requiring high speed high volume connections. ISDN is fine if you have text level demands eg email, fax etc.

11. ADSL is appropriate for home users who have exisitng phone connections as no additional equipment is required. Especially appropriate if home user requires high bandwidth access to the internet.

12. Fibre Optics: very high speed and thorough put is required, high security and low interference required.
(see this neat site: <http://www.datacottage.com/nch/fibre.htm>)

13. Wireless is flexible and provides mobility, but there is possible security problems, distance to base station can be an issue, speed will be slower, but developments are occuring all the time in this area.. Direct cabel connection will provide high speed, very reliable technology, can be made secure in dedecated format. Not as mobile. Both have costs associated especially setup and change over costs to wireless.

Exercise 6.6

(delete q. 3,4,15, 16, 17 from OLD 6.5. The remaining questions and answers then match ie copy over)

Chapter 7

Exercise 7.1 OLD 8.1

Exercise 7.2 OLD 8.2

Exercise 7.3 OLD 8.4

Exercise 7.4 OLD 8.5

Exercise 7.5 OLD 8.6

Exercise 7.6 programs provided on webSite

Exercise 7.8 programs provided on webSite