ANSWERS

A2.4 Abstraction and models

Recursion and computational thinking · A level · OCR H446 2.1.1, AQA 7517 4.4.1.3 · about 20 min

What this lesson is about

Representational abstraction, generalisation, problem reduction, and a grid model of the mat that differs from reality.

- What abstraction is, and why it is needed
- Representational abstraction
- Abstraction and reality
- Abstraction by generalisation
- Problem abstraction and reduction

Questions

6 marks in all

1. What is abstraction? [1 mark]

- ☐ **A** Removing details that do not matter for the purpose, to focus on those that do
- ☐ **B** Breaking a problem into smaller parts
- ☐ **C** Writing a program in a high-level language
- ☐ **D** Hiding the code of a program from its users

> **Answer:** A. Which details are unnecessary depends on what the model is for. Breaking a problem into parts is decomposition.

2. A metro map shows stations in the right order on each line but ignores real distances and bends. What kind of abstraction is it? [1 mark]

- ☐ **A** Representational abstraction
- ☐ **B** Abstraction by generalisation
- ☐ **C** Functional abstraction
- ☐ **D** Data abstraction

> **Answer:** A. It is a representation arrived at by removing unnecessary details.

3. "A time-of-flight sensor is a kind of distance sensor, which is a kind of sensor." Which kind of abstraction is this? [1 mark]

- ☐ **A** Abstraction by generalisation (categorisation)
- ☐ **B** Representational abstraction
- ☐ **C** Procedural abstraction
- ☐ **D** Problem reduction

> **Answer:** A. Grouping by common characteristics into an "is a kind of" hierarchy is generalisation.

4. A room is modelled as a grid of 20 cm cells, blocked if the centre of the cell is inside an obstacle. What is a danger of this abstraction? [1 mark]

- ☐ **A** A thin obstacle between cell centres is missing from the model, so a planned route could hit it
- ☐ **B** The model is too large to store
- ☐ **C** The robot cannot drive sideways
- ☐ **D** The grid cannot be searched by an algorithm

> **Answer:** A. The model differs from reality: detail smaller than a cell can disappear, and decisions made from the model can then be wrong.

5. Euler turned the Königsberg bridges puzzle into a question about points and lines. What is this an example of? [1 mark]

- ☐ **A** Problem abstraction or reduction
- ☐ **B** Decomposition
- ☐ **C** Information hiding
- ☐ **D** Automation

> **Answer:** A. Details were removed until the puzzle became a problem about a graph, which could be answered.

6. Why is abstraction needed when solving problems with computers? [1 mark]

Tick every answer that is true.

- ☐ **A** Real situations have too much detail to store and process
- ☐ **B** A simpler model can be solved by an algorithm
- ☐ **C** The same model can be reused for similar situations
- ☐ **D** It makes the model exactly match reality

> **Answer:** A, B, C. An abstraction is deliberately not the same as reality; that is what makes it useful and also what must be checked.

The task: a grid model of the mat

The mat is 100 cm by 100 cm, with (0, 0) in the bottom left corner. The four obstacles in the figure are these rectangles, each `(x, y, width, height)` in cm, where `(x, y)` is the rectangle's bottom left corner: `(25, 64, 40, 10)`, `(72, 5, 8, 50)`, `(5, 22, 30, 16)` and `(84, 40, 12, 60)` Build the 5 by 5 grid model with 20 cm cells. - Write a function `blocked(cx, cy)` that returns `True` if the point `(cx, cy)` is inside any rectangle (a point exactly on an edge counts as inside) and `False` otherwise. - A cell is blocked if its centre is blocked. Row 0 is the **top** row of the mat and column 0 the left, so the cell in row 0, column 0 has its centre at (10, 90). - Print the five rows, row 0 first, each as a string of five characters: `#` for a blocked cell and `.` for a free one. - Then print `blocked: <n>`, the number of blocked cells.

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

obstacles = [(25, 64, 40, 10), (72, 5, 8, 50), (5, 22, 30, 16), (84, 40, 12, 60)]

def blocked(cx, cy):
    return False
```

The hint students can ask for: Work out the centre of each cell from its row and column first, remembering that row 0 is the top of the mat, where y is largest. A cell is blocked if its centre is inside any one of the rectangles, edges included. Build each row as a string.

A solution

```
from bugbot import *
connect()

obstacles = [(25, 64, 40, 10), (72, 5, 8, 50), (5, 22, 30, 16), (84, 40, 12, 60)]

def blocked(cx, cy):
    for x, y, w, h in obstacles:
        if x <= cx <= x + w and y <= cy <= y + h:
            return True
    return False

count = 0
for row in range(5):
    line = ""
    for col in range(5):
        if blocked(10 + 20 * col, 90 - 20 * row):
            line = line + "#"
            count = count + 1
        else:
            line = line + "."
    print(line)
print("blocked:", count)
```

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