|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | For Examiner’s Use | | | Examiner’s Initials | | | Question | Mark | | **1** | 2 | | **2** | 10 | | **3** | 11 | | **4** | 4 | | **5** | 13 | | **6** | 12 | | **7** | 2 | | **8** | 5 | | **9** | 9 | | **10** | 9 | | **11** | 12 | | **TOTAL** | 77 |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Centre number** |  |  |  |  |  | **Candidate number** | 0 | 2 | 2 | 6 | | **Surname** | Deal | | | | | | | | | | | **Forename(s)** | James | | | | | | | | | | | **Candidate signature** |  | | | | | | | | | | | **Programming Language** | **Python (7517/1D)** | | | | | | | | | |   **A-LEVEL COMPUTER SCIENCE** |
| **Paper 1** |

Monday 11 June 2018

Morning

Time allowed: 2 hours 30 minutes

Instructions

* This is the Electronic Answer Document (EAD). Answer **all** questions by entering your answers into this document on screen. You **must save** this document at regular intervals.
* Before the examination begins, type the information needed in the boxes **at the top of this page**.
* Before the examination begins, type the information needed in the boxes **in the footers** (page 2 onwards) of this EAD.

**During the examination**

* You may print pages of your EAD. A print monitor will collect and deliver your print-out to you. You must **not** collect your own print-out.

**Exceptions**

* If you experience difficulty inserting screen shots into your EAD then you may print these separately and attach them to the back of the EAD with a reference in the correct place in the EAD. Ensure that your **Centre Number**, **Candidate Name** and **Candidate Number** are on each sheet.

**At the end of the examination**

* Save for the last time and print your EAD **on one side only** (not double-sided). A print monitor will collect and deliver your print-out to you. Check that your details are in the footers of every page. Write them in if they are not.
* Enter your signature on the front cover.
* Staple or tie all pages together in the top left-hand corner of the EAD.
* Hand in **all** pages of the EAD to the Invigilator.

**Warning**

* No extra time is allowed for printing and collating.
* It may not be possible to credit an answer if your:

– details are not printed on every page as instructed above

– screen captures are not legible to the Examiner.

Answer **all** questions.

You **must save** this document at regular intervals or you may lose your work.

**Section A**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 01** | | | | |  |  |
| **0** | **1** | **.** | **1** | As you know that ALL the boxes have been labelled incorrectly, we can work out which box contains what by taking one vegetable out of the box labeled ‘onions and carrots’ by doing so we can determine whether it has onions or carrots as it cannot contain both due to being definetly incorrectly labelled. We can then place the correct label onto that box and put the onion and carrots label onto the box we took the label from. Finally we swap the two remaining boxes we do not know as the other box is also definetly incorrectly labelled so we know we need to swap the remaining two to make them correct. | | 2/2 |
|  |  |  |  |  | | **2** |
|  |  |  |  |  | | **2** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 02** | | | | |  |  |
| **0** | **2** | **.** | **1** | 53- | | 1/1 |
| **0** | **2** | **.** | **2** | 342\*+- | | 2/2 |
| **0** | **2** | **.** | **3** | Reverse Polish Notation can be executed by a computer simply as it’s how the ALU works. | | 2/2 |
| **0** | **2** | **.** | **4** | A stack could be used to evaluate a reverse polish expression by parsing the expression. If we reach an operand we push it to the stack. If we reach an operator, we also push it to the stack. However we then pop out the top three elements in the stack. These are then evaluated bottom to top. For example if it were a 2, 7, and a + we would evaluate to 9 and then push this result back into the stack. We continue this process until there is only a single value remaining in our stack and there should be no operators left. We pop this final value out which is our evaluated result. | | 3/3  2/2 |
| **0** | **2** | **.** | **5** | Stack frame components:  1) return address.  2) local variable states. | |
|  |  |  |  |  | | 10 |
|  |  |  |  |  | | **10** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 03** | | | | |  |  |
| **0** | **3** | **.** | **1** | Adjacency (distance) matrix:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 0 | 2 | 5 | 3 | 0 | 8 | | 2 | 0 | 1 | 0 | 0 | 0 | | 5 | 1 | 0 | 0 | 0 | 4 | | 3 | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 0 | 1 | 0 | 5 | | 8 | 0 | 4 | 0 | 5 | 0 | | | 2/2 |
| **0** | **3** | **.** | **2** | An adjancey list is appropriate when there are few edges between vertices. | | 1/1 |
| **0** | **3** | **.** | **3** | As it contains a cycle which cannot be represented in a tree. | | 1/2 |
| **0** | **3** | **.** | **4** | A graph with weights/distances on it’s edges between nodes | | 1/1 |
| **0** | **3** | **.** | **5** | |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 2 |  | 2 |  |  |  |  |  | 1 |  |  |  |  | | 5 |  |  | 5 |  |  |  |  |  | 1 |  |  |  | | 3 |  |  |  | 3 |  |  |  |  |  | 1 |  |  | | 8 |  |  |  |  |  | 8 |  |  |  |  |  | 1 | | 3 |  |  | 3 |  |  |  |  |  | 2 |  |  |  | | 7 |  |  |  |  |  | 7 |  |  |  |  |  | 7 | | 4 |  |  |  |  | 4 |  |  |  |  | 4 |  |  | | 9 |  |  |  |  |  |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  | | | 5/7  digit in red should be a 3 |
| **0** | **3** | **.** | **6** | Shortest distance from node 1 to 6 | | 1/1 |
| **0** | **3** | **.** | **7** | Stores the current node we are parsing – Used to store previous node which allows the path to be recreated. | | 0/2 |
|  |  |  |  |  | | 11 |
|  |  |  |  |  | | **16** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 04** | | | | |  |  |
| **0** | **4** | **.** | **1** | The halting problem is where a computer predicts whether a program will stop or not stop without executing it. | | 2/2 |
| **0** | **4** | **.** | **2** | A computer cannot predict whether a program will halt without running it  - as the problem is non-computable / inspection alone cannot determine whether any algorithm will halt | | 0 |
| **0** | **4** | **.** | **3** | Components of a turing machine:  - read/write head  - a set of transistion rules | | ½  ½  0/1 |
| **0** | **4** | **.** | **4** | A universal turing machine is a turing machine which can emulate another turing machine within itself. - desciription of this emulated TM is stored on the universal turing machine’s tape. The UTM acts as an interpreter. | |
| **0** | **4** | **.** | **5** | As it has unlimited processing power theoretically.  - as it has an infinite amount of MEMORY / tape | |
|  |  |  |  |  | | 4 |
|  |  |  |  |  | | **8** |

**Section B**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 05** | | | | |  |  |
| **0** | **5** | **.** | **1** |  | | 12/12 |
| **0** | **5** | **.** | **2** |  | | 1/1 |
|  |  |  |  |  | | 13 |
|  |  |  |  |  | | **13** |

**Section C**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 06** | | | | |  |  |
| **0** | **6** | **.** | **1** | UpdateAfterAllowedWord()  - BUILT-IN FUNCTION - len() | | 0/1 |
| **0** | **6** | **.** | **2** | Item | | 1/1  ½  1/1  1/1  4/4  4/4 |
| **0** | **6** | **.** | **3** | Saves memory space as it has a set size.  - A linear queue could have locations avaliable that are not in use / wasted space.  - Items in a circular queue are all shuffled forward when an item is deleted from the front of the queue. Therefore no space wasted. | |
| **0** | **6** | **.** | **4** | Size of the queue is small. | |
| **0** | **6** | **.** | **5** | frontOfQueueIndex | |
| **0** | **6** | **.** | **6** | From the lists for the second and third if statements in createTileDictionary() I would remove the items: 1,2,3,5,6 and add them to the list for the first if statement. This would make the possibility of getting a tile worth 1 point the same as getting any of the other tiles combined. | |
| **0** | **6** | **.** | **7** | A string consisting of lowercase characters could be converted to upercase by having a for loop which loops for each character in the string. We could have previously defined a dictionary of every alphabet letter which has a lowercase character as the key and the upercase respective character as the value. We should also have set a output variable as an empty string before the loop. Then in our for loop we can pass the current character we’ve iterated to into the dictionary and set the value of that key to be appended to our output string via concatenation. Once the loop ends we should imediately afterwards return the output string which is now upercase. | |
|  |  |  |  |  | | 12 |
|  |  |  |  |  | | **14** |

**Section D**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 07** | | | | |  |  |
| **0** | **7** | **.** | **1** |  | | 1/1 |
| **0** | **7** | **.** | **2** |  | | 1/1 |
|  |  |  |  |  | | 2 |
|  |  |  |  |  | | **2** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 08** | | | | |  |  |
| **0** | **8** | **.** | **1** |  | | 4/4 |
| **0** | **8** | **.** | **2** |  | | 1/1 |
|  |  |  |  |  | | 5 |
|  |  |  |  |  | | **5** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 09** | | | | |  |  |
| **0** | **9** | **.** | **1** |  | | 8/8 |
| **0** | **9** | **.** | **2** |  | | 1/1 |
|  |  |  |  |  | | 9 |
|  |  |  |  |  | | **9** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 10** | | | | |  |  |
| **1** | **0** | **.** | **1** |  | | 8/8 |
| **1** | **0** | **.** | **2** |  | | 1/1 |
|  |  |  |  |  | | 9 |
|  |  |  |  |  | | **9** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | *Leave blank* |
| **Question 11** | | | | |  |  |
| **1** | **1** | **.** | **1** |  | | 11/11 |
| **1** | **1** | **.** | **2** |  | | 1/1 |
|  |  |  |  |  | | 12 |
|  |  |  |  |  | | **12** |

**If you have made any separate screen captures, put a reference in the correct place in this Electronic Answer Document.**

**Ensure that your Centre Number, Candidate Name and Candidate Number are on each sheet.**

**Attach them to the back of this document.**

NB: The AQA logo is provided for use only in the Electronic Answer Document for the 2018 A-level Computer Science Paper 1 (7517/1) examination and must not be used for any other purpose.

Copyright © 2018 AQA and its licensors. All rights reserved.