SCALE FOR PROJECT READY SET BOOLE (/PROJECTS/READY-SET-BOOLE)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere.42paris.fr:vogsphere/intra-uuid-e123e2ba-b9bc
```

Comments

This subject was created by a couple of active members of the Paris-based student association 42AI. The authors are Luc Lenôtre (llenotre) and Tristan Duquesne (tduquesn).

This subject was developed during the first half of 2021.

The writing of the subject and design of the exercises was done under the impulse and direction of Luc Lenôtre; while the proofreading, editing, and writing of "cultural" mathematical content was done mostly by Tristan Duquesne.
For future corrections of the scale or subject, please contact the 42AI association via contact@42ai.fr or the current 42AI pedagogical supervisor.

Introduction

'For any issue or suggestion, please contact 42Paris peadagogical team and 42AI pedagogical supervisor.

As usual, you have to observe the following courtesy rules:

- Remain polite, courteous, respectful, and constructive throughout the evaluation process. The well-being of the community depends on it.

- Identify with the evaluated person or group the eventual dysfunctions of the assignment. Take the time to discuss and debate the problems you may have identified.

- You must consider that there might be some differences in the understanding of and approach to project instructions, and the scope of its functionalities, between you and your peers. Always remain open-minded and grade them as fairly as possible. The pedagogy is valid only and only if peer-evaluation is conducted seriously.

The goal of the subject is to discover the basics of computer-related mathematics with Boolean Algebra and Set Theory!

Guidelines

General rules
- Only grade the work that is in the student or group's git repository.

- Double-check that the git repository does belong to the student. Ensure that the work is the one expected for the corrected exercise and don''t forget to verify that the command "git clone" is run in an empty folder.

- Check carefully that no malicious aliases were used to make you evaluate files that are not from the official repository.

- To avoid any surprises, carefully check that both the evaluating and the evaluated students have reviewed the possible scripts used to facilitate the grading.

- If the evaluating student has not completed that particular project yet, it is mandatory for them to read the entire subject prior to starting the defense.

- Use the flags available on this scale to signal an empty repository, non-functioning program, cheating, and so forth.
In these cases, the grading is over and the final grade is 0, or -42 in case of cheating. However, except the exception of cheating, you are encouraged to continue to discuss your work even if the later is in progress in order to identify any issues that may have caused the project failure and avoid repeating the same mistake in the future.

- Use the appropriate flag.

- Remember that for the duration of the defense, no other unexpected, premature, or uncontrolled termination of the program, else the final grade is 0.

- You should never have to edit any file except the configuration file if the latter exists. If you want to edit a file, take the time to explain why with the evaluated student and make sure both of you agree on this.

- Your exercises are going to be evaluated by other students,
make sure that your variable names and function names are appropriate and civil.'

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/56766/en.subject.pdf)

Exercise 00 - Adder

Complexity

Ask the student to justify the complexity of the function. It must be at most O(1) in time and O(1) in space.

✓ Yes ✕ No

Used operators

Check that the only operators that were used in the function are:

- & (bitwise AND)
- | (bitwise OR)
- ^ (bitwise XOR)
- << (left shift)
- >> (right shift)
- = (assignment)
- == , != , < , > , <= , >= (comparison operators)
- The increment operator (only to increment the index of a loop)

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✕ No

Basic tests

Check the behaviour of the function with the following parameters:

- 'adder(0, 0)' gives '0'
- 'adder(1, 0)' gives '1'
- 'adder(0, 1)' gives '1'
- 'adder(1, 1)' gives '2'
- 'adder(1, 2)' gives '3'
- 'adder(2, 2)' gives '4'

Feel free to perform more tests on your own.

✓ Yes ✕ No

Exercise 01 - Multiplier

Complexity

Ask the student to justify the complexity of the function. It must be at most O(1) in time and O(1) in space.

✓ Yes ✕ No

Used operators

Check that the only operators that were used in the function are:

- & (bitwise AND)
- | (bitwise OR)
- ^ (bitwise XOR)
- << (left shift)
- >> (right shift)
- = (assignment)
- == , != , < , > , <= , >= (comparison operators)
- The increment operator (only to increment the index of a loop)

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✕ No

Basic tests

Check the behaviour of the function with the following parameters:

- 'multiplier(0, 0)' gives '0'
- 'multiplier(1, 0)' gives '0'
- 'multiplier(0, 1)' gives '0'
- 'multiplier(1, 1)' gives '1'
- 'multiplier(1, 2)' gives '2'
- 'multiplier(2, 2)' gives '4'

Feel free to perform more tests on your own.

✓ Yes ✕ No

Exercise 02 - Gray code

Basic tests

Check the behaviour of the function. The binary representation of the returned number must correspond to the encoding of the given parameter in Gray code.
You can use an online Binary -> Gray code converter to make evaluation easier.

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✗ No

Exercise 03 - Boolean evaluation

Complexity

Ask the student to justify the complexity of the function. It must be at most O(n) in time.

✓ Yes ✗ No

Basic tests

Check the behaviour of the function with the following formulas:

- '0!' gives 'true'
- '1!' gives 'false'
- '00|' gives 'false'
- '10|' gives 'true'
- '01|' gives 'true'
- '11|' gives 'true'
- '10&' gives 'false'
- '11&' gives 'true'
- '11^' gives 'false'
- '10^' gives 'true'
- '00>' gives 'true'
- '01>' gives 'true'
- '10>' gives 'false'
- '11>' gives 'true'
- '00=' gives 'true'
- '11=' gives 'true'
- '10=' gives 'false'
- '01=' gives 'false'

Feel free to perform more tests on your own

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✕ No

Composition

Check the behaviour of the function with the following formulas:

- '11&0|' gives 'true'
- '10&1|' gives 'true'
- '11&1|' gives 'true'
- '11&1|1^' gives 'false'
- '01&1|1=' gives 'true'
- '01&1&1&' gives 'false'
- '0111&&&' gives 'false'

Feel free to perform more tests on your own.

✓ Yes ✕ No

Exercise 04 - Truth table

Complexity

Ask the student to justify the complexity of the function. It must be at most O(2^n) in time.

✓ Yes ✕ No

Basic tests

Check the behaviour of the function with the following formulas:

- 'A' must print:
  '
A	
0	0
1	1
  '
- 'A!' must print:
  '
A	

```
| 0 | 1 |
| 1 | 0 |
'
```

- 'AB|' must print:

```
'
| A | B | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |
'
```

- 'AB&' must print:

```
'
| A | B | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
'
```

- 'AB^' must print:

```
'
| A | B | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
'
```

- 'AB>' must print:

```
'
| A | B | |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
'
```

- 'AB=' must print:

```
'
| A | B | |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
'
```

- 'AA=' must print:

  ```
  '
  | A | |
  |---|---|
  | 0 | 1 |
  | 1 | 1 |
  '
  ```

Feel free to perform more tests on your own.

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✕ No

Composition

Check the behaviour of the function with the following formulas:

- 'ABC==' must print:

  ```
  '
  | A | B | C | |
  |---|---|---|---|
  | 0 | 0 | 0 | 0 |
  | 0 | 0 | 1 | 1 |
  | 0 | 1 | 0 | 1 |
  | 0 | 1 | 1 | 0 |
  | 1 | 0 | 0 | 1 |
  | 1 | 0 | 1 | 0 |
  | 1 | 1 | 0 | 0 |
  | 1 | 1 | 1 | 1 |
  '
  ```

- 'AB>C>' must print:

  ```
  '
  | A | B | C | |
  |---|---|---|---|
  | 0 | 0 | 0 | 0 |
  | 0 | 0 | 1 | 1 |
  | 0 | 1 | 0 | 0 |
  | 0 | 1 | 1 | 1 |
  | 1 | 0 | 0 | 1 |
  | 1 | 0 | 1 | 1 |
  | 1 | 1 | 0 | 0 |
  | 1 | 1 | 1 | 1 |
  '
  ```

- 'AB>A>A>' must print:
  ```
  '
  | A | B | |
  |---|---|---|
  | 0 | 0 | 1 |
  | 0 | 1 | 1 |
  | 1 | 0 | 1 |
  | 1 | 1 | 1 |
  '
  ```

By the way, the last formula is called Pierce's Law. You may want to check out what this is if you want to go deeper in mathematical logic.

Feel free to perform more tests on your own.

✓ Yes ✗ No

Exercise 05 - Negation Normal Form

Basic tests

Check the behaviour of the function with the following formulas:

- 'A'
- 'A!'
- 'AB&!'
- 'AB|!'
- 'AB>!'
- 'AB=!'

For each case, every occurence of '!' must be placed after a variable, and the truth table must be the same.

Feel free to perform more tests on your own.

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✗ No

Composition

Check the behaviour of the function with the following formulas:

- 'ABC||'
- 'ABC||!'
- 'ABC|&'
- 'ABC&|'
- 'ABC&|!'
- 'ABC^^'

- 'ABC>>'

For each case, every occurence of '!' must be placed after a variable, and the truth table must be the same.

Feel free to perform more tests on your own.

✓ Yes ✗ No

Exercise 06 - Conjunctive Normal Form

Basic tests

Check the behaviour of the function with the following formulas:

- 'A'
- 'A!'
- 'AB&!'
- 'AB|!'
- 'AB>!'
- 'AB=!'

For each case, every occurence of '!' must be placed after a variable, every conjunction must be located at the end of the formula, and the truth table must be the same.
Feel free to perform more tests on your own

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✗ No

Composition

Check the behaviour of the function with the following formulas:

- 'ABC||'
- 'ABC||!'
- 'ABC|&'
- 'ABC&|'
- 'ABC&|!'
- 'ABC^^'
- 'ABC>>'

For each case, every occurence of '!' must be placed after a variable, every conjunction must be located at the end of the formula, and the truth table must be the same.

Feel free to perform more tests on your own.

✓ Yes ✕ No

Exercise 07 - SAT

Complexity

Ask the student to justify the complexity of the function. It must be at most O(n^2) in time.

✓ Yes ✕ No

Basic tests

Check the behaviour of the function with the following formulas:

- 'A' gives 'true'
- 'A!' gives 'true'
- 'AA|' gives 'true'
- 'AA&' gives 'true'
- 'AA!&' gives 'false'
- 'AA^' gives 'false'
- 'AB^' gives 'true'
- 'AB=' gives 'true'
- 'AA>' gives 'true'
- 'AA!>' gives 'true'

Feel free to perform more tests on your own. For each case, every occurence of '!' must be placed after a variable, every conjunction must be located at the end of the formula, and the truth table must be the same.

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✕ No

Composition

Check the behaviour of the function with the following formulas:

- 'ABC||' gives 'true'
- 'AB&A!B!&&' gives 'false'
- 'ABCDE&&&&' gives 'true'
- 'AAA^^' gives 'true'
- 'ABCDE^^^^' gives 'true'

Feel free to perform more tests on your own. For each case, every occurence of '!' must be placed after a variable, every conjunction must be located at the end of the formula, and the truth table must be the same.

✓ Yes ✕ No

Exercise 08 - Powerset

Complexity

Ask the student to justify the complexity of the function. It must be at most O(2^n) in time and space.

✓ Yes ✕ No

Basic tests

Check the function's behaviour with different sets. Each times, the number of subsets in the resulting powerset must be equal '2^n' where 'n' is the length of the set.
The order of the elements in the returned array doesn't matter.

Try the following:

- '[]' gives '[[]]' (1 subset)
- '[0]' gives '[[], [0]]' (2 subset)
- '[0, 1]' gives '[[], [0], [1], [0, 1]]' (4 subset)
- '[0, 1, 2]' gives '[[], [0], [1], [2], [0, 1], [1, 2], [0, 2], [0, 1, 2]]' (8 subset)

Feel free to perform more tests on your own.

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✕ No

Exercise 09 - Set evaluation

Basic tests

Try the following:

- 'A' with '[[]]', the function must return '[]'
- 'A!' with '[[]]', the function must return '[]'
- 'A' with '[[42]]', the function must return '[42]'
- 'A!' with '[[42]]', the function must return '[]'
- 'A!' with '[[], [42]]', the function must return '[42]'
- 'AB|' with '[[0, 1, 2], []]', the function must return '[0, 1, 2]'
- 'AB&' with '[[0, 1, 2], []]', the function must return '[]'
- 'AB&' with '[[0, 1, 2], [0]]', the function must return '[0]'

- 'AB&' with '[[0, 1, 2], [42]]', the function must return '[]'
- 'AB^' with '[[0, 1, 2], [0]]', the function must return '[1, 2]'
- 'AB>' with '[[0], [1, 2]]', the function must return '[1, 2]'
- 'AB>' with '[[0], [0, 1, 2]]', the function must return '[0, 1, 2]'

Feel free to perform more tests on your own.

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✕ No

Composition

Try the following:

- 'ABC||' with '[[], [], []]', the function must return '[]'
- 'ABC||' with '[[0], [1], [2]]', the function must return '[0, 1, 2]'
- 'ABC||' with '[[0], [0], [0]]', the function must return '[0]'
- 'ABC&&' with '[[0], [0], []]', the function must return '[]'
- 'ABC&&' with '[[0], [0], [0]]', the function must return '[0]'
- 'ABC^^' with '[[0], [0], [0]]', the function must return '[0]'
- 'ABC>>' with '[[0], [0], [0]]', the function must return '[0]'

Feel free to perform more tests on your own.

✓ Yes ✕ No

Exercise 10 - Curve

Basic tests

Try passing pairs of values to the function. For each unique pair of values, the function must return a unique value between 0 and 1 (included).

Check for the use of any forbidden mathematical functions (see the subject).

✓ Yes ✕ No

Exercise 11 - Inverse function

Basic tests

Use the previous function to test this one. Try executing the function 'inverse_map(map(x, y))'. For every pair of values 'x' and 'y', this function must return the exact same value.

Check for the use of any forbidden mathematical functions (see the subject).

Yes No

Ratings

Don't forget to check the flag corresponding to the defense

✓ Ok

Empty work Incomplete work Invalid compilation Cheat Crash

Incomplete group Leaks Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation