Assignment 1

- Must be done individually. See website for due date.
- Submit all deliverables to the CourSys: https://courses.cs.sfu.ca/
- Late penalty is 10% per calendar day (each 0 to 24 hour period past due).
  - No submissions will be accepted after 2 days late (48 hours/20% penalty).
- Do not show another student your code, do not copy code found online, and do not post questions about the assignment online. Please post all questions to the Piazza forum (either as a public or private message).
- See the marking guide for details on how each part will be marked. Variable names, commenting, and code indentation are always important.

1. Height From Inches

Write a C++ program named height.cpp which asks the user to enter their height in inches, and then it prints to the screen their height in feet and inches, as well as just inches. Plus converts it to height in meters. See sample outputs below.

- There are 12 inches in a foot, and 39.3701 inches in a meter.
- Display feet and inches using ' and ', such as 5 foot 2 inches is 5'2".
- The “Write either...” line must be printed using formatting exactly as shown in the samples below. Your output prints to the screen “Write either:...”, complete with the punctuation shown in the sample outputs.
- Convert the inches into meters.
  - Hint: Carefully choose your data types.
- Make the program colourful:
  - The “(inches)” in the initial prompt must be red.
  - Each of the quoted ways to write the height must be different colours and bold.
  - You can select your own colours, but don’t use either white or black.
  - Use at least 2 colours in the metric output line, any way you like.
  - When this program (as with all labs/assignments) finishes, it must:
    - returned the colours back to normal, and
    - print a line-feed after it last output (i.e. your last cout should still have an endl).

```
1. Sample output.
Enter your height (inches): 76
Write either: "I'm 6'4"!" \ "I'm 76" tall!"
(Height in metric: 1.9304m)
Output 1: Sample output.

Output 2: Sample output.
```
2. Fruit Box Calculator

Create a C++ program named fruitbasket.cpp which helps create fruit baskets from donated apples and oranges. Its purpose is to calculate the number of baskets that can be created. One goal is that all baskets should be as similar as possible; hence, if each basket requires 4 pieces of fruit and there are 25 apples and 16 oranges then:

- 41 pieces of fruit creates 10 baskets with one piece of fruit left over.
- Each basket should get at least 2 apples and 1 orange, plus one one more piece of fruit (either an apple or an orange).

Requirements

- The program asks the user to enter (via the keyboard) three integers:
  - The number of apples donated.
  - The number of oranges donated.
  - The required number of fruit pieces per basket.
- Assume all input values are integers between 0 and 1,000,000.
  - You need not check for invalid values such as negative numbers, too big numbers (like 2 million), fractional numbers (like 3.1), or non-numeric values (like “hello world”).
  - Assume there is enough fruit to make at least one basket. (Otherwise, your program may crash, and we don’t yet know enough C++ to handle this yet!)
- Calculate the following:
  - The total number of baskets possible to create using the donated fruit, each containing exactly the required number of pieces of fruit.
  - Minimum number of apples per basket. This is, if the apples were split evenly amongst the baskets, at least how many apples are there per basket?\(^1\)
  - Minimum number of oranges per basket (same as above).
  - Once created, each basket will be tied off with three pieces of ribbon per basket. Calculate the total number of pieces of ribbon required.
  - Number of pieces of fruit required to make one more basket.
- Your program’s output should look like the samples shown below.
  - Note that data must lineup, as shown in the sample outputs.
  - Format to handle numbers up to 4 digits wide (as shown in the outputs). This should right align the values.
- Your program’s output must display using colours, and be very readable:
  - Make the initial welcome title, and all section titles be bold and a colour of your choice.
  - Make the name of the thing which the user is entering be a colour of your choice.
  - Make all the calculated numbers be a colour of your choice.
  - See the sample outputs for a possible set of colours.
- Notes:
  - Do not make any text white as depending on the terminal background colour, this can be hard to see.
  - When changing back to normal colour, use ANSI_NORMAL. Don’t use ANSI_BLACK because the terminal’s background colour could be black!
  - When your program finishes, it must return to ANSI_NORMAL on a new line.

\(^1\) Note that this turns out not to be “at least how many apples are there per basket” because in some cases the minimum number of apples + minimum number of oranges could be more than a basket requires! This is really, “There are at least this many apples available to put in each basket, even if all baskets turn out not to need this many.”
***************
Local Fruit Box Calculator
***************

Enter the number of apples donated: 25
Enter the number of oranges donated: 16
Enter the number of fruit pieces per basket: 4

Input Values:
-------------
# Apples Donated: 25
# Oranges Donated: 16
# Fruit Pieces per Basket: 4

Basket Creation Numbers:
------------------
# Baskets to create: 10
Min # apples per basket: 2
Min # oranges per basket: 1
# Ribbons to tie baskets: 30
# Pieces needed to complete one more basket: 3

Output 3: Sample output.

***************
Local Fruit Box Calculator
***************

Enter the number of apples donated: 2562
Enter the number of oranges donated: 9103
Enter the number of fruit pieces per basket: 5

Input Values:
-------------
# Apples Donated: 2502
# Oranges Donated: 9103
# Fruit Pieces per Basket: 5

Basket Creation Numbers:
------------------
# Baskets to create: 2321
Min # apples per basket: 1
Min # oranges per basket: 3
# Ribbons to tie baskets: 6963
# Pieces needed to complete one more basket: 5

Output 4: Sample output showing no extra fruit donated.
***************
Local Fruit Box Calculator
***************

Enter the number of apples donated: 100
Enter the number of oranges donated: 123456
Enter the number of fruit pieces per basket: 10

Input Values:
----------------
# Apples Donated: 100
# Oranges Donated: 123456
# Fruit Pieces per Basket: 10

Basket Creation Numbers:
-------------------------
# Baskets to create: 12355
Min # apples per basket: 0
Min # oranges per basket: 9
# Ribbons to tie baskets: 37065
# Pieces needed to complete one more basket: 4

Output 5: Sample output showing input size larger than expected (alignment problems). Your solution should also have this look.

***************
Local Fruit Box Calculator
***************

Enter the number of apples donated: 0
Enter the number of oranges donated: 5
Enter the number of fruit pieces per basket: 10

RUN FINISHED; Floating point exception; core dumped;

Output 6: Sample output showing your program may crash when it cannot make even one basket. We will not test for this condition because we don't yet know how to handle it. (The last line of the output is generated by the OS when it detects that the program crashed due to a “floating point exception”.)
3. Deliverables
Submit the items listed below to the CourSys: https://courses.cs.sfu.ca/
(Note you do not have to create a zip file for these; CourSys will accept them directly. However, your file names must be all lower case!)

- height.cpp
- fruitbasket.cpp
- ansi_codes.h: This will likely be exactly the file provided on the course website; however, you may add extra content to the file for other colours/styles you want to support.

Please remember that all submissions will be automatically compared for unexplainable similarities. We expect that everyone's submissions will be somewhat similar, given the nature of this assignment, but please make sure you do your own original work; we are checking!

Do not email/give your code to another student. Do not accept code from another student. Do not post your code online.