WORKSHEET

A7.6 Error checking and correction

Data representation · A level · AQA 7517 4.5.5.3, Eduqas A500QS 2.8 · about 30 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Parity bits, majority voting, checksums and check digits, with checksums on the robot's radio messages.

- Parity bits
- Majority voting
- Checksums
- Check digits
- Comparing the methods

Questions

6 marks in all

1. Even parity is used. What parity bit is added to the 7 bits 1101011? [1 mark]

2. Even parity is used. The byte 01101011 is received. What can the receiver conclude? [1 mark]

- ☐ **A** There is an error, because the byte has an odd number of 1s
- ☐ **B** The byte is correct
- ☐ **C** Bit 3 is wrong and can be corrected
- ☐ **D** Exactly two bits are wrong

3. Majority voting sends each bit three times. The groups 110 011 000 are received. Give the three data bits. [1 mark]

4. Which of these methods can correct an error without the data being sent again? [1 mark]

Tick every answer that is true.

- ☐ **A** Majority voting
- ☐ **B** A parity bit
- ☐ **C** A checksum
- ☐ **D** A check digit

5. A checksum adds the character codes of a message modulo 256. Which error does it always miss? [1 mark]

- ☐ **A** Two characters swapping places
- ☐ **B** One character changed
- ☐ **C** One bit flipped
- ☐ **D** A character missing

6. What does this program print? [1 mark]

```
digits = [9, 7, 8, 1, 8, 6, 1, 9, 7, 8, 2, 7]
total = 0
for i in range(12):
    weight = 1 if i % 2 == 0 else 3
    total = total + digits[i] * weight
print(total, (10 - total % 10) % 10)
```

The task: checksums on the radio

Write `checksum(text)`. The parameter `text` is a string. It returns the sum of the character codes (use `ord`) modulo 256, as a string of exactly two uppercase hex digits (you may use `format(n, "02X")`). 1. For each command in `GO 20`, `LEFT 90`, `STOP`, in that order, build the packet `<command>*<checksum>`, send it with `send()`, and print `sent <packet>`. Send the packet you built, not typed-in text. 2. For each packet in `received`, split it at the `*`, recalculate the checksum of the text part and print `<packet> ok` if it matches the checksum that arrived, or `<packet> corrupt` if it does not.

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

def checksum(text):
    return "00"

received = ["GO 20*18", "G0 20*18", "STOP*46", "LEFT 80*B4"]
```

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

Do it on the robot

www.bugbotlab.com/learn/a7-6-error-checking-and-correction/

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

Challenges

1. Show a corruption of `STOP` that the checksum misses.
2. Send each byte of `GO` three times as bits and write the majority-voting decoder.
3. Weight each character code by its position before adding. Does `OG 20` now get a different checksum from `GO 20`?