CSCI 111 Practicum - due in class by Friday

TASK 1. UNDERSTANDING PSEUDO-CODE AND TRANSLATING TO A C++ PROGRAM

   a. Using *CodeBlocks*, create a project named '*task1*' and implement the following algorithm in a C++ program.
Note: Be sure to use the recommended style.

```
BEGIN Calculate Average
   integer1 = 10
   integer2 = 12
   sum = integer1 + integer2
   average = sum/2
   PRINT "The average of " + integer1 + " and " + integer2 + "
      is " average
END Calculate Average
```

   b. Have you forgotten to include the appropriate file header as described last week? If so, add it now. ***Note: if you submit files for assessment without a file header you will be penalised***.

   c. You should have used appropriate indentation of your code as described in the *CSCI111 Style Guide*. Indentation is vital to enhance readability of code and poor usage in assessment items will result in mark penalization. If you have not indented your code, do so now.

   d. Ask your tutor to view your code to verify that the style is correct.

   e. Compile your program

   f. What data types have you chosen for the *integer1, integer2, sum* and *average* variables and why?
Answer:

   g. What is the name of the executable file created by the compile command in step (e)?
Answer:

   h. Execute your program. What is the output?
Answer:

   i. Close the file.

TASK 2. REVIEW OF PROGRAM BASICS

   a. What does the value returned from `main` indicate?

Answer:

b. Why must the *include* file `iostream` be used?
Answer:

c. What is the name of the:
   i. Standard *output* stream? _____
   ii. Standard *input* stream? _____
d. What will happen if a file submitted for assessment does not contain a file header?
Answer:

e. What are the two ways to insert new line characters into the output stream?
Answer:

f. What are the two ways to insert comments within a program?
Answer:

g. What is the global area and what is it used for? (Will be covered tomorrow)
Answer:

h. Every executable C++ statement must be terminated with a ___.

i. Explain the reason for including a data type in the `main` function's header.
Answer:

j. List at least 5 of the *built-in* data types.
Answer:

k. Write C++ statements to:
   i. Output the value `1`:

   ii. Output the character `A`:

   iii. Output the string `Hello`:

iv. Produce the following output
```
Hello
World:
```

v. Declare an integer named `anInt`:

vi. Declare a floating point variable named `aFloat`:

vii. Declare a character variable named `aChar`:

viii. Declare an integer named `a2ndInt` and initialize it's value to 0:

ix. Assign the value of 100 to `a2ndInt`:

x. Assign the value of `a2ndInt` to `anInt`:

xi. Output the value of `anInt`:

TASK 3. CHALLENGE TASKS

1:

a. Write a C++ program to implement the following pseudo-code. Remember to use the data type that is the *best* for the particular variable and the recommended style.

```
BEGIN Calculate Pay
      hrly_rate = 25.67
      hrs_worked = 23
      tax_rate = 17
      gross_income = hrly_rate x hrs_worked
      tax_payable = gross_income x tax_rate / 100
      net_income = gross_income - tax_ payable
      PRINT "Hrs worked: " + hrs_worked
      PRINT "Hrly Rate:" + hrly_rate
      PRINT "Gross Income:" + gross_income
      PRINT "less TAX:" + tax_withheld
      PRINT "Net Income:" + net_income
END Calculate Pay
```

The output should be similar to:
```
Hrs worked:            xxx
Hrly Rate:             xxx
------------------------
Gross Income:          xxx
Less TAX:              xxx
========================
Net Income:            xxx
```

Where `xxx` are the values output from the respective variables.

b. Save your work and compile and run your program.

c. Verify that the output of your program is correct. This is a vital aspect of programming and you must *always* test your program *thoroughly* before submitting it.

2:

Implement a code solution for the (lawn mowing contract cost) design task from last week.

The program helps a contractor compute the cost estimate for mowing services. It prompts her/him for the dimensions of the property and the buildings within it. The average mowing time is 1 minute per 10 feet^2 per annum (assumes 32 visits per year) @ $1 per minute. Your code solution should display appropriate data and prompts to the user. Verify that the output of your program is correct. This is a vital aspect of programming and you must *always* test your program *thoroughly* before submitting it.

TASK SUBMISSION:

Demonstrate all completed code tasks to your teacher or tutor(s).