WORKSHEET

A15.7 Calculation and theory questions

Exam preparation · A level · OCR H446 1.4.1, AQA 7517 4.4.1.2, Eduqas A500QS 1.8 · about 40 min

Name _____ Class _____ Date _____

What this lesson is about

Number representation, floating point, Boolean logic, state machines and complexity answered quickly and checked.

- Always show your working
- Two's complement
- Floating point
- Boolean logic
- State machines
- Complexity

Questions

6 marks in all

1. What is the denary value of the 8-bit two's complement number 11101100? [1 mark]

2. A normalised floating point number has mantissa 01101000 (point after the first bit) and exponent 0011. What is its denary value? [1 mark]

3. Which mantissa is normalised? [1 mark]

- ☐ **A** 10100000
- ☐ **B** 11010000
- ☐ **C** 00110000
- ☐ **D** 11100000

4. In how many rows of its truth table is NOT (A AND B) OR C true? [1 mark]

5. An algorithm has a loop over n items inside another loop over n items. What is its time complexity? [1 mark]

- ☐ **A** $O(n^2)$
- ☐ **B** $O(n)$
- ☐ **C** $O(\log n)$
- ☐ **D** $O(2^n)$

6. What does this finite state machine print? [1 mark]

```
t = {("even", "0"): "even", ("even", "1"): "odd", ("odd", "0"): "odd", ("odd", "1"): "even"}
for text in ["110", "111", ""]:
    state = "even"
    for symbol in text:
        state = t[(state, symbol)]
    print(repr(text), state)
```

The task: theory at speed

Write each method as a function, then print the answers. - `twos(bits)` : `bits` is a string of `0`s and `1`s of any length. Return its value as a two's complement integer: the leftmost bit is worth minus its normal place value. - `float_value(mantissa, exponent)` : both are strings of bits. The mantissa is two's complement with the binary point just after the leftmost bit; the exponent is a two's complement integer. Return the number's value as a `float` . - `run_fsm(transitions, state, accepting, text)` : `transitions` is a dictionary from `(state, symbol)` tuples to the next state, `state` is the start state, `accepting` is a set of accepting states, and `text` is a string of symbols. Return `True` if the machine ends in an accepting state. Then print exactly these six lines, using the functions for the values: 1. `11101100 = <value>` using `twos` ; 2. `01101000 x 2^0011 = <value>` and 3. `10100000 x 2^1111 = <value>` using `float_value` ; 4. `1101: <result>` and 5. `1001: <result>` , where the result is `accepted` or `rejected` , using `run_fsm` with the even-number-of-1s machine above (states `"even"` and `"odd"` , start `"even"` , accepting `{"even"}`); 6. `true rows: <n>` , counting the rows where `NOT (A AND B) OR C` is true with three nested loops over A, B and C. The robot stays still.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

def twos(bits):
    pass
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a15-7-theory-at-speed/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Normalise the mantissa `00011010` with exponent `0101`, keeping the value the same.
2. Add `hex_to_denary(text)` without using `int(text, 16)`, and test it on `2F` and `FF`.
3. Design a state machine that accepts binary strings ending in `01`, and test it with `run_fsm`.