Tutorial letter 102/3/2014

Introduction to Programming 1

COS1511

Semesters 1 & 2

School of Computing

IMPORTANT INFORMATION:

This tutorial letter contains the answers to the exercises in the study guide.

BAR CODE

TUTORIAL MATTER

Up to now you should have received the material listed below. If not, please **download it from *myUnisa* immediately** (see below) and also contact the Department of Despatch by sending a sms including your query and student number to 43579, or email info@unisa.ac.za.

Study Guide

DISK 2014 (with software)

Tutorial letters

COSALLF/301/4/2014
General information concerning the School and study at Unisa

COS1511/101/3/2014

Information about COS1511

Assignments

COS1511/102/3/2014 (this letter)

Please note the following

- For e-mail, use the module address namely cos1511-14-S1@unisa.ac.za if you are registered for the first semester and cos1511-14-S2@unisa.ac.za if you are registered for the second semester.

- Web addresses for downloading of tutorial matter
  my.unisa.ac.za

- You need AT LEAST 8 hours study time per week for this module.

Other contact numbers (include your student number in your enquiry)

- For enquiries about registration, send a sms to 43578
- To contact the assignment or examination section, send a sms to 43584
- If you do not receive your study material, send a sms to 43579
- For problems with *myUnisa*, send a sms to 43582
- For academic queries in the School of Computing, email the lecturer – see *myUnisa*.
- For any non-academic query you may also send and email to info@unisa.ac.za

Afrikaanssprekende studente: Studiemateriaal vir COS1511 is slegs in Engels beskikbaar. As enigiets onduidelik is, is u baie welkom om ons te kontak.

Your programs need not be identical to ours. It is, however, important that your programs yield the correct results (thus extensive testing is essential) and that you use good programming style as explained in the Study Guide. You should, for example,

- **use meaningful names for variables, constants, functions, etc,**
- **indent your code correctly,**
- **include comments,**
- **declare constants where appropriate,**
- **avoid the use of global variables,**
- **use the correct types of functions as well as**
- **the correct types of parameters.**

LESSON 1

Exercise 1.1

We repeat the program here:

```
#include <iostream>
using namespace std;
int main( )
{ cout << "Hello world"; return 0; }
```

Descriptive Comment:

There is no descriptive comment.

StandardHeaderFile:

iostream

StatementSequence:

```
cout << "Hello world"; return 0;
```

Exercise 1.2

```
//A poem
#include <iostream>
using namespace std;
int main( )
{
  cout << "Twinkle, twinkle, little bat!" << endl;
  cout << "How I wonder what you're at?" << endl;
  cout << "Up above the world you fly," << endl;
  cout << "Like a tea-tray in the sky." << endl;

  return 0;
}
```

LESSON 2

Exercise 2.1

```
(i)       ((80 / 5) + (70 / 6))
               |          |
              16    +    11
                    |
                   27


 (ii)     ((-5 + -4) - -3)
                   |
                -9     - -3
                   |
                  -6
```

```
(iii)     (((6 * 7) / 8) * 9)
                    |
               42   / 8  * 9
                    |
                 5    * 9
                    |
                   45


 (iv)     ((1 - 2) + ((3 / 4) * 5))
           |          |
          -1    +     0    * 5
                |
          -1    +          0
                |
               -1


(v)       ((-1 + (23 / -4)) + 56)
                  |
              -1 +    -5      + 56
                   |
             -6            + 56
                   |
                  50
```

Exercise 2.2

```
//Lesson 2  Exercise 2.2
//display number of seconds in a minute, hour, day and year
#include <iostream>
using namespace std;
int main()
{
  cout << "The are 60 seconds in a minute." << endl;
  cout << "The are " << 60 * 60 << " seconds in an hour." << endl;
  cout << "The are " << 60 * 60 * 24 << " seconds in a day." << endl;
  cout << "The are " << 60 * 60 * 24 * 365 << " seconds in a year." << endl;
  return 0;
}
```

Exercise 2.3

```
//Lesson 2  Exercise 2.3
#include <iostream>
using namespace std;

int main( )
{
  cout << "The remainder of 234 divided by 13 is ";
```

```
  cout << 234 - (234 / 13) * 13 << endl;
  return 0;
}
```

The round brackets are not necessary but their use makes the program more readable.

LESSON 3

Exercise 3.1

```
//Inputs three numbers and displays them in reverse order
#include <iostream>
using namespace std;

int main( )
{
  int i, j, k;
  cout << "Enter three numbers: ";
  cin >> i >> j >> k;
  cout << "In reverse: " << k << " " << j << " " << i << endl;

  return 0;
}
```

Exercise 3.2

(i) `Enter values for variables x, y and z:`

```
2 6 4
x + y / z is 3
x % z is 2
y * z / x + 2 is 14
```

(ii) `Enter values for variables x, y and z:`

```
5 1 3
x + y / z is 5
x % z is 2
y * z / x + 2 is 2
```

(iii) If `2 6 4` are entered: `y * (z / x + 2) is 24`

If `5 1 3` are entered: `y * (z / x + 2) is 2`

Exercise 3.3

```
//Fahrenheit to Celsius conversion
#include <iostream>
using namespace std;

int main( )
{
  int fahrenheit;
  cout << "Enter the temperature in Fahrenheit: ";
  cin >> fahrenheit;
  cout << "Celsius = " << 5 *(fahrenheit - 32) / 9 << endl;

  return 0;
}
```

LESSON 4

Exercise 4.1

```
//Fahrenheit to Celsius conversion (version for lesson 4)
#include <iostream>
using namespace std;

int main( )
{
  int fahrenheit, celsius;
  cout << "Enter the temperature in Fahrenheit: ";
  cin >> fahrenheit;
  celsius = 5 * (fahrenheit - 32) / 9;
  cout << "Celsius = " << celsius << endl;

  return 0;
}
```

Exercise 4.2

```
//How many boxes?
#include <iostream>
using namespace std;

int main( )
{
  int items, itemsPerBox, boxes, remainder;
  cout << "How many items to be packed? ";
  cin >> items;
  cout << "How many items fit in a box? ";
  cin >> itemsPerBox;
  boxes = items / itemsPerBox;
  remainder = items % itemsPerBox;
  cout << "You will need " << boxes << " boxes." << endl;
  cout << "There will be " << remainder << " items left over." << endl;

  return 0;
}
```

Exercise 4.3

```
int n = 10; // 10
n += 3;     // 13
n /= 2;     // 6
n++;        // 7
n %= 4;     // 3
n -= 5;     // -2
```

The final value of `n` is -2.

LESSON 5

Exercise 5.1

	j	k	m	n
Line 7:	?	?	?	?

	j	k	m	n
Line 8:	3	?	?	?

	j	k	m	n
Line 9:	3	2	?	?
Line 10:	3	2	1	?
Line 11:	3	2	1	6
Line 12:	3	2	1	6
Line 13:	9	2	1	6
Line 14:	9	3	1	6
Line 15:	9	3	1	6

The exact output is:

```
3 2 1 6
9 3 1 6
```

Exercise 5.2

```
1  //Think of a number
2  #include <iostream>
3  using namespace std;
4
5  int main( )
6  {
7    const int NUMBER = 40;
8    int answer;
9    cout << "Think of a number between 30 and 50. Write it down" << endl;
10   cout << "Then do the following calculations on paper:" << endl << endl;
11   cin.get( );
12   cout << "Double it" << endl;
13   cin.get( );
14   answer = NUMBER * 2;
15   cout << "Add 29 to this" << endl;
16   cin.get( );
17   answer += 29;
18   cout << "Double the result again" << endl;
19   cin.get( );
20   answer *= 2;
21   cout << "Subtract the original number from your answer" << endl;
22   cin.get( );
23   answer -= NUMBER;
24   cout << "Divide the answer by your original number and throw away any remainder"
       << endl;
25   cin.get( );
26   answer /= NUMBER;
27   cout << "Your final answer is " << answer << endl;
28   return 0;
29 }
```

	NUMBER	answer
Line 7:	40	?

	NUMBER	answer
Line 14:	40	80

	NUMBER	Answer
Line 17:	40	109

	NUMBER	answer
Line 20:	40	218

	NUMBER	answer
Line 23:	40	178

Line 26:	40	4

LESSON 6

Exercise 6.1

```
//Calculates the area of a room given its length and width
#include <iostream>
using namespace std;

int main( )
{
  float length, width, area;

  //Prompt for and input the measurements of the room
  cout << "Enter the width of the room: ";
  cin >> width;
  cout << "Enter the length of the room: ";
  cin >> length;

  //Calculate the area
  area = width * length;

  //Fixed-point notation, 3 digits after the decimal point
  cout.setf(ios::fixed);
  cout.precision(3);

  //Display the result

  cout << endl << "The area of the room is ";
  cout << area << " square metres." << endl;

  return 0;
}
```

Exercise 6.2

```
//Calculates the area of a room as well as the cost of a carpet
#include <iostream>
using namespace std;

int main( )
```

```
{
  const float PRICE_PER_METRE = 59.50;
  float length, width, area, price;

  //Prompt for and input the measurements of the room
  cout << "Enter the width of the room: ";
  cin >> width;
  cout << "Enter the length of the room: ";
  cin >> length;

  //Calculate the area and the total price
  area = width * length;
  price = area * PRICE_PER_METRE;

  //Fixed-point notation
  cout.setf(ios::fixed);

  //Display the area (3 digits after the decimal point)
  cout.precision(3);
  cout << endl << "The area of the room is ";
  cout << area << " square metres." << endl;

  //Display the total price (2 digits after the decimal point)
  cout.precision(2);
  cout << "The total price is R" << price << endl;

  return 0;
}
```

Exercise 6.3

(i) Implicit conversions take place in line 13 (when the integer quotient of the integers `w` and x is assigned to the floating point variable `y`), line 14 (when the value of the floating point variable `y` is subtracted from the integer value of `w`), and twice in line 16 (when firstly the quotient of the floating point variable `y` and the integer `x` is found and secondly when the integer value of the right-hand side of the assignment statement is stored in the floating point variable `answer`).

Explicit conversions take place in lines 15 and 16.

(ii) The value of `result` will be 11, and the value of `answer` will be 3.

Exercise 6.4

```
Enter three floating point numbers:

14.0    1.123    64.9999

14 1.123 65

Enter two more floating point numbers:

73.46    27.2727

27.273 73.460 65.000
```

LESSON 7

Exercise 7.1

(i) `x + y is 579`

(ii) `x + y is 123456`

(iii) `x + y is c`

In part (iii), the character '1' is stored in `x` and the character '2' is stored in `y`. In the statement `z = x + y`, these character values are implicitly converted to the numerical values 49 and 50 (the ASCII codes of '1' and '2', respectively) and then added to give 99. This number is then implicitly converted to the character with ASCII code 99, namely the character 'c'.

Exercise 7.2

```
//Determines the position of a letter in the alphabet
#include <iostream>
using namespace std;

int main( )
{
  char letter;
  int position;
  cout << "Enter an upper case letter: ";
  cin >> letter;
  position = letter - 'A' + 1;
  cout << letter << " is in position " << position
       << " in the alphabet" << endl;

  return 0;
}
```

Exercise 7.3

```
//Performs a spoonerism
#include <string>
#include <iostream>
using namespace std;

int main( )
{
  string word1, word2, spoonerism;
  char letter1, letter2;

  cout << "Enter two words: ";
  cin >> letter1 >> word1 >> letter2 >> word2;

  spoonerism = letter2 + word1 + ' ' + letter1 + word2;
  cout << "Spoonerised that is " << spoonerism << endl;

  return 0;
}
```

Exercise 7.4

```
//Dialogue generator
#include <iostream>
#include <string>
using namespace std;

int main( )
{
  string name1, name2, colour, noun, adjective;
```

```
  int number;

  cout << "Enter a person's name: ";
  cin >> name1;
  cout << "Enter another person's name: ";
  cin >> name2;
  cout << "Enter a colour: ";
  cin >> colour;
  cout << "Enter a number: ";
  cin >> number;
  cout << "Enter a noun: ";
  cin >> noun;
  cout << "Enter an adjective: ";
  cin >> adjective;
  cout << "\nDialogue" << endl;
  cout << "========" << endl;
  cout << name1 << ":\t\"Couldn't you see that the traffic light was "
       << colour << "?\"" << endl;
  cout << name2 << ":\t\"But I had " << number << " people and a "
       << noun << " in the car with me.\"" << endl;
  cout << name1 << ":\t\"That is so " << adjective
       << "! You could have had them all killed.\"" << endl;

  return 0;
}
```

Exercise 7.5

With the knowledge that you have acquired up to now, it is impossible to do this exercise as it was stated. It is essential to use a loop, but loops have not been discussed yet. (However, if one knows beforehand that 10000 will be entered when the program prompts one to enter a number, one may include 10000 `cout` statements in the program!)

```
//Computer punishment
#include <iostream>
#include <string>
using namespace std;

int main( )
{
  int n;
  string s;
  cout << "Computer punishment" << endl;
  cout << "-------------------" << endl;
  cout << "Repetitions? ";
  cin >> n;
  cin.get( );                        //necessary between cin >> and getline
  cout << "Message? ";
  getline(cin, s, '\n');

  cout << endl << s << endl;
  cout << s << endl;
  cout << s << endl;
  cout << s << endl;
// and so on and so on and so on and so on ... n times

  return 0;
}
```

LESSON 8

Exercise 8.1

`Statement1` is executed in cases (i) and (iv).

Exercise 8.2

```
(i)   if (colour == "red")
            cout << "Correct" << endl;
      else
            cout << "No, blood is red." << endl;
      cout << "What is the colour of the sky? ";
      cin >> colour;

(ii)  cout << "Enter salary: ";
      cin >> parentSalary;
      if (age < 13)
            pocketMoney += parentSalary / 20;
      else
            pocketMoney += parentSalary / 10;

(iii)  if (mark < 50)
            failed++;
       total += mark;
```

Exercise 8.3

```
  if (balance >= 0)
    cout << "Credit" << endl;
  else
    cout << "Debit" << endl;
```

Exercise 8.4

```
if (x == y)
  cout << "x is equal to y" << endl;
else
  cout << "x is not equal to y" << endl;
```

Exercise 8.5

```
// anyone born before 1945 enters free, others pay R20
  int yearBorn;
  const int CUT_OFF = 1945;
  const float FEE = 20.0;
  cout << "When were you born? ";
  cin >> yearBorn;
  if (yearBorn < CUT_OFF)
    cout << "Free entry";
  else
    cout << "Entrance fee R" << FEE;
  cout << endl;
```

Exercise 8.6

```
//Will  the carpet be large enough?
#include <iostream>
using namespace std;

int main( )
```

```
{
  float lenth, width, areaFloor, areaNotCovered;
  const float CARPET_SIZE = 100;

  cout << "Please enter length of room: ";
  cin >> lenth;
  cout << "Please enter width of room: ";
  cin >> width;
  areaFloor = lenth * width;

  if (areaFloor <= CARPET_SIZE)
    cout << "The carpet will cover the floor." << endl;
  else
  {
    areaNotCovered = areaFloor - CARPET_SIZE;
    cout << "The carpet is too small. ";
    cout << areaNotCovered << " square metres will not be covered."
         << endl;
  }

return 0;
}
```

LESSON 9

Exercise 9.1

0 times. If the condition of the while loop is False the very first time the loop is entered, the body of the loop will be skipped.

Exercise 9.2

There are two problems. Firstly, the loop control variable count is not changed inside the body of the loop. Its value stays equal to 0 and so the condition (assuming that num is greater than 0) never becomes False. This means that we have an infinite loop. We have to insert the statement

```
count++;
```

into the body of the loop.

Secondly, if a value less than or equal to 0 is entered for num, the body of the `while` loop will never be executed, and thus no value will be assigned to last. We can correct this by replacing the final `cout` statement with:

```
if (num > 0)
  cout << "The last value entered was " << last << endl;
else
  cout << "Error: the number of items cannot be zero or less." << endl;
```

So we should have:

```
total = 0.0;
cout << "Enter number of values: ";
cin >> num;
count = 0;
while (count < num)
{
  cout << "Enter a value : ";
  cin >> value;
  last = value;
  count++;
}
if (num > 0)
  cout << "The last value entered was " << last << endl;
```

```
      else
        cout << "Error: the number of items cannot be zero or less." << endl;
```

If the idea was to add all the numbers, we would also have to insert the statement

```
         total += value;
```

into the body of the loop after value has been input.

Exercise 9.3

```
//Sipho's money
#include <iostream>
using namespace std;

int main( )
{
  const float INTEREST = 0.045;
  const float START_AMOUNT = 1000.00;
  const float SAVE_AMOUNT = 500.00;
  const int NUM_YEARS = 18;
  float balance;
  int year;

//Set the initial values
  balance = START_AMOUNT;
  year = 1;

//Calculate the accumulation for numYear years
 while (year <= NUM_YEARS)
  {
     balance += balance * INTEREST;
     balance += SAVE_AMOUNT;
     year++;
 }

//Display the balance after NUM_YEARS years
  cout << "The final balance is R" << balance << endl;

  return 0;
}
```

Exercise 9.4

```
//Can all the luggage be loaded on the airplane?
#include <iostream>
using namespace std;

int main( )
{
  const int MAX_MASS = 10000;
  int mass, totalMass;

  totalMass = 0;
  cout << "Please enter mass of first piece of luggage (0 to stop): ";
  cin >> mass;
  while (mass != 0)
  {
    totalMass += mass;
    cout << "Please enter mass of next piece of luggage (0 to stop): ";
    cin >> mass;
  }

  if (totalMass > MAX_MASS)
    cout << totalMass << "kg exceeds the permissible maximum load." << endl;
```

```
  else
    cout << totalMass << "kg is fine." << endl;

  return 0;
}
```

LESSON 11

Exercise 11.1

It is permissible to use an expression in the declaration of a constant - see below.

```
//Water consumption
#include <iostream>
using namespace std;

int main( )
{
  const float RATE_0 = 10;
  const float RATE_1 = RATE_0 * 1.5;       // this is permissible
  const float RATE_2 = RATE_0 * 2.0;       // this is permissible
  float units, amount;

  cout << "Please enter number of units consumed: ";
  cin >> units;

  if (units <= 20)
    amount = 0;
  if ((units > 20) && (units <= 40))
    amount = (units - 20) * RATE_0;
  if ((units > 40) && (units <= 100))
    amount = (20 * RATE_0) + (units - 40) * RATE_1;
  if (units > 100)
    amount = (20 * RATE_0) + (60 * RATE_1) + (units - 100) * RATE_2;

  cout << "Amount due is R" << amount << endl;

  return 0;
}
```

Exercise 11.2

```
//Playschool criteria
#include <iostream>
#include <string>
using namespace std;

int main( )
{
  int ageChild, income, ageParent;
  char status;
  bool crit1, crit2, crit3, crit4;

  cout << "Age of child: ";
  cin >> ageChild;
  cout << "Is parent single? Answer Y or N: ";
  cin >> status;
  cout << "Income of parent: ";
  cin >> income;
  cout << "Age of parent: ";
  cin >> ageParent;

  crit1 = (ageChild >= 3) && (ageChild <= 5);
```

```
  crit2 = (status == 'Y') || (status == 'y');
  crit3 = income < 60000;
  crit4 = ageParent <= 30;

  if (crit1 && crit2 && crit3 && crit4)
    cout << "This toddler should be accepted." << endl;
  else
    cout << "This toddler should not be accepted." << endl;

  return 0;
}
```

Exercise 11.3

```
(i)   highSchool = grade > 7;

(ii)  teenager = (age >= 13) && (age <= 19);
                  or
      teenager = !((age < 13) || (age > 19));

(iii) found = (x >= 0) && (x % 4 == 0);
```

Exercise 11.4

```
//Guess a number between 1 and 100 in 10 or less tries
#include <iostream>
using namespace std;

int main( )
{
  const int SECRET = 23;            // or any other value between 1 and 100
  int guess, numberOfTries;
  bool found;

  found = false;
  numberOfTries = 0;

  while ((!found) && (numberOfTries < 10))
  {
    cout << "Please enter a guess (1-100): ";
    cin >> guess;
    numberOfTries++;
    if (guess == SECRET)
      found = true;
  }

  if (found)
    cout << "Well done! You got the number in "
         << numberOfTries << " guesses." << endl;
  else
    cout << "Tough luck! Your 10 guesses are over." << endl;

  return 0;
}
```

LESSON 12

Exercise 12.1

```
// Is one number equal to the sum of two others?
#include <iostream>
using namespace std;

int main ( )
```

```
{
  int a, b, c;

  cout << "Please enter three numbers, a, b and c: ";
  cin >> a >> b >> c;

  if ((a + b) == c)
    cout << "Yes, a + b = c";
  else if ((a + c) == b)
    cout << "Yes, a + c = b";
  else if ((b + c) == a)
    cout << "Yes, b + c = a";
  else
    cout << "No, one number is not equal to the sum of the other two.";
  cout << endl;
  return 0;
}
```

Exercise 12.2

```
//Two dice throws
#include <iostream>
using namespace std;

int main( )
{
  int throw1, throw2, sum;

  cout << "Please enter 2 numbers representing"
       << " the throws of a pair of dice: ";
  cin >> throw1 >> throw2;
  sum = throw1 + throw2;

  if ((sum == 7) || (sum == 11))
    cout << "You win!";
  else if (sum == 2)
    cout << "Snake eyes!";
  else if (sum == 12)
    cout << "Good shot!";
  else
    cout << "Try again.";
  cout << endl;

  return 0;
}
```

Exercise 12.3

```
//Leap year or not
#include <iostream>
using namespace std;

int main( )
{
  int year;
  bool leap;

  leap = false;
  cout << "Please enter year: ";
  cin >> year;

  if ((year % 100) == 0)
  {
```

```
      if ((year % 400) == 0)
leap = true;
  }
  else
      if ((year % 4) == 0)
        leap = true;

  if (leap)
      cout << year << " is a leap year." << endl;
  else
      cout << year << " is not a leap year." << endl;

  return 0;
}
```

Exercise 12.4

```
//Cereal discount
#include <iostream>
using namespace std;

int main( )
{
  float amount, discountPerc, finalAmount;

  cout << "How much did the customer spend? R";
  cin >> amount;

  if (amount < 50)
    discountPerc = 0.10;
  else if (amount < 70)
    discountPerc = 0.20;
  else if (amount < 100)
    discountPerc = 0.30;
  else if (amount < 200)
    discountPerc = 0.40;
  else
    discountPerc = 0.50;

  finalAmount = amount - discountPerc * amount;
  cout.setf(ios::fixed);
  cout.precision(2);
  cout << "Amount due is R" << finalAmount << endl;

  return 0;
}
```

LESSON 13

Exercise 13.1

```
//Choices of university
#include <iostream>
using namespace std;

int main( )
{
  float mark, earn;
  char category;
  cout << "What was her average mark? ";
  cin >> mark;
  // find category of mark
  if (mark >= 90)
    category = 'A';
```

```
  else if (mark >= 75)
    category = 'B';
  else if (mark >= 60)
    category = 'C';
  else
   category = 'D';

  cout << "What did she earn? ";
  cin >> earn;

  switch (category)
  {
    case 'A':
      cout << "She may go to any university"
           << " and she will get a car." << endl;
    break;
    case 'B':
      if (earn > 5000)
        cout << "She may go to any university"
             << " and she will get a car." << endl;
      else
        cout << "She may go to any university"
             << " but she will not get a car." << endl;
    break;
    case 'C':
      cout << "She has to study at the nearest university." << endl;
    break;
    case 'D':
      cout << "She cannot go to university." << endl;
  }

  return 0;
}
```

Exercise 13.2

```
//Two dice throws (Lesson 13's version)
#include <iostream>

int main( )
{
  int throw1, throw2, sum;

  cout << "Enter 2 numbers representing the throws of a pair of dice: ";
  cin >> throw1 >> throw2;
  sum = throw1 + throw2;
  switch (sum)
  {
    case 7:
    case 11:
        cout << "You win!";
        break;
    case 2:
        cout << "Snake eyes!";
    break;
    case 12:
        cout << "Good shot!";
    break;
    default:
        cout << "Try again.";
}
cout << endl;
```

```
return 0;
}
```

Exercise 13.3

```
// Parkade payment
#include <iostream>
using namespace std;

int main( )
{
 char typeOfVehicle;
 int hours, due;

 cout << "Car or Truck? Please enter C or T: ";
 cin >> typeOfVehicle;
 cout << "How many hours was the vehicle parked? ";
 cin >> hours;

 switch (hours)
 {
  case 1:
    due = 2;
  break;
  case 2:
    due = 3;
  break;
  case 3:
  case 4:
  case 5:
    due = 5;
  break;
  default:
    due = 10;
 }
  if ((typeOfVehicle == 'T') || (typeOfVehicle == 't'))
    due += 1;
  cout << "You owe R" << due << endl;

  return 0;
}
```

Exercise 13.4

```
//Determines the number of days in a given month
#include <iostream>
using namespace std;

int main( )
{
 int month, year, numberOfDays;
 bool leap;

 cout << "Which month (1-12) are you interested in? ";
 cin >> month;
 cout << "Which year? ";
 cin >> year;

 switch (month)
{
   case 1:
   case 3:
   case 5:
   case 7:
   case 8:
```

```
   case 10:
   case 12:
      numberOfDays = 31;
      break;
   case 4:
   case 6:
   case 9:
   case 11:
numberOfDays = 30;
break;
   case 2:
      leap = ( ((year % 4) == 0) && !((year % 100) == 0) )||
         ((year % 400) == 0);
if (leap)
   numberOfDays = 29;
else
   numberOfDays = 28;
break;
   default:
cout << "Error. Month has to be between 1 and 12." << endl;
numberOfDays = 0;
 }
 cout << "The " << month << "th month of " << year
      << " has " << numberOfDays << " days." << endl;

 return 0;
}
```

LESSON 14

Exercise 14.1

```
//Determines the average height of people in survey
#include <iostream>
using namespace std;

int main( )
{
  int numPeople, j;
  float oneHeight, totHeight, aveHeight;

  cout << "How many people participated in the survey? ";
  cin >> numPeople;

  if (numPeople > 0)
  {
    totHeight = 0;
    j = 1;
    while (j <= numPeople)
    {
      cout << "Enter height in metre: ";
      cin >> oneHeight;
      totHeight += oneHeight;
j++;
    }
    aveHeight = totHeight / numPeople;

    cout.setf(ios::fixed);
    cout.precision(2);
    cout << "Average height is " << aveHeight << "m" << endl;
 }
 else
    cout << "No people in the survey." << endl;
```

```
 return 0;
}

 Exercise 14.2
//Maintains a cheque account
#include <iostream>
using namespace std;

int main( )
{
  float balance, transac;

  cout << "What is the balance of the account now? R";
  cin >> balance;
  cout << endl << "Give a sequence of transactions. Press <Enter> after each."
       << endl << "Give positive values for deposits and "
       << "negative values for cheques written."
       << endl << "Enter 0 to end." << endl << endl;
  cin >> transac;
  balance += transac;

  cout.setf(ios::fixed);
  cout.precision(2);

  while (transac != 0)
  {
    cout << "Now the balance of the account is R" << balance << endl;
    cin >> transac;
    balance += transac;
  }

  return 0;
}
```

Exercise 14.3

```
//Ten in the bed
#include <iostream>
using namespace std;

int main( )
{
  int num = 10;

  while (num > 1)
  {
    cout << "There were " << num << " in the bed" << endl
         << "And the little one said:" << endl
         << "\"Roll over, roll over!\"" << endl
         << "So they all rolled over, " << endl
         << "And one fell out," << endl;
    num--;
  }
  cout << "There was 1 in the bed" << endl
       << "And the little one said:" << endl
       << "\"Good night!\"" << endl << endl;

  return 0;
}
```

Exercise 14.4

In this question we assume that at least one salary has to be input.

```
//Salaries above a given value
```

```
#include <iostream>
using namespace std;

int main( )
{
  const float COMPARE_SALARY = 100000;
  float salary, percentageAbove;
  char more;
  int numTotal = 0;
  int numAbove = 0;

  do
  {
    cout << "Please enter salary: ";
    cin >> salary;
    numTotal ++;
    if (salary > COMPARE_SALARY)
      numAbove++;
    cout << "Are there more salaries to be input (Y or N)? ";
    cin >> more;
  } while ((more == 'Y') || (more == 'y'));

  cout.setf(ios::fixed);
  cout.precision(2);
  percentageAbove = numAbove * 100.0 / numTotal;
  cout << endl << percentageAbove <<  "% of the salaries are above R"
       << COMPARE_SALARY << endl;

  return 0;
}
```

Exercise 14.5

```
The value of i is 5
```

The first declaration of `i` (namely `int i = 23;`) is overridden by the second declaration (inside the main function). This (second) declaration is valid throughout the main function, except if another variable with the same name were to be declared inside a block nested in the main function. This is what happens with the third declaration of variable `i`. It is done inside the `while` loop. All references inside the body of the loop refer to that specific i. Once the loop is exited, however, it is no longer accessible and the reference to `i` does not refer to that block variable any longer.

Thus the declaration of `i` that is valid at the point in the program where the `cout` statement is, is the second declaration of the variable, namely

```
int i = 5;
```

Its value is not changed from the initialised value of 5.

LESSON 15

Exercise 15.1

```
//Displays sum of odd numbers
#include <iostream>
using namespace std;

int main( )
{
  int n, sum;

  cout << "Up to what number do you want the sums of the odd numbers? ";
  cin >> n;
  sum = 0;
```

```
  for (int i = 1; i <= n; i+=2)   // we could also have i++ followed by
                                  // if (i%2 == 1)     in next line
  {
    sum += i;
    cout << "Sum up to " << i << " is " << sum << endl;
  }

  return 0;
}
```

Exercise 15.2

```
// Displays ASCII characters
#include <iostream>
using namespace std;

int main( )
{
  char c;

  for (int i = 32; i <= 255; i++)
  {
    c = i;
    if (i < 100)                    // to align 2 digit and 3 digit numbers
      cout << i << "  " << c << "    ";
    else
      cout << i << " " << c << "    ";
    if (i % 8 == 7)                 // display the info in 8 columns
      cout << endl;
 }

 return 0;
}
```

Exercise 15.3

```
//Displays ASCII values of upper-case characters
#include <iostream>
using namespace std;

int main( )
{
  int asciiValue;

  for (char c = 'A'; c <= 'Z'; c++)    // this is permissible
  {
    asciiValue = c;
    cout << c << " has ASCII number " << asciiValue << endl;
  }

  return 0;
}
```

Exercise 15.4

```
//Ten in the bed   (version for Lesson 15)
#include <iostream>
using namespace std;

int main( )
{
  for (int num = 10; num > 1; num--)
  {
    cout << "There were " << num << " in the bed" << endl
```

```
         << "And the little one said:" << endl
         << "\"Roll over, roll over!\"" << endl
         << "So they all rolled over, " << endl
         << "And one fell out," << endl;
 }
 cout << "There was 1 in the bed" << endl
      << "And the little one said:" << endl
      << "\"Good night!\"" << endl << endl;

 return 0;
}
```

LESSON 16

```
Exercise 16.1
//Calculates y as a function of x
#include <iostream>
using namespace std;

int main( )
{
  int startVal, endVal, x, y;

  cout << endl << endl;
  cout << "y = x*x*x - 3*x + 1" << endl << "--------------------" << endl;
  cout << endl << endl;
  cout << "Please enter first start value: ";
  cin >> startVal;
  cout << "Please enter first end value: ";
  cin >> endVal;
  cout << endl << endl;

  while ((startVal != 0) || (endVal != 0))
  {
    for (x = startVal; x <= endVal; x++)
    {
      y = (x*x - 3)*x + 1;
      cout << "x = " << x << "\t   y = " << y << endl;
    }
    cout << endl << endl;
    cout << "Please enter next start value: ";
    cin >> startVal;
    cout << "Please enter next end value: ";
    cin >> endVal;
    cout << endl << endl;
 }
  return 0;
}
```

Exercise 16.2

```
//Triangular multiplication table
#include <iostream>
#include <string>
using namespace std;

int main( )
{
  int product;

// heading
  for (int col = 1; col <= 9; col++)
    cout << '\t' << col;
```

```
  cout << endl;

 //table
 for (int row = 1; row <= 9; row++)
 {
    cout << row;
    for (int col = 1; col <= row; col++)
    {
       product = row * col;
       cout << '\t' << product;
    }
    cout << endl;
}

  return 0;
}
```

LESSON 17

```
 Exercise 17.1
//Lottery numbers
#include <iostream>
using namespace std;

int main( )
{
  int num1, num2, num3, num4, num5, num6;
  srand(time(0));

  // first random number
  num1 = rand( ) % 49  +  1;

  // second random number
  do
  {
    num2 = rand( ) % 49  +  1;
  } while (num1 == num2);

  // third random number
  do
  {
    num3 = rand( ) % 49  +  1;
  } while ((num1 == num3) || (num2 == num3));

  // fourth random number
  do
  {
    num4 = rand( ) % 49  +  1;
  } while ((num1 == num4) || (num2 == num4) || (num3 == num4));

 // fifth random number
 do
 {
   num5 = rand( ) % 49  +  1;
 } while ((num1 == num5) || (num2 == num5) || (num3 == num5)
                         || (num4 == num5));

 // sixth random number
 do
 {
   num6 = rand( ) % 49  +  1;
 } while ((num1 == num6) || (num2 == num6) || (num3 == num6)
                   || (num4 == num6) || (num5 == num6));

 // output
```

```
 cout << "The 6 random numbers are:" << endl;
 cout << ' ' << num1 << ' ' << num2 << ' ' << num3
      << ' ' << num4 << ' ' << num5 << ' ' << num6 << endl << endl;

 return 0;
}
```

Exercise 17.2

(i)
```
y = (fabs(x)) * 2
  = (fabs(-6.32) * 2
  =   6.32       * 2
  = 12.64
```

(ii)
```
y = (2 * (pow(x,2))) - 1
  = (2 * (pow(2,2))) - 1
  =  2 *     4       - 1
  =     8            - 1
  =     7
z = (2 * (pow(x-1,2)))
  = 2 *   pow(1,2)
  = 2 *     1
  = 2
```

(iii)
```
y = sqrt(fabs(x))
  = sqrt(fabs(-4))
  = sqrt(4)
  = 2
```

(iv)
```
z = sqrt(pow(x,2) + pow(y,2))
  = sqrt(pow(3,2) + pow(4,2))
  = sqrt(9 + 16)
  = sqrt(25)
  = 5
```

Exercise 17.3

```
//Pythagoras
#include <iostream>
#include <cmath>
using namespace std;

int main( )
{
  float a,b,c;
  cout << "Please enter the values of the ";
  cout << "two shorter sides of the triangle: ";
  cin >> b >> c;
  a = sqrt(pow(b, 2) + pow(c, 2));
  cout.setf(ios::fixed);
  cout.precision(2);
  cout << "The length of the longest side is " << a << endl;

  return 0;
}
```

Exercise 17.4

```
// Investigates the function toupper
#include <iostream>
using namespace std;

int main( )
{
```

```
  char c, cConverted;

  for (int i = 1; i <= 5; i++)        //we test 5 cases at a time
  {
    cout << "Please enter any character: ";
    cin >> c;
    cConverted = toupper(c);
    cout << "The function toupper changes it to " << cConverted << endl;
  }

  return 0;
}
```

The function toupper changes lower case letters of the alphabet to the corresponding upper case letters. It does not change upper case letters, digits or punctuation characters.

Exercise 17.5

```
//Shakespeare
#include <iostream>
using namespace std;

int main( )
{
  const int NUMBER_OF_WORDS = 15;       //number of words in a 'sentence'
  char c;
  int numberOfLetters;

  srand(time(0));

  // loop to construct one sentence
  for (int i = 1; i <= NUMBER_OF_WORDS; i++)
  {
    // number of letters in i-th word
    numberOfLetters = rand( ) % 27  +  1;

    // loop to construct i-th word
    for (int j = 1; j <= numberOfLetters; j++)
    {
      // next character is generated and displayed
      c = rand( ) % 26  +  65;
      cout << c;
    }                  // the i-th word has now been displayed

    // blank between words is displayed
    if (i < NUMBER_OF_WORDS)
      cout << ' ';
  }                    // all words of the sentence have now been displayed
  // full stop at end of sentence
  cout << '.' << endl << endl;

  return 0;
}
```

LESSON 18

Exercise 18.1

Concept	Line number(s)	Notes
Function heading	5	
Formal parameters	5	They are `par1`, `par2` and `par3`.
Return type	5	The return type is `bool`

Function call	14	The function is called inside the condition of the `if` statement
Actual parameters	14	They are `var1`, `var2` and `var3`.
First line	10	Execution starts at the `main` function

Exercise 18.2

```
//Checks whether three numbers represent the sides of a triangle
#include <iostream>
using namespace std;

// test whether a+b > c
bool isGreater(float a, float b, float c)
{
  return ((a + b) > c);
}

// test whether 3 numbers represent the sides of a triangle
// (sum of any 2 sides should be greater than the other side)
bool isTriangle(float par1, float par2, float par3)
{
  return (isGreater(par1, par2, par3) && isGreater(par1, par3, par2)
                                && isGreater(par2, par3, par1));
}

int main( )
{
  float var1, var2, var3;

  cout << "Please enter three numbers: ";
  cin >> var1 >> var2 >> var3;

  if (isTriangle(var1, var2, var3))
    cout << "The 3 numbers represent the 3 sides of a triangle." << endl;
  else
    cout << "The 3 numbers do not represent the 3 sides of a triangle."
         << endl;

  return 0;
}
```

Exercise 18.3

```
//Chirping cricket and the temperature
#include <iostream>
using namespace std;

// formula
float approximateTemp(float numberP)
{
  return (numberP + 160)/4;
}

//conversion from Fahrenheit to Celsius
float tempCelsius(float degreeP)
{
  return (degreeP -  32) * 5 / 9;
}

int main( )
{
  float number, fahrTemperature, celsiusTemperature;
```

```
  cout << "Please enter the number of cricket chirps per minute: ";
  cin >> number;

  fahrTemperature = approximateTemp(number);
  celsiusTemperature = tempCelsius(fahrTemperature);

  cout.setf(ios::fixed);
  cout.precision(2);
  cout << "The approximate temperature is ";
  cout << fahrTemperature << " degrees F, or ";
  cout << celsiusTemperature << " degrees C" << endl;

  return 0;
}
```

Exercise 18.4

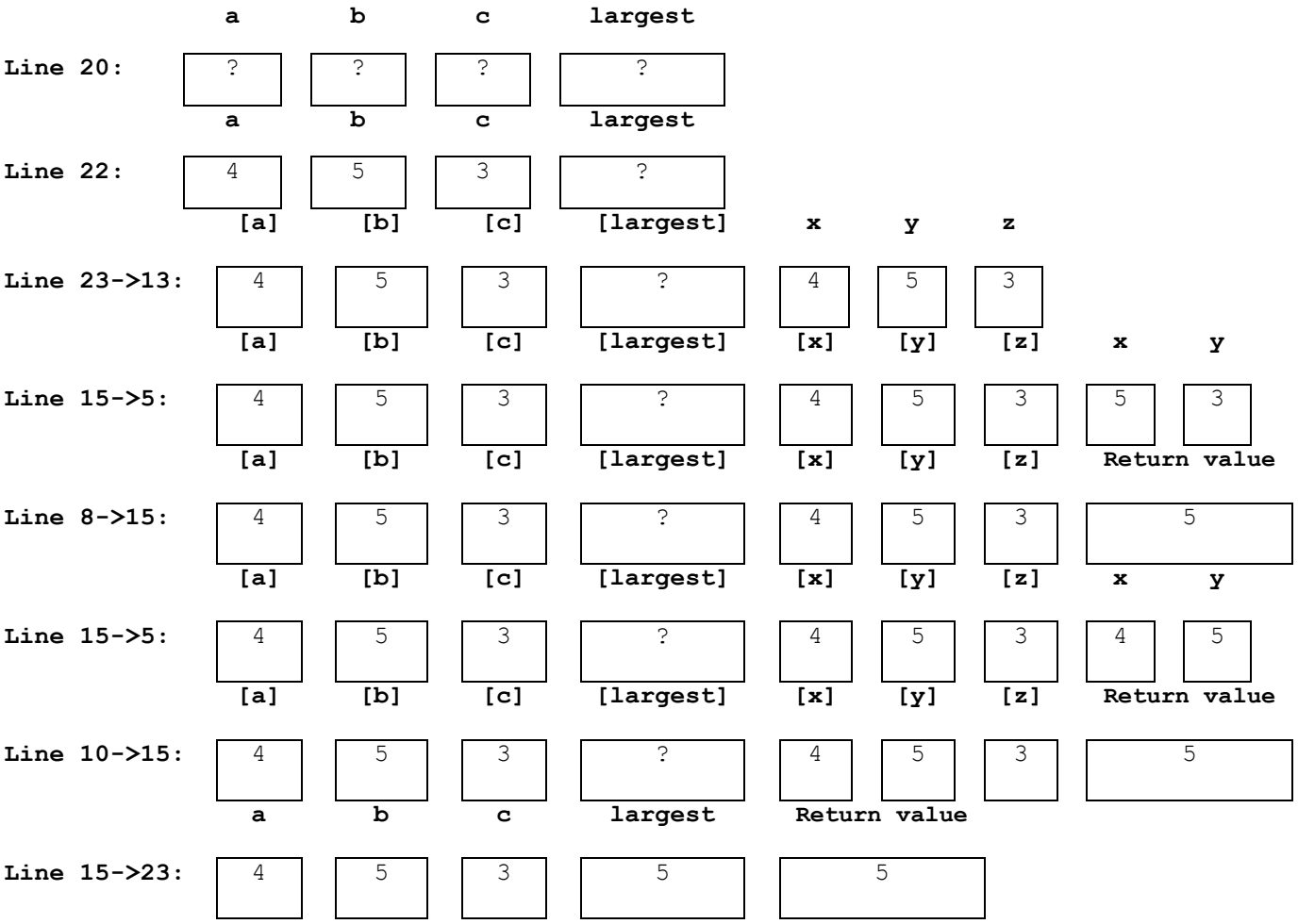
```
//Area of a triangle
#include <iostream>
#include <cmath>
using namespace std;

float area(float p1, float p2, float p3)
{
  float s;
  s = 0.5 * (p1 + p2 + p3);

  return sqrt(s * (s - p1) * (s - p2) * (s - p3));
}

int main( )
{
  float a, b, c, areaTriangle;

  cout << "Please enter the lengths of the 3 sides of the triangle: ";
  cin >> a >> b >> c;

  areaTriangle = area(a, b, c);

  cout << "The area of the triangle is " << areaTriangle << endl;

  return 0;
}
```

Exercise 18.5

```
Here is the program:
1     //Determine the maximum of three values
2     #include <iostream>
3     using namespace std;
4
5     float max2(float x, float y)
6     {
7        if (x > y)
8           return x;
9        else
10          return y;
11    }
12
13    float max3(float x, float y, float z)
14    {
15       return max2(x, max2(y, z));
16    }
17
```

```
18    int main( )
19    {
20       float a, b, c, largest;
21       cout << "Enter three numbers: ";
22       cin >> a >> b >> c;
23       largest = max3(a, b, c);
24       cout << "The maximum is: " << largest << endl;
25       return 0;
25    }
```

For the purposes of clarity, we included a separate box in the solution below in those variable diagrams which depict a function returning a value.



LESSON 19

Exercise 19.1

```
//Displays all powers of x (=2) less than a limit
#include <iostream>
using namespace std;

// find xP to the power of nP
float iPow(int xP, int nP)
{
  float answer = 1;
  for (int i = 1; i <= nP; i++)
     answer *= xP;

  return answer;
}
```

```
int main( )
{
  float limit, result;
  int n, x;
  cout << "Enter a limit for the calculations: ";
  cin >> limit;

  x = 2;
  n = 1;

  result = iPow(x, n);
  while (result < limit)
  {
    cout << x << " to the power " << n << " = " << result << endl;
    n++;
    result = iPow(x, n);
  }

  return 0;
}
```

Exercise 19.2

- Variable `n` is declared at the beginning of the program outside the declaration of all functions and is accessible throughout the program, i.e. throughout all three functions. It is a global variable.
- Variable `answer` is declared at the beginning of the program and it is natural to think that it will be accessible throughout the program, i.e. in all three functions. This is, however, **not** the case. In function product we have another declaration of `answer`. This `answer` (declared in function `product`) is accessible in `product` only (lines 16 to 20). It is actually a local variable of the function. All references to answer inside `product`, will be to this local variable. The `answer` defined at the beginning of the program will be accessible in functions `sum` and `main`. All references to `answer` inside `sum` and `main`, will be to this global variable.
- In both function `sum` and function `product`, a variable `i` is declared. Both declarations are done inside for loops. In both cases the variable will only be accessible inside the relevant for loop. Hence, in function `sum`, the (local) variable `i` will be accessible in lines 9 and 10, and in function `product`, the (local) variable `i` will be accessible in lines 17 and 18.

Exercise 19.3

```
//Determines the sum and product of 1 to n (Version 2)
#include <iostream>
using namespace std;

// sum from 1 up to n
int sum(int n)
{
  int s = 0;
  for (int i = 1; i <= n; i++)
    s += i;

  return s;
}

// product from 1 up to n
int product(int n)
{
  int p = 1;
  for (int i = 2; i <= n; i++)
    p *= i;

  return p;
}
```

```
int main( )
{
  int upperLimit, answerSum, answerProd;

  cout << "Enter a positive integer: ";
  cin >> upperLimit;

  answerSum = sum(upperLimit);
  cout << "The sum of 1 up to " << upperLimit << " is "
       << answerSum << endl;

  answerProd = product(upperLimit);
  cout << "The product of 1 up to " << upperLimit << " is "
       << answerProd << endl;

 return 0;
}
```

LESSON 20

Exercise 20.1

```
//Draws a frame of asterisks
#include <iostream>
using namespace std;

void displayFrame(int w, int h)
{
  cout << endl;
  for (int i = 1; i <= h; i++)            // rows
  {
     for (int j = 1; j<= w; j++)          // columns
    {
      if ((i == 1) || ( i == h))          // top and bottom
        cout << "* ";
      else if ((j == 1) || (j == w))      // left and right
        cout << "* ";
      else
        cout << "  ";               // "empty" body
    }
    cout << endl;
  }
}

int main( )
{
  int width, height;

  cout << "Please enter the width and height of the rectangle: ";
  cin >> width >> height;
  displayFrame(width, height);

  return 0;
}
```

Exercise 20.2

```
//Draws a tree
#include <iostream>
using namespace std;

//  'spacesP' indicates the number of space characters to display and
//  'asterisksP' indicates the number of asterisks in the line
```

```
void displayOneLine(int spacesP, int asterisksP)
{
  for (int i = 1; i <= spacesP; i++)
    cout << " ";
  for (int i = 1; i <= asterisksP; i++)
  cout << "* ";
  cout << endl;
}

int main( )
{
  int size;

  cout << "Please enter the size of the tree: ";
  cin >> size;
  cout << endl;

// display the tree top
  for (int i = 1; i <= size; i++)
    displayOneLine(size - i,i);

// display the trunk
  displayOneLine(size - 1,1);
  displayOneLine(size - 1,1);

  return 0;
}
```

Exercise 20.3

```
//Displays a histogram for a series of values
#include <iostream>
using namespace std;

void displayRow(int n, char c)
{
  for (int i = 1; i <= n; i++)
    cout << c;
  cout << endl;
}

int main( )
{
  int value, total, many, averageInt;
  float average;
    total = 0;
    many = 0;

    cout << "Enter the values (negative to end):" << endl;
    cin >> value;

    while (value >= 0)
    {
       if (value > 80)
         do
         {
           cout << "Value cannot be more than 80. Enter again: ";
           cin >> value;
         } while (value > 80);

      total += value;
      many++;
      displayRow(value, '*');
      cin >> value;
    }
```

```
    if (many > 0)
    {
      average = float(total)/many;
      cout << "The average is " << average << endl;
      averageInt = int(average + 0.5);
      displayRow(averageInt, '+');
    }
    else
      cout << "There were no values entered." << endl;

 return 0;
}
```

LESSON 21

Exercise 21.1

```
// Simulates throwing dice until a total of 7 is thrown
#include <iostream>
using namespace std;

void calcTotalDice(int & totalP)
{
  int die1 = rand( )%6 + 1;
  int die2 = rand( )%6 + 1;
  cout << "Throw: " << die1 << " and " << die2 << endl;
  totalP = die1 + die2;
}

int main( )
{
  int count, total;
  srand(time(0));

  calcTotalDice(total);
  count = 1;
  while (total != 7)
  {
   calcTotalDice(total);
   count++;
  }
  cout << "It took " << count << " throws ";
  cout << "before 7 was thrown." << endl;

  return 0;
}

Exercise 21.2
//Display a histogram for a series of values (version of lesson 21)
#include <iostream>
using namespace std;

void inputAndValidate(int & valueP)
{
 cin >> valueP;
 if (valueP > 80)
     do
     {
       cout << "Value cannot be more than 80. Enter again: ";
       cin >> valueP;
     } while (valueP > 80);
 }
```

```
void displayRow(int n, char c)
{
  for (int i = 1; i <= n; i++)
      cout << c;
  cout << endl;
  }

int main( )
{
  int value, total, many, averageInt;
  float average;

  total = 0;
  many = 0;

  cout << "Enter the values (negative to end):" << endl;
  inputAndValidate(value);             // first value is input and validated

  while (value >= 0)
  {
    total += value;
    many++;
    displayRow(value, '*');
    inputAndValidate(value);          // next value is input and validated
  }

  if (many > 0)
  {
    average = float(total)/many;
    cout << "The average is " << average << endl;
    averageInt = int(average + 0.5);
    displayRow(averageInt, '+');
  }
  else
    cout << "There were no values entered." << endl;

 return 0;
}
```

Exercise 21.3

```
//Area and circumference of a circle
#include <iostream>
using namespace std;

const float PI = 3.142857;

// finds area and circumference of a circle
void findAreaAndCircum(float radiusP, float & areaP, float & circumP)
{
  areaP = PI * radiusP * radiusP;
  circumP = 2 * PI * radiusP;
}

int main( )
{
  float radius, area, circumference;

    for (int i = 1; i <= 4; i++)
    {
       cout << endl << "Please enter the radius of the circle: ";
       cin >> radius;
       findAreaAndCircum(radius, area, circumference);
       cout << "The area of the circle is " << area << endl;
```

```
        cout << "The circumference of the circle is " << circumference << endl;
    }

    return 0;
}
```

```
 Exercise 21.4
//Area and circumference of a circle  (version 2)
#include <iostream>
using namespace std;

const float PI = 3.142857;

// finds area of a circle
float areaCircle(float radiusP)
{
  return (PI * radiusP * radiusP);
}

// finds circumference of a circle
float circumCircle(float radiusP)
{
  return (2 * PI * radiusP);
}

int main( )
{
  float radius, area, circumference;

  for (int i = 1; i <= 4; i++)
  {
    cout << endl << "Please enter the radius of the circle: ";
    cin >> radius;

    area = areaCircle(radius);
    circumference = circumCircle(radius);
    cout << "The area of the circle is " << area << endl;
    cout << "The circumference of the circle is " << circumference << endl;
 }

  return 0;
}
```

LESSON 22

Exercise 22.1

```
//Increments a number with 15
#include <iostream>
using namespace std;

//This function adds 15 to an input value.
void increment15(int & n)
{
  n += 15;
}

int main( )
{
  int i;

  cout << "Enter an integer value: ";
  cin >> i;
  increment15(i);
```

```
  cout << "New value: " << i << endl;

  return 0;
}
```

Exercise 22.2

```
//Two values are swapped
#include <iostream>
using namespace std;

// The function makes use of a temporary variable to swap the values of
// two variables.
void swap(int & n1, int & n2)
{
  int temp;
  temp = n1;
  n1 = n2;                  // n1 now contains the original value of n2
  n2 = temp;                // n2 now contains the original value of n1
}

int main( )
{
  int first, second;

  cout << "Please enter the two values to be swapped: ";
  cin >> first >> second;
  swap(first, second);
  cout << " The swapped values are " << first << " and " << second << endl;
  swap(first, second);
 cout << " and, when swapped again, the values are " << first
      << " and " << second << endl;

  return 0;
}
```

Exercise 22.3

```
//Three values are rotated
#include <iostream>
using namespace std;

// n1, n2 and n3 are rotated
void rotate(int & n1, int & n2, int & n3)
{
  int temp;
  temp = n1;
  n1 = n2;              // n1 now contains the original value of n2
  n2 = n3;              // n2 now contains the original value of n3
  n3 = temp;            // n3 now contains the original value of n1
}

int main( )
{
  int first, second, third;

  cout << "Please enter the three values to be rotated: ";
  cin >> first >> second >> third;

//loop to rotate the values three times
  for (int i = 1; i <= 3; i++)
  {
    rotate(first, second, third);
    cout << " After rotation #" << i << ": ";
    cout << first << "  " << second << "  " << third << endl;
  }
```

```
  return 0;
}
```

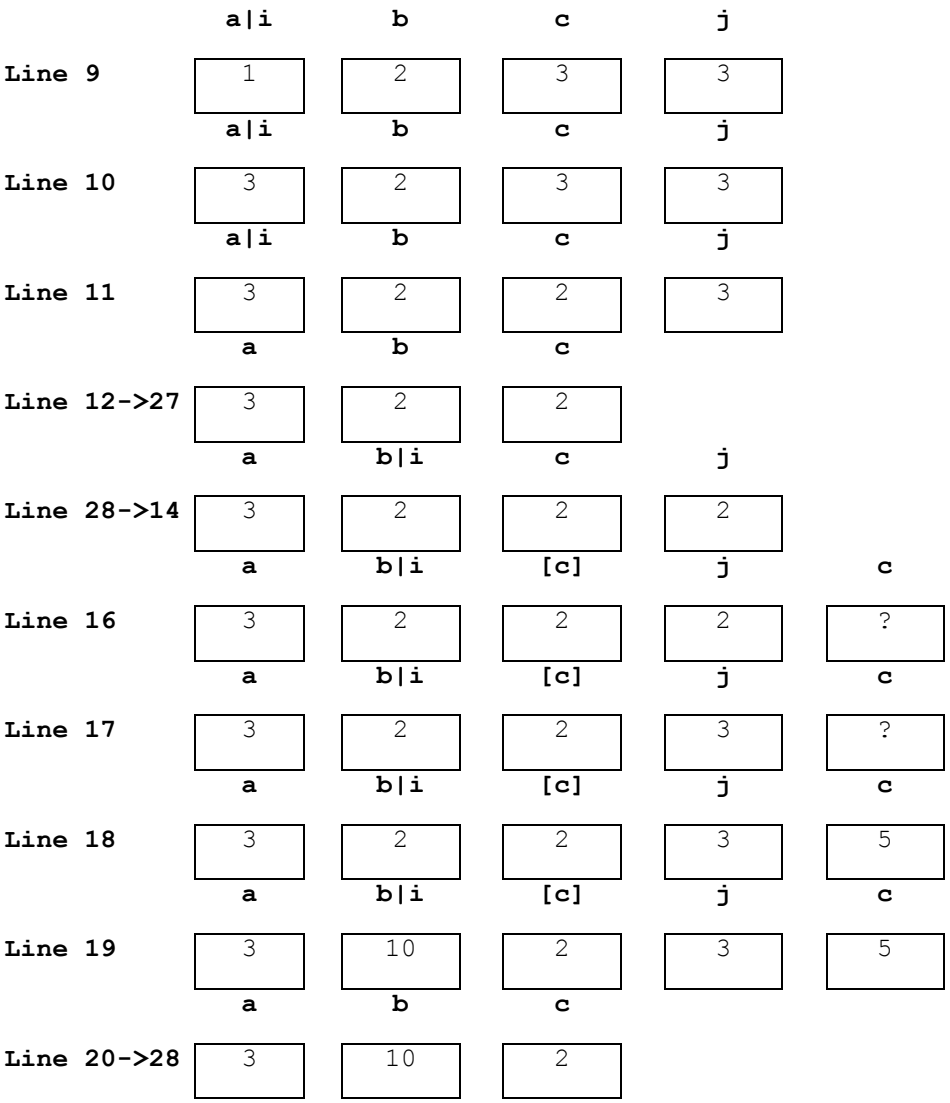
LESSON 23

Exercise 23.1

	a	b	c	x	z
Line 14	6				
Line 15	6	7			
Line 16	6	7	8		
Line 17->5	6	7	8	6	8
Line 7	6	7	8	16	8
Line 8	6	17	8	16	8
Line 9	6	17	8	16	18
Line 10->17	6	17	8		

Exercise 23.2

	a	b	c	j
Line 5	?	?	?	
Line 24	1	?	?	
Line 25	1	2	?	
Line 26	1	2	3	
Line 27->7	1	2	3	2



This program makes use of *global* variables. We hope, however, that after having worked through this exercise, you will be too scared ever to use global variables! The variables `a`, `b` and `c` are declared as global variables in line 5 and are given values in lines 24 to 26. When function `funcP` is called in line 27, `a` is mapped onto `i` and `b` onto `j`.

Because `i` is a reference parameter, we normally interpret it as a temporary name for variable `a`. Unfortunately it is not so simple. *Both* variable `a` and variable `i` may be used in the function! (The reason is, of course, that `a` is defined globally.) Any changes to `i` in function `funcP` will be reflected in the value of `a` (which is no surprise) and any changes to `a` in function `funcP` *will be reflected in the value of `a` and also in the value of `i`* (this may be surprising). Suppose we add the following statements to those of *`funcP`* between lines 8 and 9:

```
a += 2;

i += 3;
```

After the first of these two statements have been executed, the value of *both* `a` and `i` would be 3, and after the second of these two statements have been executed, the value of *both* `a` and `i` would be 6.

Because `j` is a value parameter, we would expect to put square brackets around `b` in the variable diagram to indicate that the variable is unaccessible. That is, however, not the case. *Both* variable `b` and variable `j` may be used in the function! (The reason is, of course, that `b` is defined globally.) We do *not* put any brackets around it in the diagram, because `b` does not have anything to do with `j`. Any changes to `j` will not be reflected in the value of `b` (no surprise) but we may refer to `b` inside function `funcP` and change its value if we like (this may be surprising). Suppose we add the following statements to those of `funcP` between lines 8 and 9:

```
b += 5;
```

```
j += 2;
```

After the first of these two statements have been executed, the value of `b` would be 7 and the value of `j` would still be 2, and after the second of these two statements have been executed, the value of `b` would (still) be 7 and the value of `j` would be 4.

Variable `c` does not play any role in the call to function `funcP`. However, because it is a global variable, it is accessible inside the function - we may change its value inside function `funcP`. No brackets are used in the variable diagram.

When control jumps back from function `funcP` to function `main` (12 → 27), the values of `a`, `b` and `c` are, respectively, 3, 2 and 2. Now function `funcQ` is called. Here `b` is mapped onto `i` and `c` onto `j`.

Because `i` is a reference parameter, we normally interpret it as a temporary name for `b`. Unfortunately it is not so simple. *Both* variable `b` and variable `i` may be used in the function. (The reason is, once again, that `b` is defined globally.) Any changes to `i` in function `funcQ` will be reflected in the value of `b` (which is no surprise) and any changes to `b` in function `funcP` *will be reflected in the value of* `b` *and also in the value of* `i` (this may be surprising). Suppose we add the following statements to those of `funcQ` between lines 15 and 16:

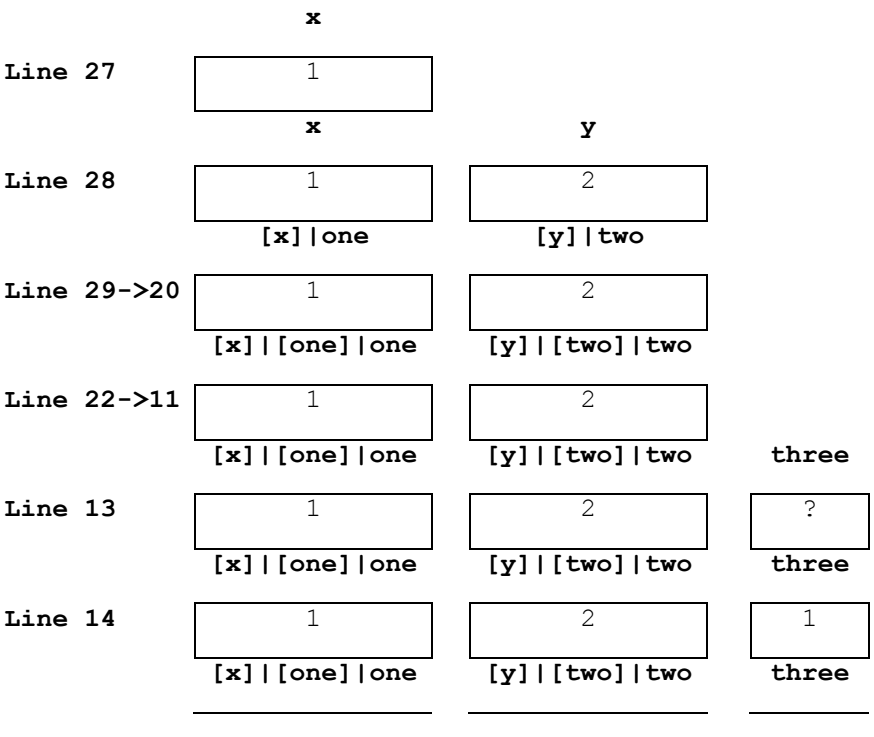
```
b += 2;

i += 3;
```

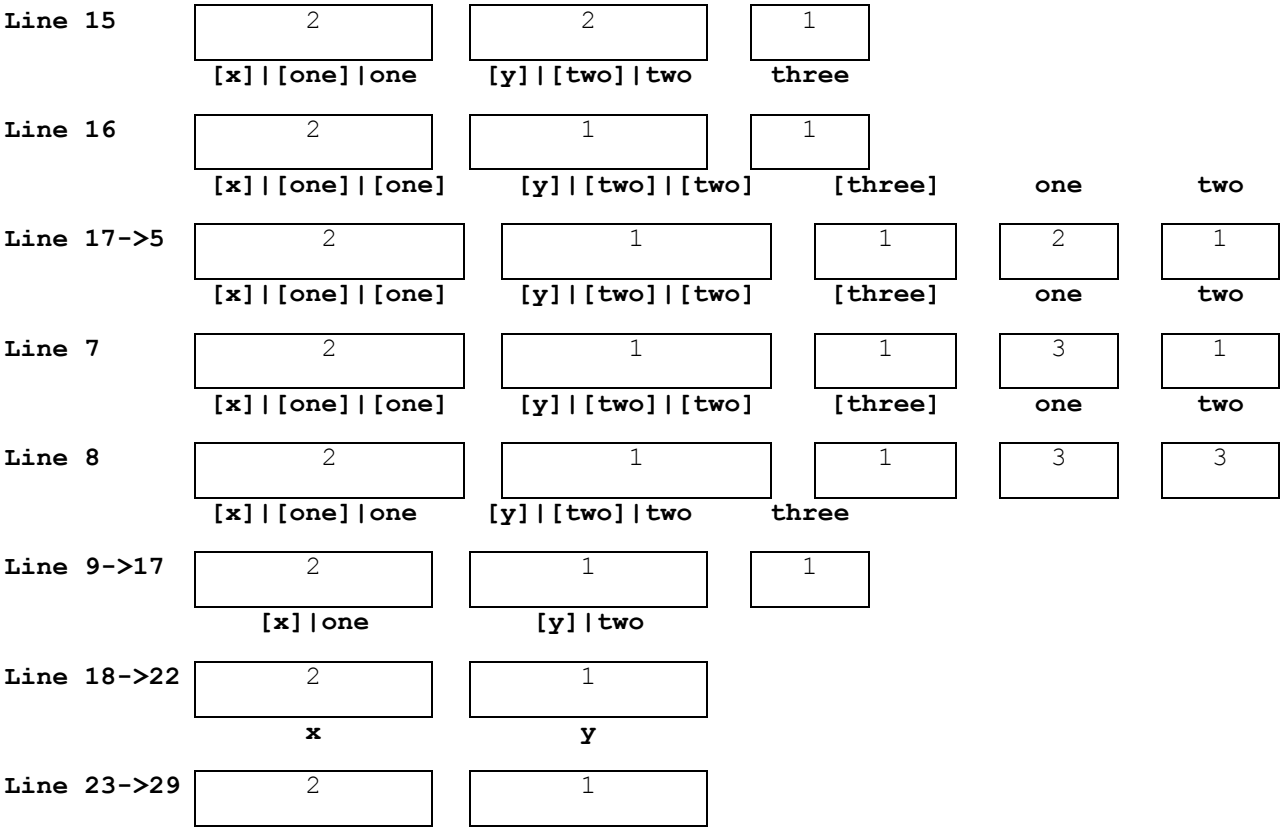
After the first of these two statements have been executed, the value of *both* `b` and `i` would be 4, and after the second of these two statements have been executed, the value of *both* `b` and `i` would be 7.

Because `j` is a value parameter, we would expect to put square brackets around `c` in the variable diagram to indicate that the variable is unaccessible.It is, however, not the case. When the function is entered, the globally defined variable `c` is still accessible. Then, in line 16 a variable `c` is declared as a local variable inside function `funcQ` and this overrides the global declaration. The global `c` is now inaccessible in this function and *that* is the reason for the square brackets around the globally defined `c` in the variable diagram of line 13. When `c` is altered in the function, it has no influence on either `j` or the globally defined `c`.

So the moral of the story is: **Avoid the use of global variables.**

Exercise 23.3





In this exercise we have nested function calls. Look at how the temporary names of reference parameters are indicated in the diagrams – we put square brackets around the names that are temporarily replaced. When the function `funcC` is executed, we are dealing with value parameters. The names of all the other variables are inaccessible and we make new copies of one and two.

The moral of this exercise is that you should use different names for the actual and formal parameters of a function.

LESSON 24

Exercise 24.1

```
//Swaps largest element with first element
#include <iostream>
using namespace std;
int main( )
{
  const int N = 10;          // number of elements in the array
  int max, index;
  int numbers[] = {10, 3, 56, 7, 0, 5, 44, 99, 76, 1};
// Find maximum element and subscript of maximum element
  max = numbers[0];
  index = 0;
  for (int j = 1; j < N; j++)
  {
    if (numbers[j] > max)
    {
      max = numbers[j];
      index = j;
    }
  }
// Swop
  swap(numbers[index], numbers[0]);
//   Note: Instead of the above statement,
//   we could use the following two statements:
//   numbers[index] = numbers[0];
//   numbers[0] = max;
```

```
// Output
  cout << "Changed array:" << endl;
  for (int j = 0; j < N; j++)
    cout << numbers[j] << " ";
  cout << endl;

return 0;
}
```

```
 Exercise 24.2
//Displays the elements of an array in reverse order
#include <iostream>
using namespace std;

main( )
{
  const int N = 10;       // number of values in the array
  float numbers[N];

// Input
  cout << "Enter " << N << " floating point numbers separated by spaces:"
       << endl;
  for (int j = 0; j < N; j++)
    cin >> numbers[j];

// Display in reversed order
  cout << endl << "Reversed array:" << endl;
  for (int j = N - 1; j >= 0; j--)
    cout << numbers[j] << " ";
  cout << endl;

  return 0;
}
```

Exercise 24.3

```
//Checks whether the elements in an array are in ascending order
#include <iostream>
using namespace std;

main( )
{
  bool ascending;
  const int N = 20;          //equal to the number of elements in the array
  int numbers[] = {12, 15, 17, 23, 37, 40, 55, 54, 70, 77,
              79, 80, 84, 86, 89, 91, 92, 100, 123, 126};
  int index;

 // Check whether in ascending order
  ascending = true;
  index = 0;
  while (ascending && (index < N - 1))
  {
    index++;
    if (numbers[index] < numbers[index - 1])
      ascending = false;
  }

// Output
  if (ascending)
    cout << "Yes, the array is in ascending order." << endl;
  else
  {
    cout << "No, the array is not in ascending order. " << endl;
```

```
    cout << "The value in position " << index  << ", namely "
         << numbers[index] << ", is the first value out of order." << endl;
  }

  return 0;
}
```

Exercise 24.4

```
//Assigns the values of one array to another
#include <iostream>
using namespace std;

main( )
{
  const int N = 10;               // size of the two arrays
  float array1[N], array2[N];

// Input
  cout << "Enter " << N << " floating point numbers separated by spaces:"
       << endl;
  for (int j = 0; j < N; j++)
    cin >> array1[j];

// Assign to second array
  for (int j = 0; j < N; j++)
    array2[j] = array1[j];

 cout << "The corresponding values of array2 and array1 are now the same."
      << endl;

 return 0;
}
```

Exercise 24.5

```
//Sorts an array of 15 random integers  (version 2)
#include <iostream>
using namespace std;

int main( )
{
  const int NUM_VALS = 15;       // number of values to be generated and sorted
  int values[NUM_VALS];
  int nextVal, current;

  srand(time(0));

  for (int i = 0; i < NUM_VALS; i++)
  {
    nextVal = rand( ) % 1000;       // generates 'new value'
    current = i - 1;                // index of last value already sorted

    // compares 'new value' with all values that are already sorted,
    // starting at the last value so far and
    // shifting every value larger than 'new value' one position to the right
    while ((current >= 0) && (values[current] > nextVal))
    {
      values[current+1] = values[current];
      current--;
    }
    values[current + 1] = nextVal;      // inserts 'new value' at correct
                                        // position

    cout << "After the " << i+1 << "th value has been generated, "
```

```
         << "the sorted array looks as follows:" << endl;
    for (int j = 0; j <= i; j++)
      cout << values[j] << ' ';
    cout << endl;
  }

  cout << endl << endl << "The final sorted array:" << endl;
  for (int i = 0; i < NUM_VALS; i++)
    cout << values[i] << "  ";
  cout << endl << endl;

  return 0;
}
```

LESSON 25

Exercise 25.1

```
//Various functions for arrays:  exercise 1
#include <iostream>
using namespace std;

const int N = 10; // size of the array
// (alternatively a maximum size should appear in the declaration,
// then the actual size has to be input in the main function and
// then passed as an additional parameter to the called functions)
// input the values for the elements of the array
void inputArray(int a[])
{
  cout << "Please enter " << N << " integers:" << endl;
  for (int i = 0; i < N; i++ )
    cin >> a[i];
}
// determine whether x appears in the array
bool inArray(const int a[], int x)
{
 for (int i = 0; i < N; i++)
   if (x == a[i])
     return true;

 return false;
}

// find smallest element in the array
int smallest(const int a[])
{
  int s = a[0];
  for (int i = 1; i < N; i++)
    if (a[i] < s)
       s = a[i];
  return s;
}
// reverse the array
void reverseArray(int a[])
{
 for (int i = 0; i < N/2; i++)
   swap(a[i], a[N-1-i]);
}

// output
void outputArray(const int a[])
{
  for (int i = 0; i < N; i++)
    cout << a[i] << "  ";
```

```
  cout << endl;
}

int main( )
{
  int array[N];                    // N was declared as a global constant
  int number, smallestElement;

  inputArray(array);

  cout << endl << "Please enter an integer: ";
  cin >> number;
  if (inArray(array, number))
    cout << "Yes, " << number << " appears in the array." << endl;
  else
    cout << "No, " << number << " does not appear in the array." << endl;

  smallestElement = smallest(array);
  cout << endl << "Smallest element in the array: "
       << smallestElement << endl;

  reverseArray(array);
  cout << endl << "Reversed array is:" << endl;
  outputArray(array);
  return 0;
}
```

Exercise 25.2

Run the program if you want to get the answer to the question. 1 indicates open and 0 indicates closed.

```
//Post boxes managed by Peter
#include <iostream>
using namespace std;

const int NUM_OF_BOXES = 50;

// investigate every j-th post box: close it if open, open it if closed
void openAndClose(bool a[], int j)
{
 for (int i = j; i <= NUM_OF_BOXES; i += j)
   a[i-1] = ! a[i-1];

 // display status at this point: 1 indicates open and 0 indicates closed
 cout << endl << j << ":\t ";
 for (int i = 0; i < NUM_OF_BOXES; i++)
   cout << a[i];
 }

int main( )
{
  bool postBox[NUM_OF_BOXES];    // true if box is open, false if closed

// at the beginning all the boxes are closed
  for (int i = 0; i < NUM_OF_BOXES; i++)
    postBox[i] = false;

// step size: 2, 3, 4, ..., NUM_OF_BOXES
  for (int j = 2; j <= NUM_OF_BOXES; j++)
    openAndClose(postBox, j);
  cout << endl;

  return 0;
}
```

Exercise 25.3

```
//Calculates the mark of a student
#include <iostream>
using namespace std;

const int NUM_QUESTIONS = 10;     // size of the array
// (alternatively a maximum size should appear in the declaration,
// then the actual size has to be input in the main function and
// then passed as an additional parameter to the called functions)
// input and validate answers (T or F) and store them in an array
void inputAndValidateAnswers(char a[])
{
 // input
 cout << "Please enter " << NUM_QUESTIONS << " answers (T or F): ";
 for (int i = 0; i < NUM_QUESTIONS; i++)
    cin >> a[i];

 // validate
 for (int i = 0; i < NUM_QUESTIONS; i++)
    while ((a[i] != 'T') && (a[i] != 'F'))
    {
       cout << endl << "Answer " << i+1 << " is not valid. Type it again: ";
       cin >> a[i];
    }
 }
// calculate mark
int mark(const char correctAnswersP[], const char studentAnswersP[])
{
 int m = 0;
 for (int i = 0; i < NUM_QUESTIONS; i++)
    if (correctAnswersP[i] == studentAnswersP[i])
      m++;

    return m;
}

int main( )
{
 int result;
 char correctAnswers[NUM_QUESTIONS];
 char studentAnswers[NUM_QUESTIONS];

 cout << endl << "CORRECT ANSWERS" << endl;
 inputAndValidateAnswers(correctAnswers);

 cout << endl << "ANSWERS of STUDENT" << endl;
 inputAndValidateAnswers(studentAnswers);

 result = mark(correctAnswers, studentAnswers);
 cout << endl << "The mark out of " << NUM_QUESTIONS << " is " << result;
 cout << endl;

 return 0;
}
```

LESSON 26

Exercise 26.1

```
//Determines whether one matrix is the transposition of another
#include <iostream>
using namespace std;
```

```
const int N = 10;    // size of two N x N matrices

// input values
void inputDataIntoSquareMatrix(int a[][N])
{
for (int i = 0; i < N; i++)
{
   cout << "Enter " << N << " values for row " << i << ": ";
   for (int j = 0; j < N; j++)
      cin >> a[i][j];
   cout << endl;
   }
}
// determine whether matrix a is the transposition of matrix b
bool transposition(const int a[][N], const int b[][N])
{
 for (int i = 0; i < N; i++)
   for (int j = 0; j < N; j++)
     if (a[i][j] != b[j][i])
       return false;

   return true;
  }

int main( )
{
  int matrixA[N][N], matrixB[N][N];

// input
  cout << "FIRST MATRIX" << endl << endl;
  inputDataIntoSquareMatrix(matrixA);

  cout << endl << "SECOND MATRIX" << endl << endl;
  inputDataIntoSquareMatrix(matrixB);

// check whether they are the transposition of each other
 if (transposition(matrixA, matrixB))
    cout << endl << "Yes, the matrices are the transposition of each other";
 else
    cout << endl << "No, the matrices are not the transposition"
         << " of each other";
 cout << endl << endl;

 return 0;
}
```

Exercise 26.2

```
//Rainfall data over several years
#include <iostream>
using namespace std;

const int NUM_YEARS = 6;          // data spans so many years
const int FIRST_YEAR = 2000;      // first year for which data exist
const int NUM_MONTHS = 12;        // months in each year

// input rain data and store it in a 2-dim array
void inputRainData(float r[][NUM_MONTHS])
{
  for (int i = 0; i < NUM_YEARS; i++)
  {
    cout << "Enter " << NUM_MONTHS << " values for year "
         << FIRST_YEAR + i << ": ";
    for (int j = 0; j < NUM_MONTHS; j++)
      cin >> r[i][j];
```

```
    cout << endl;
  }
}

// find total rainfall during all these years
float totalRain(const float r[][NUM_MONTHS])
{
 float sum = 0;
 for (int i = 0; i < NUM_YEARS; i++)
   for (int j = 0; j < NUM_MONTHS; j++)
     sum += r[i][j];

 return sum;
}
// determine month and year with highest rainfall (first time)
// as well as number of times that this occurred
void getHighest(const float r[][NUM_MONTHS], float & highP,
                          int & yearP, int & monthP, int & nrTimesP)
{
// initialise
  highP = -1;
  yearP = -1;
  monthP = -1;
  nrTimesP = 0;

 // loop over all the data
 for (int i = 0; i < NUM_YEARS; i++)
    for (int j = 0; j < NUM_MONTHS; j++)
      if (r[i][j] > highP)
      {
 highP = r[i][j];   // maximum up to now
 yearP = i;         // first year when this occurred
 monthP = j;        // first month when this occurred
 nrTimesP = 1;      // first time that this occurred
      }
      else if (r[i][j] == highP)
    nrTimesP++;       // this maximum has occurred previously already
}

int main( )
{
  float rainData[NUM_YEARS][NUM_MONTHS];
  float total, annualAverage, monthlyAverage, maxRainfall;
  int yearMax, monthMax, nrTimes;

  cout.setf(ios::fixed);
  cout.precision(2);

// input
  inputRainData(rainData);

// total rainfall
  total = totalRain(rainData);

// find and display annual average
  annualAverage = total / NUM_YEARS;
  cout << endl << "Average annual rainfall: " << annualAverage << "mm." << endl;

// find and display monthly average
  monthlyAverage = total / (NUM_YEARS * NUM_MONTHS);
  cout << "Average monthly rainfall: " << monthlyAverage << "mm." << endl;

// determine when and how many times highest monthly rainfall occurred
  getHighest(rainData, maxRainfall, yearMax, monthMax, nrTimes);
```

```
// display result
  cout << endl << "The maximum rainfall was " << maxRainfall << "mm." << endl;
  cout << "It occurred in year " << FIRST_YEAR + yearMax
       << " and month " << monthMax+1 << " for the first time. " << endl;
  cout << "It happened " << nrTimes << " time(s)." << endl << endl;

  return 0;
}
```

Exercise 26.3

```
//Theatre reservations
#include <iostream>
using namespace std;

 const int NUM_ROWS = 5;
 const int NUM_SEATS = 9;
 const char ROW_CHAR[] = {'A', 'B', 'C', 'D', 'E'};

// mark 'num' seats in row 'r' with character 'marker', starting at seat 's'
void markSeats(char planP[][NUM_SEATS], char marker, int r, int s, int num)
{
  for (int i = s-1; i < s-1 + num; i++)
     planP[r][i] = marker;
}

// display the current theatre plan
void displayPlan(const char planP[][NUM_SEATS])
{
// display heading at the top of plan
  cout << endl << endl;
  for (int i = 1; i <= NUM_SEATS; i++)
    cout << '\t' << i << ' ';
  cout << endl;

 // display the plan itself
  for (int i = 0; i < NUM_ROWS; i++)
  {
    cout << ROW_CHAR[i] << ": ";
    for (int j = 0; j < NUM_SEATS; j++)
      cout << '\t' << planP[i][j] << ' ';
    cout << endl;
  }

 // display SCREEN at the bottom of the plan
  cout << "\t\tS\tC\tR\tE\tE\tN" << endl;
}

// find numerical index of row
int charToNumber(char c)
{
   return toupper(c) - 'A';    // 0, 1, 2, 3, or 4 will be returned
}

// test whether there are 'num' unreserved seats in row r starting at seat s
// (seat index as on plan)
// and display a message if an error occurs
void checkIfValid(const char planP[][NUM_SEATS], int r, int s, int num,
                                                          bool & valid)
{
int lastNumber;   // number of last ticket
char rowChar;

valid  = true;
lastNumber = s + num - 1;             //index as on plan
```

```
// Are too many tickets required?
 if (lastNumber > NUM_SEATS)
 {
   cout << "PROBLEM: There are not " << num << " seats available here";
   cout << endl;
   valid = false;
 }
 // Is one of the required seats already reserved?
 else
   for (int i = s - 1; i < lastNumber; i++)
      if (planP[r][i] == 'R')
      {
        cout << "PROBLEM: " << ROW_CHAR[r] << i+1 << " is already reserved";
        cout << endl;
        valid = false;
      }
}

int main( )
{
  char plan[NUM_ROWS][NUM_SEATS];
  char row;
  int numberOfTickets, seatNumber, rowNumber;
  bool validBooking;

// headings
  cout << "Theatre reservations: Rows A to E and seats 1 to 9"
       << endl << "To stop, type Q for row" << endl << endl;

// initially all seats should contain the character '-'
  for (int i = 0; i < NUM_ROWS; i++)
    markSeats(plan, '-', i, 1, NUM_SEATS);
  displayPlan(plan);

// first reservation
  cout << endl << "In which row do you want seats? ";
  cin >> row;                                          // index as on plan

// loop over all reservations
  while (toupper(row) != 'Q')
  {
    rowNumber = charToNumber(row);
    cout << "How many seats? ";
    cin >> numberOfTickets;
    cout << "Starting at which number? ";
    cin >> seatNumber;                                 // index as on plan
// test if booking is valid, and if valid, make the reservation:
    checkIfValid(plan, rowNumber, seatNumber, numberOfTickets, validBooking);
    if (validBooking)
    {
      markSeats(plan, 'R', rowNumber, seatNumber, numberOfTickets);
      cout << numberOfTickets << " seats reserved in row " << row << endl;
    }

// next reservation
    displayPlan(plan);
    cout << endl << "In which row do you want seats? ";
    cin >> row;
  }
  return 0;
}
```

LESSON 27

Exercise 27.1

```
//Converts a string to uppercase
#include <iostream>
#include <string>
using namespace std;

// convert string character by character
string upperCase(string s)
{
  string upper = "";
  int len = s.size( );
  for (int i = 0; i < len; i++)
    upper += toupper(s[i]);

  return upper;
}

int  main( )
{
  string sentence, newSentence;

  cout << "Original string:" << endl;
  getline(cin, sentence, '\n');
  newSentence = upperCase(sentence);
  cout << endl << "Converted to upper case:"
       << endl << newSentence << endl;

  return 0;
}
```

Exercise 27.2

```
//Finds and displays initials
#include <iostream>
#include <string>
using namespace std;

int main( )
{
  string namesAndSurname, initials;
  int pos;

  cout << "Please enter full names and surname, all separated by spaces: "
      << endl;
  getline(cin, namesAndSurname, '\n');
  initials = namesAndSurname[0];
  pos = namesAndSurname.find(" ");      // position of next space character
  while (pos > 0)
  {
    initials = initials + namesAndSurname[pos+1];
    pos = namesAndSurname.find(" ", pos+1);  // position of next space
  }
  cout << "Initials are: " << initials << endl;

  return 0;
}
```

Exercise 27.3

```
//Displays every word of a sentence on a separate line
// (removes punctuation symbols , and . and ? and ! at end of words)
#include <iostream>
```

```
#include <string>
using namespace std;

// removes punctuation symbol at end of word
string removed(string w)
{
 int len, i;
 len = w.size( );
  i = len - 1;               // indicates last character
  if ((w[i] == ',') || (w[i] == '!') || (w[i] == '?') || (w[i] == '.'))
    return w.substr(0, i);
  else
    return w;
}

int main( )
{
  string sentence, word;
  int start, pos, lengthOfWord;

  cout << "Please enter a sentence: ";
  getline(cin, sentence, '\n');

  start = 0;
  pos = sentence.find(" ");       // finds position of first space character

  while (pos > 0)
  {
    lengthOfWord = pos - start;
    word = sentence.substr(start, lengthOfWord);
    word = removed(word);         // removes possible punctuation symbol
    cout << word << endl;
    start = start + lengthOfWord + 1;
    pos = sentence.find(" ", start);   // finds position of next space
  }
// last word of sentence
  word = sentence.substr(start);
  word = removed(word);            // removes punctuation symbol
  cout << word << endl;

  return 0;
}
```

Exercise 27.4

We adapt the second version of the solution of Activity 27.c, namely the version where the string member function replace is used. Note that 'word' in this context does not mean a stand-alone English word, but rather a (sub)string of characters.

```
// Replaces all occurrences of a word in a sentence with
// another word (an adaptation of the solution to Activity 27.c).
#include <iostream>
#include <string>
using namespace std;

int main( )
{
  string sentence, word1, word2;
  int position, lengthSecondWord;

  // Input a sentence and two 'words'
  cout << "Enter a sentence: ";
  getline(cin, sentence, '\n');
```

```
  cout << "Enter a word (substring) to search for: ";
  cin >> word1;
  cout << "Enter a word (substring) to replace it with: ";
  cin >> word2;
  lengthSecondWord = word2.size( );

  // Search for the first word and replace all occurrences
  // of it with the second word
  position = sentence.find(word1);
  // find position of first occurrence of word1
  while (position != -1)
  {
    sentence.replace(position, word1.size( ), word2);
    position = sentence.find(word1, position + lengthSecondWord)
    // Finds position of next occurrence of word1.
    // The search starts directly after the last character
    // of word2 that has just replaced word1.
  }

  // Display the new sentence
  cout << "The new sentence is: " << sentence << endl;

  return 0;
}
```

LESSON 28

Exercise 28.1

```
//A time struct and its manipulation (add seconds to a specific time)
#include <iostream>
using namespace std;

struct Time
{
   int hours, minutes, seconds;
};

// add interval to time and get new time
void changeTime(Time & timeP, int interval)
{
  int t;
  t = timeP.seconds + interval;
  timeP.seconds = t % 60;
  t = timeP.minutes + t / 60;
  timeP.minutes = t % 60;
  timeP.hours = (timeP.hours + t / 60) % 24;
}

int main( )
{
  Time timeNow;
  int toBeAdded;

  cout << "Enter time. Give hours, minutes and seconds, "
       << "separated by blanks: ";
  cin >> timeNow.hours >> timeNow.minutes >> timeNow.seconds;
 cout << "How many seconds should be added? ";
 cin >> toBeAdded;

 changeTime(timeNow, toBeAdded);

 cout << "New time is " << timeNow.hours << " " << timeNow.minutes
      << " " << timeNow.seconds << endl;
```

```
 return 0;
}
```

Exercise 28.2

We have to:

1. define the required `struct TagInfo`*,*

2. write the user dialogue,

3. display the tag.

Apart from `getPassengerInfo` *we need at least two new functions. We called them* `getDeliveryInfo` *(for point 2 above) and* `displayTag` *(for point 3 above). We also decided to write three other functions, namely* `displayPassengerInfo` *that displays some of the user's dialogue,* `displayChars` *that displays a row of characters, and* `displayOneLine` *that displays one line of text.*

We cheated a little bit in this program, particularly in the `displayTag` function. We used the `width` member function of `cout`, which you probably haven't seen yet. Like `setf` and `precision`, the `width` member function can be used to specify how `cout` should display the values inserted in it. In particular, `width` sets the number of columns that should be used for the next insertion into the `cout` stream, and fills up any unused places on the left with blanks. Note that `width` only has a temporary effect, i.e. for the next use of `cout` only. For subsequent cout statements, `width` is reset to the default (with no blanks inserted).

We used `width` because without it, you would have to use a number of messy if statements to determine how many digits the mass field contains, to be able to display the correct number of spaces for the tag. (If you didn't print a border for your tag, you wouldn't have had this problem.).

```
//Luggage management at ABC AIRLINES
#include <iostream>
#include <string>
using namespace std;

const int TAG_WIDTH = 70;                          // related to tag size
struct TagInfo
{
 string name, flight, destination;
 float mass;
 char choice;
 string address[4];
};

// a dummy function (get information from ticketing and weighting system)
void getPassengerInfo(TagInfo & t)
{
  t.name = "Josephine van der Merwe";
  t.flight = "ABC123";
  t.destination = "JHB - Johannesburg International";
  t.mass = 20.0;
}
// display information from ticketing and weighting system
void displayPassengerInfo(TagInfo t)
{
  cout << "ABC Airlines" << endl;
  cout << "============" << endl;
  cout << "Passenger: " << t.name << endl;
  cout << "Flight: " << t.flight << endl;
  cout << "Destination: " << t.destination << endl;
  cout << "Mass of luggage: " << t.mass << " kg" << endl;
  cout << "============" << endl;
}
```

```
// users dialogue concerning luggage
void getDeliveryInfo(TagInfo & t)
{
// the information was correct: proceed
  cout << "============" << endl;
  cout << "We provide a FREE delivery service - up to 100 km from airport."
       <<   endl << "Would you like your luggage delivered (Y/N)? ";
  cin >> t.choice;
  if (toupper(t.choice) == 'Y')
  {
    //  luggage has to be delivered
    cin.get( );               // necessary between cin >> and getline
    cout << "Enter the address of your final destination (four lines):"
         << endl;
    for (int i = 0; i < 4; i++)
      getline(cin, t.address[i], '\n');
    cout << "Thank you!" << endl
         << "Your luggage will be delivered within 3 hours of arrival."
         << endl << "============" << endl << endl;
  }
}
// display a row of the same characters
void displayChars(char c, int i)
{
  for (int j = 0; j < i; j++)
    cout << c;
}
// display one line: start with *, then the information,
// then the correct number of blanks, followed by *,
// and move to next line
void displayOneLine(string info)
{
  cout << "*  " << info;
  displayChars(' ', TAG_WIDTH - 4 - info.size( ));
  cout << "*" << endl;
}
// display information to be printed on tag
void displayTag(TagInfo t)
{
  displayChars('*', TAG_WIDTH);
  cout << endl;
  displayOneLine("Name:        " + t.name);
  displayOneLine("Flight:      " + t.flight);
  displayOneLine("Destination: " + t.destination);

  //Can't use displayOneLine to display mass
  cout <<  "*  Mass:      ";
  cout.setf(ios::fixed);
  cout.precision(1);
  cout.width(6);            //See comment before code
  cout << t.mass << " kg";
  displayChars(' ', TAG_WIDTH - 24);
  cout << "*" << endl;

  if (toupper(t.choice) != 'Y')
    displayOneLine("*** PICK UP AT AIRPORT ***");
  else
  {
    displayOneLine("BAGGAGE TO BE DELIVERED TO:");
    for (int i = 0; i < 4; i++)
      displayOneLine("             " + t.address[i]);
  }
  displayChars('*', TAG_WIDTH);
  cout << endl;
}
```

```
int main( )
{
  TagInfo passenger;
  char correct;
  getPassengerInfo(passenger);
  displayPassengerInfo(passenger);

// check if the above is correct
  cout << "Is all the above information correct (Y/N)? ";
  cin >> correct;
  if (toupper(correct) != 'Y')
    cout << endl << "PLEASE INFORM THE CHECK-IN STAFF." << endl
         << "============" << endl << endl;
  else    // information was correct
  {
    getDeliveryInfo(passenger);
    displayTag(passenger);
  }

  return 0;
}
```

LESSON 29

Exercise 29.1

```
//LekkerKoop Ltd: analysis of production and sales
#include <iostream>
#include <string>
using namespace std;

const int MAX_ITEMS = 100;

struct ItemInfo
{
  string description;
  int nrProduced, nrSold, difference;
};
// input information of one item
void inputData(ItemInfo & thisItem, int nr)
{
  cin.get( );                          // necessary between cin and getline
  cout << endl << "Item " << nr+1 << endl << "Description: ";
  getline(cin, thisItem.description, '\n');
  cout << "Number produced: ";
  cin >> thisItem.nrProduced;
  cout << "Number sold: ";
  cin >> thisItem.nrSold;
  thisItem.difference = thisItem.nrProduced - thisItem.nrSold;
}
// output information of one item
void outputData(ItemInfo thisItem)
{
  cout << thisItem.description;
  cout << "\t\t" << thisItem.nrProduced;
  cout << "\t\t" << thisItem.nrSold;
  cout << "\t\t" << thisItem.difference << endl;
}
int main( )
{
  int n;
  ItemInfo itemData[MAX_ITEMS];
```

```
  cout << "How many types of items (maximum " << MAX_ITEMS << ")? ";
  cin >> n;

  for (int i = 0; i < n; i++)
    inputData(itemData[i], i);

  cout << endl << endl
       << "LEKKERKOOP LTD:   ITEMS WHERE PRODUCTION EXCEEDS SALES"
       << endl << endl;
  cout << "Description \tNo produced \tNo sold \tDifference"
       << endl << endl;

  for (int i = 0; i < n; i++)
    if (itemData[i].difference > 0)
      outputData(itemData[i]);

  cout << endl;

  return 0;
}
```

Exercise 29.2

You will see that we made three additional changes to the version in the Study Guide, namely we used `i + 1` (instead of `i`) in the relevant `cout` statements to indicate the number of the student or the number of the assignment. The reason is to have a better interface on the screen - humans normally count from 1 and not from 0.

```
//Process student assignment marks, second version. ( exercise 2)
#include <iostream>
#include <string>
using namespace std;

const int NUM_STUDS = 10;   // number of students
const int NUM_MARKS = 4;    // number of assignments

struct Student
{
  string name, studNo;
  int marks[NUM_MARKS];
  float average;
};

// input information of one student
void inputData(Student & studentP)
{
  cout << "Enter the student's information" << endl;
  cout << "Student number: ";
  cin >> studentP.studNo;
  cout << "Name: ";
  cin >> studentP.name;
  cout << "Marks:" << endl;
  for (int i = 0; i < NUM_MARKS; i++)
  {
     cout << "Assignment " << i + 1 << ": ";
     cin >> studentP.marks[i];
  }
}
// display information of one student (including average)
void displayData(Student studentP)
{
 cout << endl;
 cout << "Student number: " << studentP.studNo << endl;
 cout << "Name: " << studentP.name << endl;
 cout << "Marks:" << endl;
 for (int i = 0; i < NUM_MARKS; i++)
```

```
   cout << "   Assignment " << i + 1 << ": " << studentP.marks[i] << endl;
 cout << "Average: " << studentP.average << endl;
}
// calculate average of one student and store it in the average field
void average(Student & studentP)
{
  int total = 0;
  for (int i = 0; i < NUM_MARKS; i++)
    total += studentP.marks[i];
  studentP.average = float(total) / NUM_MARKS;
}
int main( )
{
  Student students[NUM_STUDS];
  float grandTotal = 0;
  float classAverage;

  //Input data and calculate average for each student
  for (int i = 0; i < NUM_STUDS; i++)
  {
    cout << endl << "Student " << i + 1 << endl;
    inputData(students[i]);
    average(students[i]);
    grandTotal += students[i].average;
  }
  //Calculate and display class average
  classAverage = grandTotal / NUM_STUDS;
  cout << endl << "The class average is " << classAverage << endl;

  //Display all students who did above average
  cout << endl << "These students did above average:" << endl << endl;
  for (int i = 0; i < NUM_STUDS; i++)
    if (students[i].average > classAverage)
      displayData(students[i]);
  return 0;
}
```

LESSON 30

Exercise 30.1

We have three member functions, namely `inputData`, `displayData` and `average`. Only `inputData` is a mutator. The other two are inspectors - they do not alter the data.

```
//Process student assignment marks, third version. (Lesson 30 exercise 1)

#include <iostream>
#include <string>
using namespace std;

const int NUM_STUDS = 10;   // number of students
const int NUM_MARKS = 4;    // number of assignments

class Student
{
  public:
    void inputData( );
    void displayData( ) const;
    float average( ) const;
  private:
    string name, studNo;
    int marks[4];
};
// input information of one student
```

```
void Student::inputData( )
{
  cout << "Enter the student's information" << endl;
  cout << "Student number: ";
  cin >> studNo;
  cout << "Name: ";
  cin >> name;
  cout << "Marks:" << endl;
  for (int i = 0; i < NUM_MARKS; i++)
  {
    cout << "Assignment " << i << ": ";
    cin >> marks[i];
  }
}
// display information of one student
void Student::displayData( ) const
{
  cout << endl;
  cout << "Student number: " << studNo << endl;
  cout << "Name: " << name << endl;
  cout << "Marks:" << endl;
  for (int i = 0; i < NUM_MARKS; i++)
    cout << "   Assignment " << i << ": " << marks[i] << endl;
}
// find average mark of one student
float Student::average( ) const
{
  int total = 0;
  for (int i = 0; i < NUM_MARKS; i++)
    total += marks[i];

  return float(total) / NUM_MARKS;
}
int main( )
{
  Student students[NUM_STUDS];
  float averages[NUM_STUDS];
  float grandTotal = 0;
  float classAverage;

  //Input data and calculate average for each student
  for (int i = 0; i < NUM_STUDS; i++)
  {
    cout << "Student " << i << endl;
    students[i].inputData( );
    averages[i] = students[i].average( );
    grandTotal += averages[i];
  }
  //Calculate and display class average
  classAverage = grandTotal / NUM_STUDS;
  cout << endl << "The class average is " << classAverage << endl;

  //Display all students who did above average
  cout << endl << "These students did above average:" << endl;
  for (int i = 0; i < NUM_STUDS; i++)
    if (averages[i] > classAverage)
      students[i].displayData( );

  return 0;
}
```

Exercise 30.2

Note that it is necessary to pass `account1` as a parameter when we call `inputTransactions`. Also note that it should be a reference parameter - instances of classes are passed in a way similar to structs, not like arrays.

```
// Processes a banking account and displays a monthly statement
// Version 2   ( exercise 2)
#include <iostream>
using namespace std;

const float DEPOSIT_FEE = 1.00;
const float BALANCE_FEE = 0.50;
const float WITHDRAWAL_FEE = 1.50;
const float OVERDRAWN_FEE = 5.00;

struct Transaction
{
   char type;
   float amount;
};

const int MAX_TRANSACT = 30;
class Account
{
  public:
    Account( );
    void deposit(float a);
    float balanceEnquiry( );
    void withdrawal(float a);
    void displayStatement( ) const;
  private:
    float balance;
    Transaction transacts[MAX_TRANSACT];
    int numTransacts;
    float feeTotal;
};

Account::Account( )
{
   balance = 0;
   numTransacts = 0;
   feeTotal = 0;
}
void Account::deposit(float a)
{
   balance += a;
   feeTotal += DEPOSIT_FEE;
   balance -= DEPOSIT_FEE;
   transacts[numTransacts].type = 'D';
   transacts[numTransacts].amount = a;
   numTransacts++;
}
float Account::balanceEnquiry( )
{
   feeTotal += BALANCE_FEE;
   balance -= BALANCE_FEE;
   transacts[numTransacts].type = 'B';
   transacts[numTransacts].amount = 0;
   numTransacts++;

   return balance;
}
void Account::withdrawal(float a)
{
   balance -= a;
   if (balance >= 0)
   {
      feeTotal += WITHDRAWAL_FEE;
      balance -= WITHDRAWAL_FEE;
```

```
   }
   else
   {
      feeTotal += OVERDRAWN_FEE;
      balance -= OVERDRAWN_FEE;
   }
   transacts[numTransacts].type = 'W';
   transacts[numTransacts].amount = a;
   numTransacts++;
}
void Account::displayStatement( ) const
{
   cout << endl << "Monthly Statement" << endl;
   cout << "=================" << endl;
   cout.setf(ios::fixed);
   cout.precision(2);
   for (int i = 0; i < numTransacts; i++)
   {
      switch (transacts[i].type)
      {
         case 'D':
            cout << "Deposit\t\tR" << transacts[i].amount << endl;
            break;
         case 'B':
            cout << "Balance enquiry" << endl;
            break;
         case 'W':
            cout << "Withdrawal\tR" << transacts[i].amount << endl;
            break;
      }
   }
   cout << "Total Fee\tR" << feeTotal << endl;
   cout << "---------------------------------" << endl;
   cout << "Closing balance\tR" << balance << endl;
}

void inputTransactions(Account & accountP)
{
   char type;
   float amount;

   cout << "Enter the transactions for the month" << endl;
   cout << "(D)eposit, (B)alance enquiry, (W)ithdrawal, E(X)it:"
          << endl;
   cin >> type;
   while (toupper(type) != 'X')
   {
      switch(toupper(type))
      {
         case 'D':
            cin >> amount;
            accountP.deposit(amount);
            break;
         case 'B':
            accountP.balanceEnquiry( );
            break;
         case 'W':
            cin >> amount;
            accountP.withdrawal(amount);
     }
      cin >> type;
   }
}

int main( )
```

```
{
   Account account1;

   inputTransactions(account1);
   cout << endl;
   account1.displayStatement( );

   return 0;
}
```

Exercise 30.3

```
//Swimmingpool
#include <iostream>
using namespace std;
class SwimmingPool
{
  public:
    SwimmingPool(int l, int w, int d, int s);
    int volume( ) const;
  private:
    int length, width, deepDepth, shallowDepth;
};
SwimmingPool::SwimmingPool(int l, int w, int d, int s)
{
  length = l;
  width = w;
  deepDepth = d;
  shallowDepth = s;
}
int SwimmingPool::volume( ) const
// The volume of the pool consists of a cube plus a wedge.
// The cube's dimensions are l x w x s.
// The wedge is l x w if considered from above. The side view is a
//     triangle with height (d-s),
//     and length l (perpendicular to the height).
{
  float v;

  v = (length * width * shallowDepth)  +
      (0.5 * length * (deepDepth - shallowDepth)) * width;

  return int(v + 0.5);   // round up to get an integer value for volume
}

int main( )
{
  int len, wid, deepDep, shallowDep, vol;

  cout << "Please give the dimensions of the swimming pool." << endl;
  cout << "Length: ";
  cin >> len;
  cout << "Width: ";
  cin >> wid;
  cout << "Depth at deep end: ";
  cin >> deepDep;
  cout << "Depth at shallow end: ";
  cin >> shallowDep;
  SwimmingPool pool(len, wid, deepDep, shallowDep);

  vol = pool.volume( );

  cout << endl << "The volume is " << vol << endl << endl;
```

```
  return 0;
}
```

Exercise 30.4

At this stage you will not be able to test your program because of the missing implementation of the member functions.

```
// Program to use the WavSound class
#include <iostream>
#include <string>
using namespace std;

class WavSound
{
 public:
    WavSound( );
    void loadFile(string fName);
    bool isLoaded( ) const;
    void play( );
    void stop( );
    void rewind( );
  private:
    bool loaded;
    string fileName;
};

int main( )
{
  WavSound sound1;
  char option;
  string fName;

  cout << "This program plays .wav sound files" << endl;
  cout << "= = = = = = = = = = = = = = = = = =" << endl;
  cout << "Choose one of the following options:" << endl;
  cout << "(L)oad, (P)lay, (S)top, (R)ewind, e(X)it: ";
  cin >> option;

  option = toupper(option);
  while (option != 'X')
  {
    switch (option)
    {
      case 'L':
        cout << "Enter the name of a .wav file: ";
        cin >> fName;
        sound1.loadFile(fName);
        if (!sound1.isLoaded( ))
          cout << "     ** The file " << fName << " was not found **" << endl;
        break;
      case 'P':
        if (sound1.isLoaded( ))
          sound1.play( );
        else
          cout << "** You must load a file first **" << endl;
        break;
      case 'S':
        sound1.stop( );
        break;
      case 'R':
        sound1.rewind( );
    }
```

```
    cout << "(L)oad, (P)lay, (S)top, (R)ewind, e(X)it: ";
    cin >> option;
    option = toupper(option);
  }

  return 0;
}
```