KIT103/KMA155 Programming Assignment 2: Logic

Enter your answers to these questions in the accompanying Python script file `programming2.py`. Unlike programming assignment 1, this time many of your answers will be function definitions, so instead of a dictionary to store your answers the script file contains many function 'stubs'. These stubs have a header specifying their name and parameters but currently return dummy values.

Submit your completed script file to the *Programming Assignment 2: Logic* dropbox on MyLO by **1500 (3pm) Wednesday 28 August 2019**.

Test your solutions thoroughly. Your submission is expected to run without failure (even if it doesn't produce the correct answer for each question). If we have to correct your submission in order for it to run then the maximum total mark you can receive will be 3/5.

Tip: You have been provided with a useful function in one of the practical classes to use while testing your solutions to Questions 2–4.

Question 1: Riding the Logic Circuit (1.5 marks)

The Fantastic Function Fun Park has a popular roller coaster (called the Logic Circuit) with a somewhat complex set of rules for who can go on it. They have tasked you with writing a function that can tell people whether they can ride it or not. The function has three parameters: the person's height in whole cm, age in whole years and a Boolean indicating if they are accompanied by an adult. It can return three possible string messages: 'Sorry, you cannot ride' if the rules exclude a person from riding the Logic Circuit; 'Find an adult' if the person (child in this case) can ride as long as they come back with an adult; and 'You can ride' if they are allowed to ride the Logic Circuit.

The rules are as follows:

- Anyone less than 120cm or taller than 200cm cannot ride
- Anyone younger than 7 cannot ride
- If a child aged between 7 and 9 (inclusive) wishes to ride, they can do so if they are accompanied by an adult, otherwise they are told that they must 'Find an adult'
- Everyone else is allowed to ride the roller coaster

Your task is to implement the rules above by completing the implementation of `q1_coaster_check` in the assignment script file. It is currently very pessimistic, telling all patrons that they are not allowed on. Complete the implementation by using any combination of if statements and Boolean expressions. The if statements may be nested as you see fit.

Note that:

- the function must `return` the appropriate value, *not* print it; and
- the string messages must be exactly as above.

Full marks will be given to solutions that use as few tests as possible. A solution that works for all cases, but which repeats a test or implements each rule completely separately, will get 1 mark. A solution that works for only many possible inputs will receive 0.5 marks, while a solution that works rarely or not at all will receive no marks.

Sample Test Data

Question 2: Implementing predicates as functions (1 mark)

There are stub (i.e., incomplete) implementations of each of these predicates as functions in the assignment script file (named `q2_a` through `q2_d`). Replace `None` in each of these stub functions with a Boolean expression that implements the predicate as it is written. The functions already have the required parameters listed. All parameters are Boolean.

a. $\neg(a \wedge b) \wedge (a \vee b)$
b. $a \wedge (\neg b \vee \neg c \vee \neg d)$
c. $\neg a \wedge a$
d. $(a \vee b) \wedge (b \vee c)$

Question 3: Simplifying predicates (1 mark)

For each subpart (a)–(d) in Question 2, write a simplified implementation. There are stub functions named `q3_a` through `q3_d` for you to complete.

Question 4: The Letter Detector (1.5 marks)

The ACME Logic Company has a simple object recognition system that can identify when a particular arrangement of curves represents a letter (not which letter, only that it is one of a subset of all letters). They have a camera that can detect curves similar to opening and closing parentheses ( ) in the upper and lower parts of an image. Although the image may be hand-drawn, their detector 'sees' it as if it were like part of a segmented display, like this:

a ( ) b
c ( ) d

To determine if the arrangement of curves looks like a letter they use the following truth table (for convenience the truth table is presented using 0 for False and 1 for True, but the function accepts four Boolean variables a–d and returns a Boolean value). The Letter column is only for information; the function will not attempt to actually identify the letter.

a	b	c	d	Output	Letter
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	c
0	0	1	1	1	o
0	1	0	0	0	
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	1	d
1	0	0	0	0	

a	b	c	d	Output	Letter
1	0	0	1	1	S
1	0	1	0	1	E
1	0	1	1	1	b
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	B

You have two tasks:

a. Transfer the information in the truth table to the Karnaugh map stored in `q4_kmap` in the script file, treating the columns as *ab* and rows as *cd*. `q4_map` is a list of lists, but you can treat it as if it were a table: replace the appropriate locations with 1s. **Tip:** Draw the Karnaugh map on paper first and identify the groups. You won't be able to show the groups in the script file, but this will help with part (b). (0.5 marks)
b. Replace the complicated Boolean expression currently in `q4_acme_letter_detector` with a simplified expression determined from your Karnaugh map. (1 mark)

How your assignment is assessed

Your submitted Python script will be assessed initially by a Python program that will execute your code and check the values generated by your answers against the expected results. This produces a tab-delimited text file named AssessmentReport.txt with columns containing the *expected* result, the *actual* result produced by your code, the *score* achieved for that question (and *maximum* score possible), and any *feedback* explaining the score.

Next, a human assessor will inspect your submission and adjust the computer-based assessment accordingly: code that produced the wrong value but is close to correct will have some marks restored; code that produced the correct answer but did so incorrectly may have marks reduced. The assessor will add further feedback to the assessment report text file as needed.

You can download the Assessment Report from MyLO: go to Assignments then click feedback link in the Evaluation Status column. View it by opening in Excel or Numbers.

Assessment criteria

Submissions that require modification in order to run without error will receive a maximum mark of 3/5.

- Question 1:
  - Full marks (1.5) for a solution that is correct and uses as few Boolean tests as possible
  - 1 for a solution that works for all cases but which repeats a test or implements each rule completely separately
  - 0.5 for a solution that works for only many possible inputs
  - 0 for a solution that works rarely or not at all
- Question 2: Answers must be valid translations of the Boolean expressions into Python to receive any marks

- Question 3: Full marks (0.25) will be awarded to answers that are valid simplifications of the original expression, expressed in Python. Half marks may be awarded to an answer that is partially simplified (that is, is part way to the correct answer)
- Question 4a: Full marks (0.5) will be awarded to a K-map with correct placement of 1s. Half marks will be awarded if the rows and columns have been switched. Partial marks may be awarded if the placement of 1s is almost correct (e.g., a single 1 out of place)
- Question 4b:
    - Full marks (1) for a solution that correctly implements a fully-simplified Boolean expression, based on the K-map
    - Half marks (0.5) for a solution that produces the correct result but has only been partially simplified
    - No marks for a solution that doesn't produce the correct result for all valid inputs, or which has not been modified from the original code

Working independently: This is an individual assignment so your submission should be substantially your own work.
