C++ Fundamentals
Exercises

Control Structures

1. Write a program that inputs a 4 digit year and then calculates whether or not it is a leap year. Note that there are 4 cases to consider:

years divisible by 400:	e.g. 2000	is a leap year
years divisible by 100:	e.g. 1900	not a leap year
years divisible by 4:	e.g. 2012	is a leap year
other years :	e.g. 2011	not a leap year

2. Using a variation of the above program, calculate the number of days in the inclusive date range *'1st January 2000'* to *'31st December 2999'*.

3. Write a program that prints out the square, cubes and fourth power of the first 20 integers.

4. Write a program that prints out the first 20 Fibonacci numbers. You can use the web to find out the definition of the Fibonacci sequence.

5. Write a program that calculates the ratio of successive pairs of the first 20 Fibonacci numbers. Does the ratio appear to converge to a number?

6. Write a program that prints out the sum, difference, product and dividend of two complex numbers.

7. Use the *split()* function to split a fullname such as
   "Julius Caesar"
   into two strings called firstName and lastName.

8. Use the *range* function to generate two separate tuples containing the list of integers from 10 to 19 and from 30 to 39. Tuples are immutable, so how can you form a tuple that has all the elements of the other two tuples?

9. *int(x[,radix])* converts a string to an int using the supplied radix. Use this function to compute the decimal value of the binary number 101010101010101010101.

10. Create a dictionary consisting of people's names and salaries. Initialise the dictionary with at least 5 pairs of names and salaries. Now print the dictionary. Why is the printed order different from the order you used to initialise the dictionary?

11. Write a function that calculates the area of a rectangle. Decide how many input parameters your function needs. The area should be returned from the function. Write a test program that calls your function with different sets of test data.

12. Write a function that rotates the values of 3 variables. For example:

```
int x = 100
int y = 200
int z = 300
Rotate( ... )
# x is now 200
# y is now 300
# z is now 100
```

13. Write a function to calculate Factorials. Try out

```
factorial(1)
factorial(10)
factorial(40)
factorial(100)
```

14. Write a function that takes a string and capitalises the first character of the string and ensures the remaining characters are converted to lower case. Use the following test data:

```
UpperFirst("test1")
UpperFirst("mIxEdCaSe")
UpperFirst("UPPER")
UpperFirst("lower")
UpperFirst("oPPOSITE")
```

15. Write a function that takes an integer list as a parameter and doubles the value of each element of the array.

16. Write a function that takes two *int* arrays (same size) as parameters and adds the arrays together, element by element. Print out the array as part of the function.

Exception Handling

17. Write a program that calculates the factorial of an integer in the range 2 to 10. Add exception handling code to prevent calculating a result where the input number is larger than 10, or any negative integer. Make sure you can handle the case where the input is not an integer.

Classes

18. Create a class that represents a bank account. Add methods to allow a customer to **deposit()** and **withdraw()** money and provide a method **getBalance()**. Write a test program to check out your class.

19. Modify the previous example to add facilities for providing an overdraft. You will need to define a **setOverdraft()** method.

20. Write a program that counts the number of lines in a file.

21. Write a file copy program.

22. Create a file call *TestData.txt* with test data consisting of one number per line using your favourite editor. Your job is to read the entire file into memory so that you can compute the sum of all the numbers.

23. Try the previous example with other test files that may contain non integral data. Use exception handling to filter out lines that don't contain integers.

24. Write a program that concatenates three files into a new file.

25. Write a program that reads a file, reverses the order of lines in the file and then saves the changes in a new file.