WORKSHEET

A2.6 Decomposition, composition and automation

Recursion and computational thinking · A level · OCR H446 2.1.3, AQA 7517 4.4.1.9 · about 20 min

Name _____ Class _____ Date _____

What this lesson is about

Thinking procedurally, compound procedures and compound data, and putting a model into action.

- Decomposition
- Composition
- Automation

Questions

5 marks in all

1. What is procedural decomposition? [1 mark]

- ☐ **A** Breaking a problem into sub-problems that each accomplish an identifiable task, which may be further subdivided
- ☐ **B** Combining procedures into a compound procedure
- ☐ **C** Removing unnecessary detail from a problem
- ☐ **D** Running a model to solve a problem

2. Which are examples of composition? [1 mark]

Tick every answer that is true.

- ☐ **A** A fetch procedure built from drive_to and grip
- ☐ **B** A tree made of nodes, each holding a list of smaller trees
- ☐ **C** A route stored as a list of shapes, each a list of legs
- ☐ **D** Removing the colours from a map

3. Put the steps of automation in order. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ `Execute the code`

___ `Create an algorithm`

___ `Implement the model in data structures`

___ `Implement the algorithm in program code`

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

```
def double(x):
    return x * 2

def add_three(x):
    return x + 3

def compose(f, g):
    def both(x):
        return f(g(x))
    return both

h = compose(double, add_three)
print(h(4))
```

5. When thinking procedurally about a problem, what should you identify? [1 mark]

Tick every answer that is true.

- ☐ **A** The components of the problem
- ☐ **B** The components of the solution
- ☐ **C** The order of the steps needed
- ☐ **D** The sub-procedures necessary
- ☐ **E** The programming language's keywords

The task: a route built from pieces

A **leg** is a tuple `(direction, cm)`, where `direction` is one of `"forward"`, `"backward"`, `"left"` or `"right"` and `cm` is a distance. A **shape** is a list of legs. A **route** is a list of shapes. The starter builds this route from two shapes: `step = [("forward", 20), ("right", 20)]`, `square = [("forward", 20), ("right", 20), ("backward", 20), ("left", 20)]`, `route = [step, square, step]` Write three procedures, each built from the one before: - `drive_leg(leg)` prints `leg <direction> <cm>` and drives that leg at speed 60, sliding sideways for `left` and `right` (the robot never turns); - `drive_shape(shape)` drives every leg of a shape, using `drive_leg`; - `drive_route(route)` drives every shape of a route, using `drive_shape`. Drive the route, then print `route done: <shapes> shapes, <legs> legs`, which for this route is `route done: 3 shapes, 8 legs`. The robot should finish 40 cm right of and 40 cm up from where it started.

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

step = [("forward", 20), ("right", 20)]
square = [("forward", 20), ("right", 20), ("backward", 20), ("left", 20)]
route = [step, square, step]

def drive_leg(leg):
    direction, cm = leg

def drive_shape(shape):
    pass

def drive_route(route):
    pass
```

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

Do it on the robot

www.bugbotlab.com/learn/a2-6-decomposition-composition-and-automation/

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

Challenges

1. Add a shape `zigzag` and a new route that uses it twice, without changing any of the three procedures.
2. Draw the route as a tree like the tidy job. What are the leaves?
3. Decompose "play a game of robot football" as far as three levels. Which sub-procedures would two robots share?