Text book information:
AQA A Level Computer Science by Bob Reeves
ISBN: 978-1-471-83951-1
You will each need to buy a copy of this and you will need to bring it to every lesson

William Perkin C of E High School

Appendix 1

Computer Science

Section 1: Java setup

You are expected to complete this section.

Installing JDK8

1. Go to http://www.oracle.com/technetwork/java/javase/downloads/index.html , download and install Java development kit 8 (JDK 8).

Installing NetBeans 8

1. Go to https://netbeans.org/ and download and install **NetBeans IDE 8.x** (where x is any number).
2. You will use the IDE to write Java code therefore download and install **Java SE**.

Learning to code in Java

Most of you will be new to Java, but may have programmed in another language if you studied Computer Science GCSE.

If you are new to Computer Science and programming, you will need to spend some time going through the online Java tutorials and learning the basics of programming before you attempt the tasks below.

If you have programmed before in another language, then start off with the tutorials to familiarise yourself with the new syntax. When you feel ready, have a go at the tasks, but keep the tutorials available for help reference.

Link to Java online tutorials: http://www.homeandlearn.co.uk/java/java.html

Continue to next page for section 2

Section 2: First Java programs

You are expected to complete both tasks in this section.

<u>Task 1: Computing Body Mass Index</u>

Body Mass Index (BMI) is a measure of health based on height and weight. It can be calculated by taking your weight in kilograms and diving it by the square of your height in metres.

$$BMI = \frac{weight\ (kg)}{(height)^2\ (m)^2}$$

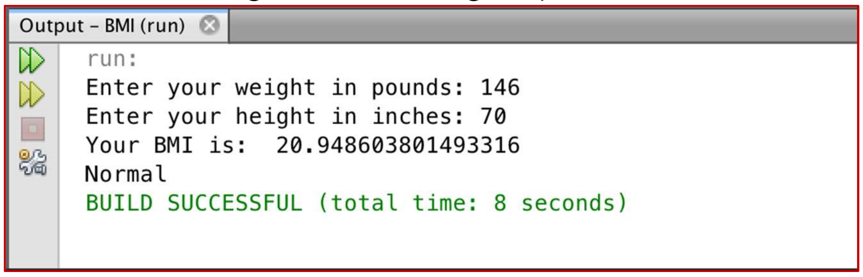
The interpretation of BMI for people 20 years or older is as follows.

BMI	Interpretation
Below 18.5	Underweight
18.5 – 24.9	Normal
25.0 – 29.9	Overweight
Above 30.0	Obese

The following program will prompt the user to enter their weight in pounds and height in inches and displays the BMI of the user.
1 pound = 0.45359237 kg
1 inch = 0.0254 meter

```java
package bmi;

import java.util.Scanner;

public class BMI {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Prompt the user to enter their weight in pounds
        System.out.print("Enter your weight in pounds: ");
        double weight = input.nextDouble();

        //Prompt the user to enter their height in inches
        System.out.print("Enter your height in inches: ");
        double height = input.nextDouble();

        final double KG_PER_POUNDS = 0.45359237;  // Constant
        final double M_PER_INCH = 0.0254;  // Constant

        // Compute BMI
        double weightInKg = weight * KG_PER_POUNDS;
        double heightInMetres = height * M_PER_INCH;
        double bmi = weightInKg / (heightInMetres * heightInMetres);

        // Display result
        System.out.println("Your BMI is:  " + bmi);
        if (bmi < 18.5)
            System.out.println("Underweight");
        else if (bmi < 25)
            System.out.println("Normal");
        else if (bmi <30)
            System.out.println("Overweight");
        else
            System.out.println("Obese");
    }
}
```

Enter the information below to get the following output.



```
run:
Enter your weight in pounds: 146
Enter your height in inches: 70
Your BMI is:  20.948603801493316
Normal
BUILD SUCCESSFUL (total time: 8 seconds)
```

What you need to do:

1. Write the code above into Java
2. Run the code and enter the same values as the output shown above, you should get the same results!
3. Create a new evidence document in a program such as MS word
4. Type a narration in your evidence document to explain how the program work. Your narration should be detailed enough to give a novice a good understanding of what the program is doing.

<u>Task 2: Lottery Number Generator</u>

You have been asked to produce your own program to randomly generate lottery numbers. It must generate six random numbers between 1 and 49 inclusive + one for the bonus ball, note that the numbers should not repeat in any draw.

What you need to do:

1. Research how random numbers can be created in Java
2. Design and create your program
3. Test your program several times to demonstrate that the numbers are truly random.
4. Print screen your code and test output and add it to your evidence document.
5. Type a narration of how your program works. Your narration should be detailed enough to give a novice a good understanding of what the program is doing.

Continue to next page for section 3

Section 3: Coding challenges

You are expected to stretch your knowledge and skills and attempt as much as possible from this section, researching the areas you don't know.

Task 3: Opposites

Your teacher asks you to develop a program that will help her KS2 students to practice opposite words for their examination. The program should randomly select two different pairs of words from the lists below and display a question based on the selection.

Word lists

[hot, summer, hard, dry, simple, light, weak, male, sad, win, small, ignore, buy, succeed, reject, prevent, exclude]

[cold, winter, soft, wet, complex, darkness, strong, female, happy, lose, big, pay attention, sell, fail, accept, allow, include]

For example if **hot** and **weak** are selected then the question displayed is

"hot is to cold as **weak** is to ____________?"

When the user types their answer the program should display whether the user is correct or not.

The program should start by asking the user for their name. The program should then display ten random questions. After displaying 10 questions the program should display the user's name and their final score out of 10.

```
score ← 0
word_list1 ← [hot, summer, hard, dry, heavy, light, weak, male, sad, win, small, ignore, buy,
                    succeed, reject, prevent, exclude]

word_list2 ← [cold, winter, soft, wet, light, darkness, strong, female, happy, lose, big, pay
attention, sell,
                     fail, accept, allow, include]

PROCEDURE make_question(number1, number2)
        OUTPUT word_list1[number1], "is to", word_list2[number1], "as",
                      word_list1[number2] , "is to __________________ ?"
END PROCEDURE

INPUT name
FOR index FROM 1 TO 10
        pick1 ← random(0, LEN(word_list1) - 1) #LEN returns the length of the list
        pick2 ← random(0, LEN(word_list2) - 1) #LEN returns the length of the list
        {random(a,b) generates a random number between a and b inclusive}

        WHILE pick2 = pick1
                pick2 ← random(0, LEN(word_list2) - 1)
        END WHILE

        {Now make a question}
        make_question(pick1, pick2)
        correct_answer ← word_list2[pick2]
        INPUT user_answer
        IF user_answer = correct_answer THEN
                score ← score + 1
                OUTPUT "Correct answer"
        ELSE
                OUTPUT "Wrong answer"
        END IF
        NEXT index
END FOR
OUTPUT name, "you got", score, "out of 10"
```

What you need to do:

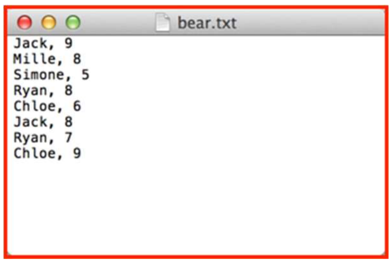
1. Create a Java program that implements the algorithm above.
2. Test your program to demonstrate that it works.
3. Print screen your code and test output and add it to your evidence document.
4. Type a narration of how your program works. Your narration should be detailed enough to give a novice a good understanding of what the program is doing.

<u>Task 4: Expanding functionality</u>

The teacher teaches three different classes (bee, bear, duck) and wants to use this test as an assessment. Before this test can be used as an assessment she has to ensure that no question is repeated, in the same test. For example if the following should not be allowed in the same test.

hot is to cold as light is to _________ ?
winter is to summer as hot is to __________ ?

The teacher would also like to store the result into a text file so that she can analyse the data at a later stage. The figure below shows an example of a part of the file for bear.



```
Jack, 9
Mille, 8
Simone, 5
Ryan, 8
Chloe, 6
Jack, 8
Ryan, 7
Chloe, 9
```

What you need to do:

1. Create a Java program that solves this problem
2. Test your program to demonstrate that it works.
3. Print screen your code and test output and add it to your evidence document.
4. Type an evaluation of your solution.