2.1 Elements of Computational Thinking: Test Pack

OCR H446 · Calculators not allowed · about 35 minutes

Total Marks

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 Nia Okafor is part of a small software team designing a new online catalogue and booking system for Bramwell Public Library. Members will be able to search the catalogue, reserve books that are currently out on loan, and receive a notification when a reserved book becomes available.

(a) Define what is meant by abstraction as a computational thinking technique. **(2)**

(b) Define what is meant by decomposition as a computational thinking technique. **(2)**

(c) Using decomposition, identify three separate sub-problems that Nia's team would need to solve in order to build the library system described above. For one of your three sub-problems, state one piece of detail that could be abstracted away (ignored) at this early design stage. **(4)**

(Total for Question 1 is 8 marks)

© Revision Library | revisionlibrary.co.uk

2 A CPU in a laptop used by Kwame's college uses a 4-stage instruction pipeline, Fetch, Decode, Execute and Write-back, to increase throughput by overlapping the execution of successive instructions. Each stage takes exactly one clock cycle, and one clock cycle lasts 2 ns. The processor must execute a program made up of 10 instructions.

(a) State two reasons why pipelining increases the throughput of a processor compared with executing each instruction's stages sequentially, with no overlap between instructions. **(2)**

(b) Calculate the total time, in ns, to execute all 10 instructions if pipelining is NOT used, i.e. each instruction completes all 4 stages before the next instruction begins. **(3)**

(c) Calculate the total time, in ns, to execute all 10 instructions using the 4-stage pipeline, assuming no stalls or hazards occur. **(3)**

(d) Calculate the speed-up factor achieved by using the pipeline for this program, giving your answer correct to 2 decimal places. **(2)**

(e) State what is meant by a pipeline hazard, and give one example of a situation that could cause the pipeline to stall. **(2)**

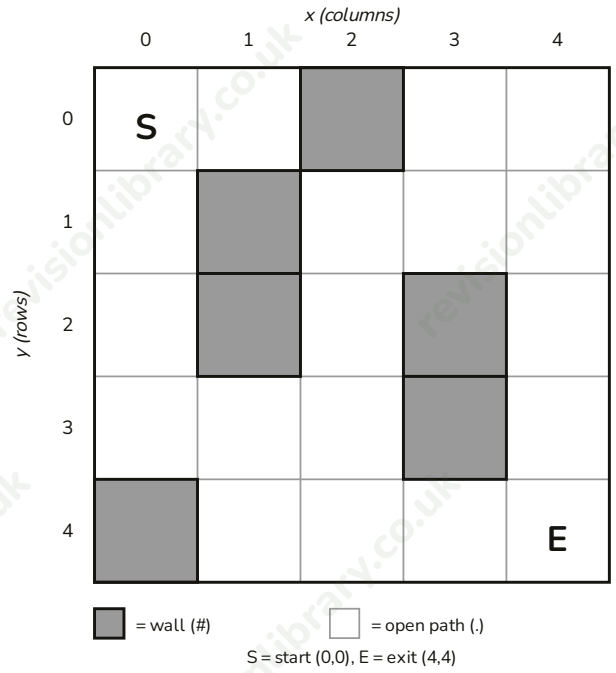
(Total for Question 2 is 12 marks)

© Revision Library | revisionlibrary.co.uk

3 Zainab is programming a robot to escape a small square maze on a 5 by 5 grid. Columns are numbered x = 0 to 4 (left to right) and rows are numbered y = 0 to 4 (top to bottom). The robot starts at S (0,0) and must reach the exit E (4,4). The function below searches for a path using backtracking, trying directions in the fixed order right, down, left, up:

```
FUNCTION FindPath(x, y)
  IF NOT ValidMove(x, y) THEN
    RETURN FALSE
  ENDIF
  IF grid[x][y] = "E" THEN
    RETURN TRUE
  ENDIF
  MarkVisited(x, y)
  IF FindPath(x + 1, y) THEN
    RETURN TRUE
  ENDIF
  IF FindPath(x, y + 1) THEN
    RETURN TRUE
  ENDIF
  IF FindPath(x - 1, y) THEN
    RETURN TRUE
  ENDIF
  IF FindPath(x, y - 1) THEN
    RETURN TRUE
  ENDIF
  RETURN FALSE
ENDFUNCTION
```

ValidMove(x, y) returns TRUE only if (x, y) is inside the grid, is not a wall, and has not already been visited during this search.



(a) Explain what the final line, RETURN FALSE, allows the algorithm to do, and state which computational thinking technique this line is an example of. **(2)**

(b) Trace the algorithm starting at S(0,0). State the sequence of coordinates visited up to and including the point at which the first backtrack occurs, and state the coordinate to which the algorithm backtracks. **(4)**

(c) Continuing from the backtrack in part (b), state the full sequence of coordinates that leads the algorithm to the exit E, following the same priority order (right, down, left, up) at each step. **(4)**

(Total for Question 3 is 10 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 2.1 Elements of Computational Thinking: Test Pack

Question 1

(a) identifies that abstraction involves removing or hiding unnecessary detail from a problem (oe) **B1**

(a) identifies that only the detail essential to solving the problem in hand is retained/kept (oe) **B1**

(a) Answer: Abstraction is the process of removing unnecessary detail from a problem so that only the information essential to solving it is retained.

(b) identifies that decomposition involves breaking a large/complex problem down (oe) **B1**

(b) identifies that it is broken into smaller sub-problems that are easier to solve/manage individually (oe) **B1**

(b) Answer: Decomposition is the process of breaking a large, complex problem down into a number of smaller sub-problems, each of which is easier to solve or manage on its own.

(c) valid sub-problem 1, e.g. searching the catalogue for a book by title/author/ISBN **B1**

(c) valid sub-problem 2, e.g. placing/managing a reservation on a book that is on loan **B1**

(c) valid sub-problem 3, e.g. detecting a reserved book's return and notifying the member **B1**

(c) valid abstracted detail linked to one stated sub-problem, e.g. the exact method of sending the notification (email/SMS/app) can be ignored at this stage, as only the requirement to notify the member matters (oe, ft) **B1**

(c) Answer: e.g. (1) catalogue search, (2) reservation handling, (3) notification on return; abstracted detail: how the notification is delivered (email/SMS/app) can be ignored at this stage.

Question 2

(a) identifies that while one instruction is at one stage, a different instruction can be at another stage at the same time, so stages are not left idle (oe) **B1**

(a) identifies that this increases the number of instructions completed per unit of time / reduces the overall time to complete multiple instructions (oe) **B1**

(a) Answer: While one instruction is being decoded, the next instruction can be fetched at the same time (and so on for other stages), so pipeline stages are not left idle; this increases the number of instructions completed per unit time.

(b) correct method: total cycles = number of instructions x number of stages = 10 x 4 **M1**

(b) correct cycles = 40, multiplied by the clock cycle time of 2 ns **A1**

(b) cao 80 ns **A1**

(b) Answer: 80 ns

(c) correct method: total cycles = number of stages + (number of instructions - 1) = 4 + 9 **M1**

(c) correct cycles = 13, multiplied by the clock cycle time of 2 ns **A1**

(c) cao 26 ns **A1**

(c) Answer: 26 ns

(d) correct method: speed-up = time without pipelining (part b) divided by time with pipelining (part c) **M1**

(d) cao awrt 3.08 **A1**

(d) Answer: 3.08

(e) identifies a pipeline hazard as a situation that prevents the next instruction from executing in its designated clock cycle as planned (oe) **B1**

(e) valid example, e.g. a branch instruction whose outcome is not known until the Execute stage, so the processor may have already fetched/decoded the wrong following instructions (control hazard); or an instruction that needs a result from a previous instruction that is not yet available (data hazard) **B1**

(e) Answer: A pipeline hazard is a situation that prevents the next instruction executing in its planned clock cycle, e.g. a branch instruction whose outcome is unknown until the Execute stage, meaning instructions fetched after it may have to be discarded and the pipeline stalled.

© Revision Library | revisionlibrary.co.uk

Question 3

(a) explains that it allows the algorithm to undo the last move and return control to the previous call so a different direction can be tried (oe) **B1**

(a) identifies backtracking as the computational thinking technique **B1**

(a) Answer: It allows the search to backtrack: control returns to the previous call of FindPath so that a different, untried direction can be attempted. This is an example of backtracking.

(b) correctly identifies (1,0) as the first cell visited after S (moving right) **B1**

(b) correctly identifies that from (1,0), right and down are both blocked by walls **B1**

(b) correctly identifies that left is already visited and up is out of the grid, so (1,0) is a dead end **B1**

(b) correctly states that the algorithm backtracks to (0,0) to try the next direction (down) **B1**

(b) Answer: Visited: (0,0), (1,0). (1,0) is a dead end (right and down are walls, left is visited, up is out of the grid), so the algorithm backtracks to (0,0).

(c) correctly continues (0,0) - (0,1) - (0,2) - (0,3) after the backtrack (down is tried first each time, as other options are blocked) **B1**

(c) correctly continues (1,3) - (2,3), moving right along row y=3 **B1**

(c) correctly identifies that right from (2,3) is blocked, so the path continues down to (2,4) **B1**

(c) cao completes the path (3,4) - (4,4), reaching E **B1**

(c) Answer: (0,0) - (0,1) - (0,2) - (0,3) - (1,3) - (2,3) - (2,4) - (3,4) - (4,4) = E

© Revision Library | revisionlibrary.co.uk