WORKSHEET

A2.7 Thinking ahead and thinking logically

Recursion and computational thinking · A level · OCR H446 2.1.2, AQA 7517 4.4.1.1 · about 25 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Inputs, outputs and preconditions, caching and reuse, decisions and conditions, and solving logic problems.

- Inputs, outputs and preconditions
- Caching
- Reusable components
- Thinking logically
- Following and writing algorithms

Questions

6 marks in all

1. What is a precondition? [1 mark]

- ☐ **A** Something that must be true before a subroutine or algorithm runs for it to work correctly
- ☐ **B** The first line of a program
- ☐ **C** A condition tested at the end of a loop
- ☐ **D** The output an algorithm must produce

2. Which are drawbacks of caching? [1 mark]

Tick every answer that is true.

- ☐ **A** The cached copy can be out of date (stale)
- ☐ **B** It uses extra memory or storage
- ☐ **C** Deciding what to keep and when to discard it adds complexity
- ☐ **D** Repeated requests are always slower

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

```
cache = {}
calls = 0

def fib(n):
    global calls
    calls = calls + 1
    if n in cache:
        return cache[n]
    if n < 2:
        answer = n
    else:
        answer = fib(n - 1) + fib(n - 2)
    cache[n] = answer
    return answer

print(fib(10), calls)
```

4. Why use reusable program components? [1 mark]

Tick every answer that is true.

- ☐ **A** They save development time
- ☐ **B** Code that has been used and tested many times is more reliable
- ☐ **C** A fault is fixed in one place for every program that uses it
- ☐ **D** They are always faster than code written for one job

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

```
for culprit in ["Ada", "Bolt", "Cog"]:
    ada = culprit == "Bolt"
    bolt = culprit != "Bolt"
    cog = culprit != "Cog"
    if [ada, bolt, cog].count(True) == 1:
        print(culprit)
```

6. Which constructs are enough to write any algorithm? [1 mark]

Tick every answer that is true.

- ☐ **A** Sequence
- ☐ **B** Assignment
- ☐ **C** Selection
- ☐ **D** Iteration
- ☐ **E** Recursion

The task: routes with a cache

The robot moves on a grid, only ever one cell right or one cell up. `routes(right_steps, up_steps)` is the number of different routes from one corner to a cell that many steps right and up. Both parameters are whole numbers. If either is 0 there is exactly 1 route (a straight line); otherwise every route arrives either from the cell to the left or from the cell below, so `routes(r, u) = routes(r - 1, u) + routes(r, u - 1)`. - Write `routes` recursively, with a dictionary `cache` so no pair is ever worked out twice. - Count every call of `routes` in a global `calls`, including calls answered from the cache. - **Precondition**: both parameters must be 0 or more. At the start of `routes`, if either is negative, `raise ValueError` with a message of your choice. Then print, in this order: 1. `routes(3, 2) = <answer>`; 2. `routes(16, 16) = <answer>`, with `calls` set back to 0 just before this call; 3. `calls: <n>`, the calls made by that one call (without a cache this would be over a billion); 4. `refused: <message>`, by calling `routes(-1, 4)` inside `try` and printing the error's message in `except ValueError`.

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

cache = {}
calls = 0

def routes(right_steps, up_steps):
    if right_steps == 0 or up_steps == 0:
        return 1
    return routes(right_steps - 1, up_steps) + routes(right_steps, up_steps - 1)

print("routes(3, 2) =", routes(3, 2))
```

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

Do it on the robot

www.bugbotlab.com/learn/a2-7-thinking-ahead-and-thinking-logically/

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

Challenges

1. Remove the cache from `routes` and time `routes(12, 12)`. Then work out roughly how long `routes(16, 16)` would take.
2. A robot caches its map of the room. Give one situation where the cache helps and one where it causes a crash.
3. Remove clue 4 from the race. How many orders fit now, and what does that tell you about checking a solution?