2017 VCE Algorithmics (HESS) examination report

General comments

The 2017 Algorithmics (HESS) examination contained two sections. Section A was composed of 20 multiple-choice questions and Section B was composed of 16 short-answer and extended-answer questions. Students achieved scores across the range of available marks, with higher-scoring students demonstrating an impressive grasp of challenging content.

The majority of students were able to attempt most questions on the examination, demonstrating good time management.

Questions 1, 11c. and 12b. required students to identify an appropriate abstract data type (ADT) as part of outlining or explaining an algorithm. It is crucial that students take care to clearly articulate how their chosen ADT represents their specific problem. In answering Question 1, for instance, it was important that students stated that the pieces of glass were to be represented as nodes in a graph, and their adjacency as edges. This is also good practice for questions that do not explicitly ask for the ADT but require a method to solve an unfamiliar problem (for example, Questions 15 and 16). It is significantly more difficult to describe a method well where the underlying representation is unclear.

Where a question specifies multiple aspects of a given problem to be addressed, it is incumbent on students to engage with each of these aspects in some order. To that end, students are encouraged to use dot points or shorter sentences with clear linking language to help articulate their thoughts, and proofread their answers to ensure that each of the aspects of the question is addressed by one of their dot points or sentences.

Students were able to write algorithms in pseudocode in response to unseen contexts (Questions 3 and 8) and identify errors and correct them in an existing algorithm (Questions 11a. and 11b.). Some attempts adhered too closely to the syntax of a programming language. This is not necessary and can impose some unnecessary constraints on the students' thinking. Where existing functions are given (Question 3), students are strongly encouraged to use them.

When discussing the running times or efficiency of algorithms, it is important that students can confidently use Big-O notation to make comparisons.

Finally, students are reminded to be vigilant about reading the current year's study design and any notices carefully, as it was evident that some students were still working with concepts from earlier editions of the study design.

Specific information

Note: Student responses reproduced in this report have not been corrected for grammar, spelling or factual information.

This report provides sample answers or an indication of what answers may have included. Unless otherwise stated, these are not intended to be exemplary or complete responses.

The statistics in this report may be subject to rounding resulting in a total more or less than 100 per cent.

Section A – Multiple-choice questions

Question	% A	% B	% C	% D	% No Answer	Comments
1	50	15	32	4	0	It is important for students to understand that Big-O notation describes an upper bound, and so is only used for analysis of worst-case running times.
2	6	75	6	13	0	
3	0	1	98	1	0	
4	1	99	0	0	0	
5	1	95	2	2	0	
6	3	1	2	94	0	
7	11	58	11	20	0	Lengthening the tape guarantees that *M*, run with *T*, still terminates with an identical answer because the additional parts of tape will never be seen by the head of *M*.
8	0	5	91	5	0	
9	22	3	58	17	0	The computability of a function uses the idea of an idealised human.
10	2	13	63	22	0	
11	5	3	6	87	0	
12	17	17	57	9	0	While other answers meet some of the requirements, breadth-first search is the only algorithm that meets all requirements.
13	3	85	2	10	0	
14	7	52	11	29	0	To analyse $\frac{f(n)}{g(n)}$, students should consider what happens when n becomes increasingly large; namely, both the 6 in $f(n)$ and the $\log n$ in $g(n)$ become negligible relative to $4n^2$.
15	4	15	12	70	0	
16	88	2	7	2	1	
17	12	85	2	1	0	
18	91	1	4	5	0	

Question	% A	% B	% C	% D	% No Answer	Comments
19	6	74	16	5	0	On review, it was acknowledged that this question was ambiguous, so students were awarded a mark for any response.
20	94	1	1	5	0	

Section B

Question 1

Marks	0	1	2	3	Average
%	14	28	32	27	**1.7**

Responses were required to describe how the polygons and their adjacency could be represented using an appropriate ADT (for example, a graph ADT) and to outline an algorithm that could reasonably be used to solve the problem. For full marks, responses needed to include a discussion of precisely how that particular algorithm would solve the problem using language specific to the ADT.

While most responses received at least some marks, students are reminded that they must make their understanding explicit in their solution – some responses merely named an appropriate algorithm, while others hinted at a graph representation but did not elaborate.

Students are advised to avoid using pseudocode in a question that asks them to 'outline' an algorithm.

Question 2a.

Marks	0	1	2	3	Average
%	22	35	28	16	**1.4**

Appropriate design patterns included brute force and divide-and-conquer. High-scoring responses clearly described the manner in which the given design pattern would generate the set of all possible words from a given input by describing how the algorithm would iterate over each character or vowel.

The following is an example of a high-scoring response.

> *Brute force could be used. A tree traversal could be used where each node represents a semi-complete permutation of the word. Each node would have a child connected by an edge to the words that could be made by changing the next vowel in the word. This repeats until all words possible are generated and the leaves (words with all vowels marked) can be taken as output.*

Question 2b.

Marks	0	1	2	3	Average
%	14	32	35	19	**1.6**

The modification requiring the generation of all possible sentences in addition to all possible words increases the size of the output from 3^v where *v* is the number of vowels to $3^v \times w!$ where *w* is the number of words in the sentence. This may make the problem intractable due to a combinatorial explosion and may make the problem unsolvable if the number of words is sufficiently large.

It was evident that some students would benefit from revising factorials and combinatorics, as a number of students made a reasonable conceptual attempt but did not seem to have the mathematics with which to ground their answer.

Question 3

Marks	0	1	2	3	4	5	6	Average
%	14	12	12	14	6	13	30	3.5

The following is a possible solution using an iterative approach.

```
Algorithm IsUnlocked

      For a = 0 to 9

           set(0, a)
           For b = 0 to 9
                set(1, b)
                For c = 0 to 9
                     set(2, c)
                     If unlock() is TRUE
                          return (a, b, c)
     return "Not Found"
```

This question was answered well, with most students showing an understanding that some nested loops were necessary for an iterative approach and that dials would need to be changed inside the loops. Common sources of error included not returning the correct combination (for example, returning "FOUND!" instead), not checking whether the lock was unlocked at any stage and not iterating over all possible combinations. Some responses seemingly ignored the functions given in the questions, often leading to unnecessarily complicated approaches.

There were a few students who attempted a recursive solution, often with a good deal of success.

Question 4

Marks	0	1	2	3	4	Average
%	32	19	12	24	13	1.7

Algorithms run in sequence will have a total running time equal to the sum of their individual running times – for example, $O(n^2) + O(n) + O(logn) = O(n^2 + n + logn) = O(n^2)$. Algorithms that are nested will have a total running time equal to the product of their individual running times – for example, $O(n^2) \times O(n) \times O(logn) = O(n^3 logn)$. As a result, for the same starting algorithms, running them in a nested fashion will significantly increase the time complexity relative to running them in sequence.

While many students were able to answer parts of this question, there were few responses that addressed the entirety of the question. Some otherwise high-scoring responses did not make a comparative statement, while others omitted examples.

Question 5

Marks	0	1	2	3	Average
%	17	32	38	14	1.5

The Halting Problem asks, given a program *P* and input *I*, will *P* halt on *I*? It would be decidable if there existed a general algorithm that would be able to solve the Halting Problem in a finite number of steps; however, Turing showed that no such algorithm can exist, therefore the Halting Problem

is undecidable. This implies that automatic program verification is not possible in the absolute sense, as most programs are reducible to the Halting Problem.

While many students were able to discuss some elements of the Halting Problem and the fact that it was undecidable, few were able to subsequently discuss the implication of that on program verification.

There were many responses that showed a lack of understanding of the Halting Problem itself, and instead attempted to prove its undecidability using a by-contradiction argument. This was not necessary and was rarely done correctly.

Question 6

Marks	0	1	2	3	4	5	6	Average
%	23	18	23	17	12	6	2	2.1

A range of responses was seen for this question, with the following elements common to most high-level responses.

- One of the goals of Hilbert's program was to show that all mathematics was decidable: in other words, to show that any statement of first-order logic could be methodically verified as universally valid or not valid.
- Church's and Turing's work on lambda calculus and Turing machines, respectively, came about as a consequence of needing to precisely define what was meant by 'methodically'.
- Turing defined the Halting Problem and showed that it is undecidable, demonstrating that there existed a class of undecidable problems and thereby demonstrating that mathematics as a whole could not be decidable.
- Out of Church and Turing's work came the Church-Turing Thesis as Turing noticed that their computational formalisms were equivalent.

While most students were able to engage with some parts of the question, few addressed the question in its entirety. Common omissions included not addressing the given prompt of why the program was a failure and not demonstrating any link between Hilbert's work and Church and Turing's work.

A few students conflated Hilbert's program with a computer program. It is worth noting that Hilbert, Church and Turing were all working primarily in a time that predates digital computers.

The following is an example of a high-scoring response.

> *Hilbert believed that mathematics should have axioms from which all mathematical truths can be discovered while remaining consistent. He also suggested that it should be decidable meaning that all statements can be verified algorithmically. This led to the Hilbert program which attempted to find these axioms. This led on to Church and Turing's work which attempted to determine what was computable. This led to Turing and Church's individual ideas known as the Turing Machine and Church's lambda calculus in order to determine what can be computed. These two attempted to answer this, one with a hypothetical machine and another with mathematics. Eventually, these two ideas were combined to form the Church-Turing thesis which determined the upper limits of computability by stating that all problems that are unsolvable via a Turing machine or lambda calculus can't be solved by a computer. This is how Hilbert has influenced them.*

Question 7a.

Marks	0	1	2	Average
%	25	27	48	**1.3**

This question was answered well, although students are reminded to pay careful attention to the command terms given in the question. Where asked to 'describe', it is insufficient to simply state the limit.

The following is an example of a high-scoring response.

> *Many problem is intractable because the amount of resources required for solving them is too large to be practical, even though an algorithm may exist for solving them, such as problems in NP-hard.*

Question 7b.

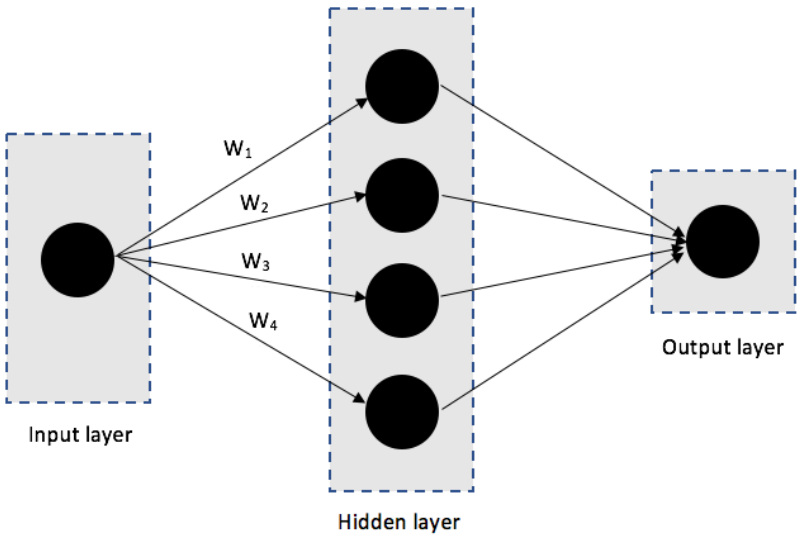
Marks	0	1	2	3	Average
%	50	29	16	5	**0.8**

Neural networks may simplify a complex problem and narrow the focus to two variables (quantised): input and output. No direct implementation of an algorithm needs to be created as the neural network is trained on input data. For instance, it may be very difficult to write a direct algorithm for facial recognition, but given sufficient training data a neural network can overcome this limitation.

This question was not answered well, with many students incorrectly referring to parallel processing or reducing time complexity of a problem as advantages of neural networks. Some students left this question blank, demonstrating a lack of familiarity with the topic. Students should note that they are required to study **all** of the 'Alternative methods of computation' listed on the VCAA Algorithmics (HESS) approved lists webpage.

Question 7c.

Marks	0	1	2	3	Average
%	31	23	36	10	**1.3**



High-scoring responses included the following characteristics:

- input(s) and output(s) and one or more hidden layers
- directed edges between inputs and neuron layer and directed links from neuron layer to the output
- some annotation of weights on the edges.

Question 8a.

Marks	0	1	2	3	4	5	Average
%	21	14	19	14	22	11	**2.4**

There were many high-scoring responses to this question. Some responses, like the student example below, sorted the final list themselves, while others used a sort function after adding all the children's ages to a common list.

Some lower-scoring responses exhibited no understanding of how to find a median, with some finding the mean instead. The concept of a median is fundamental to many applications in Algorithmics (HESS), and students must ensure they know it well.

The following is an example of a high-scoring response.

```
Input: room 1; list of children's age with length n, room 2; list of
       children's age with length n+1
Output: median age of all rooms
j <- 0
k <- 0
finalList <- List
while j < n or k < n+1
      if jth childage in room 1 ≤ kth child age of room 2
            append jth child age to finalList
            j +=1
      else
            append kth child age to finalList
            k +=1
return finalList[n]
```

Question 8b.

Marks	0	1	2	Average
%	54	24	22	**0.7**

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + O(1) & if \min(n_1, n_2) > 0 \\ O(1) & otherwise \end{cases}$$

This question was not answered well, with common errors including omitting the base case and writing $T(n) = 2T\left(\frac{n}{2}\right)$ as the recurrence relation.

Question 8c.

Marks	0	1	2	3	Average
%	20	39	20	21	**1.4**

The following is an example of a high-scoring response.

I would recommend the search algorithm as it uses 2 questions to halve the range the median could be in while the brute force finds the median in 2n+1 times. Therefore the search has time complexity of $O(logn)$ while brute force has $O(n)$ making search more efficient.

Question 9

Marks	0	1	2	3	4	5	Average
%	34	25	13	8	11	9	**1.7**

This question proved challenging for many students. Students are encouraged to take note of the following.

- Proofs by contradiction require a specific structure: they begin by assuming the opposite to what is being proven, and proceed logically until they reach a contradiction, thereby invalidating the original assumption and hence proving its opposite. The structure of a proof by contradiction (or, equally, a proof by induction) is central to its function.
- The use of dot points, short sentences, diagrams and clear linking language (hence, therefore, since, etc.) is highly encouraged.
- It is crucial to clearly define any terminology or labels that will be used throughout the proof. Some responses used the same label – for example, (u,v) – for two different edges, making it impossible for the proof to be cogent.

The following is an example of a high-scoring response.

Suppose that the smallest weighted edge (u,v) is not in the minimum spanning tree.

Since it is a tree, there must exist a path between u and v, not via this edge. Let this path be P.

As this is a minimum spanning tree, any edge (a,b) in P must be the minimum edge connecting the two sets of nodes on its two sets.

But edge (a,b) cannot weigh less than the smallest weighted edge (u,v), which can also connect the two sets of nodes.

This is a contradiction, and hence the initial assumption must be wrong.

*Hence (u,v**)** must be in any minimal spanning tree.*

Question 10a.

Marks	0	1	2	Average
%	65	17	18	**0.6**

The best case running time is $\Omega(N)$, which would occur if the initial *M*, *P* and *S* chosen resulted in his sister being the last child left in the circle and she wins the prize.

Many students incorrectly wrote that the best case running time would occur when *N* = 1. It is important to understand that best case running times refer to the best possible running time that can be achieved given some input size, *N*, and not the shortest running time that can be achieved by assuming some very small, specific *N* value.

Question 10b.

Marks	0	1	2	Average
%	65	16	19	**0.6**

Yes, because `GetWinner` needs to be run at most $M \times P \times S$ times, and $0 \leq P < N$ and $0 \leq S < N$, so the worst case running time will be $O(N^3 \times M)$.

Question 11a.

Marks	0	1	Average
%	20	80	**0.8**

Zero domes. The loop never runs, as it doesn't start out as empty.

This question was very well answered, with most students correctly identifying the error in logic.

Question 11b.

Marks	0	1	2	Average
%	9	14	77	**1.7**

```
while L is not empty
      Vaccinate all people in the first dome in L.
      Remove the first element in L.
```

This question was also very well answered, with most students demonstrating a good control of basic 'while' logic and list behaviour.

Where asked to correct an algorithm, students are encouraged to stick to the given structure and notation rather than rewriting the algorithm completely.

Question 11c.

Marks	0	1	2	3	Average
%	17	25	34	24	**1.7**

Many students scored either two or three marks for this question. Responses that did not score highly tended to state the ADT without any description of how it would represent the domes and their adjacency and did not have any way of returning the maximum number of days.

A few responses tried to describe how their graph ADT would be implemented (for example, an adjacency matrix or adjacency list). This is well outside the scope of the study.

The following is an example of a high-scoring response.

Graph ADT could be used.

Let node be each dome. Let an edge (u, v) exist if dome u is adjacent to dome v.

A breadth first search algorithm could be used, and then return the maximum depth of the tree generated by breadth first search.

Question 12a.

Marks	0	1	2	Average
%	22	28	50	**1.3**

A wide range of responses was accepted, with the most common response involving connecting a new player to another new player and then running the algorithm as normal, or connecting a new player to a random existing player and then running the algorithm as normal.

Question 12b.

Marks	0	1	2	3	Average
%	46	29	18	6	**0.9**

Many responses did not note the requirement to finish in a reasonable time, which should suggest the need to use a heuristic approach. Other responses did not explain how a graph ADT would represent their data, or omitted any discussion of how a stated algorithm would actually work on the given problem.

The following is an example of a high-scoring response.

> *A Breadth first search with heuristics could be used to find suggestions. Using breadth first search to search through friends before more distant acquantances and heuristics to determine whether the player has a good enough score to be recommended could be done in a reasonable time. This assumed that players are nodes linked to only their friends in a graph without cycles. i.e. shared friends are not shown in friends of friends.*

Question 13

Marks	0	1	2	3	Average
%	14	22	45	18	**1.7**

Most students showed a good understanding of the way a decrease-and-conquer algorithm reduces the search space, or identified the described algorithm as a standard binary search.

Some responses were not able to achieve full marks because they did not state a design pattern (although they did describe how it worked), while other responses did not engage with the specific requirement of fewer than 16 questions. Where the question makes such numerical specifications, students are required to use mathematics to substantiate their argument.

Question 14a.

Marks	0	1	2	Average
%	47	42	11	**0.7**

Starting from the depot, greedily go to the next closest delivery destination each time, until all packages have been returned, and then drive back to the depot.

This question described a standard Travelling Salesman Problem but it was not answered well. Common errors included ignoring the requirement to solve the problem quickly (and not necessarily optimally), not returning back to the depot, and using single-source shortest path algorithms.

Students are encouraged to view a greedy approach as a useful and easy heuristic that can be applied to many problems, and they should only resort to more advanced heuristic methods (for example, simulated annealing) where there is a good reason to do so.

Question 14b.

Marks	0	1	2	Average
%	62	22	16	**0.6**

Use brute force to generate all possible tours starting and ending at the depot, and select the shortest that was generated.

A simple brute-force approach sufficed because there was no stipulation to find the shortest route in a reasonable time.

Question 15

Marks	0	1	2	3	4	Average
%	43	28	16	10	3	**1.1**

The following is one possible method that could be used:

- Represent each arrangement of tiles as a state-in-a-state diagram
- Starting at Figure 1, recursively generate other states using a backtracking approach, backtracking whenever a previously reached state is found.
- If Figure 2 is generated, then backtrack all the way back to Figure 1, noting the path taken.
- If all possible states are generated and Figure 2 is not encountered, then it is not possible to rearrange the hard drive as desired.

This question proved challenging for students. Some responses attempted to treat each hard drive as a node, with little success. Other low-scoring responses broadly described a brute force approach, but made no attempt to describe the representation of the problem.

Question 16

Marks	0	1	2	3	Average
%	40	35	20	4	**0.9**

The starting state in the simulation (the 5 × 5 grid, the shark and the fish) can be used as the root of a game tree, and all subsequent possible moves can be generated in the game tree. In order to select the best possible move when it is its turn, the shark must consider which of its moves would minimise the distance from the fish – that is, of all child nodes to the current state, which child node would result in the lowest possible distance from the fish? As the fish moves randomly, moving to the square that minimises the average position of the fish's next move would suffice.

Many students attempted to describe the minimax algorithm and ignored the context. Others found it difficult to approach the requirement that the fish moves randomly rather than optimally. Where a question requires a standard algorithm to be modified in some way, students are encouraged to make a reasonable assumption and proceed logically from that assumption.