ANSWERS

A14.8 Moral, ethical, social and cultural issues

Software development, law and ethics · A level · OCR H446 1.5.2, AQA 7517 4.8.1 · about 45 min

What this lesson is about

Automated decisions, AI, surveillance, personal data at scale, censorship, piracy, the environment and accessibility, with an autonomous robot that explains every decision.

- Why computer scientists carry responsibility
- The words in the question
- The issues the specifications name
- Case study: the autonomous delivery robot
- Structuring a discussion

Questions

5 marks in all

1. A CV-screening system trained on a company's past hiring decisions rejects most applicants from one group. What is the most likely cause? [1 mark]

- ☐ **A** Bias in the training data, which reflects unfair past decisions
- ☐ **B** A syntax error in the program
- ☐ **C** The computer deliberately chose to discriminate
- ☐ **D** Too few features in the user interface

> **Answer:** A. A machine learning system learns the patterns in its data, including unfair ones.

2. Which of these are named by OCR as moral and ethical issues of digital technology? [1 mark]

Tick every answer that is true.

- ☐ **A** Automated decision making
- ☐ **B** Censorship and the internet
- ☐ **C** Layout, colour paradigms and character sets
- ☐ **D** Clock speed
- ☐ **E** Normalisation

> **Answer:** A, B, C. OCR also names computers in the workforce, AI, environmental effects, monitoring behaviour, analysing personal information, and piracy and offensive communications.

3. AQA says software and its algorithms embed moral and cultural values. What does this mean? [1 mark]

- ☐ **A** Each rule in a program is a decision someone made about what should happen to people
- ☐ **B** Programs contain comments about ethics
- ☐ **C** Software can only be written by philosophers
- ☐ **D** Computers have their own values

> **Answer:** A. A robot that gives way to some people and not others has a value built into its code.

4. Why does the scale of software raise the stakes of a developer's decisions? [1 mark]

- ☐ **A** One program can run on millions of devices, so one choice affects millions of people
- ☐ **B** Larger programs are always slower
- ☐ **C** Scale only matters for hardware
- ☐ **D** Big programs cannot contain bias

> **Answer:** A. Scale lets an individual developer do great good, or great harm, at once.

5. Why does an autonomous robot that logs each decision with its reason help with accountability? [1 mark]

- ☐ **A** Its decisions can be checked, questioned and corrected afterwards
- ☐ **B** It makes the robot drive faster
- ☐ **C** It means no human is ever responsible
- ☐ **D** It removes the need for testing

> **Answer:** A. An explainable decision can be examined when something goes wrong; an unexplained one cannot.

The task: explain every decision

The robot starts 66 cm from a person standing in the corridor, facing them. Write a program that approaches safely and **explains every decision it makes**. Repeat: read the gap with `distance()` and decide: - if the gap is more than 40 cm, the decision is `drive`: LED green, then move forward 5 cm at speed 60; - if the gap is more than 25 cm but not more than 40 cm, the decision is `slow`: LED orange, then move forward 2 cm at speed 30; - otherwise the decision is `stop`: LED red, and leave the loop without moving. Print a line only when the decision **changes** from the previous one (including the first decision), in the form `<time> s: <decision>, because the gap is <gap> cm`, where `<time>` is `clock()` and `<gap>` is the reading the decision was based on. Change the LED at the same moment. After the loop, play a note and print `handing over to a human`. That makes exactly four lines. Do not touch the person.

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

last = ""      # the previous decision, so a line is printed only when it changes
```

The hint students can ask for: Each time round the loop, read the gap and decide from it which of the three decisions applies. Keep the last decision in a variable, and only print and change the LED when the new decision differs from it. The stop decision ends the loop; the hand-over comes after.

A solution

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

last = ""
while True:
    gap = distance()
    if gap > 40:
        decision, colour = "drive", "green"
    elif gap > 25:
        decision, colour = "slow", "orange"
    else:
        decision, colour = "stop", "red"
    if decision != last:
        led(colour)
        print(f"{clock()} s: {decision}, because the gap is {gap} cm")
        last = decision
    if decision == "drive":
        forward(60, distance=5)
    elif decision == "slow":
        forward(30, distance=2)
    else:
        break

tone(440, 0.5)
print("handing over to a human")
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.