VCE Algorithmics (HESS): Performance descriptors

|  |
| --- |
| **ALGORITHMICS (HESS) UNIT 3 OUTCOME 2****SCHOOL-ASSESSED COURSEWORK** |
| **Performance descriptors** |
|  |
| **Unit 3****Outcome 2** **On completion of this unit the student should be able to define and explain algorithmic design principles, design algorithms to solve information problems using basic algorithm design patterns, and implement the algorithms.** | **DESCRIPTOR: typical performance in each range** |
| **Very low** | **Low** | **Medium** | **High** | **Very high** |
| Identifies limited elements of sequence, selection and repetition in a given algorithm.Manually executes a simple sequence of simple steps within pseudocode. | Interprets simple structured pseudocode with sequence, selection and simple iteration with minimal errors.Identifies some recursive, iterative, brute-force search design pattern and greedy design pattern features within pseudocode for an algorithm.Describes some elements of the concept of modularisation. | Interprets and manually executes pseudocode containing nested iteration and the use of ADTs.Describes the concepts of the recursive, iterative, brute-force search and greedy design patterns.Describes how the concept modularisation has been applied within a particular piece of pseudocode for an algorithm. | Interprets and manually executes pseudocode containing complex use of ADTs or simple recursion.Describes how the recursive, iterative, brute-force search or greedy design pattern have been applied within a particular piece of pseudocode for an algorithm. | Interprets and manually executes pseudocode containing complex use of ADTs and recursion.Completely and precisely describes how the recursive, iterative, brute-force search or greedy design pattern have been applied within a particular piece of pseudocode for an algorithm. |
| Designs simple algorithms and writes these in pseudocode with significant task scaffolding required, and the final algorithm is only an effective method for a trivial subset of problem instances.Identifies some algorithm design approaches. | Designs simple algorithms and writes these in pseudocode, with some task scaffolding. The algorithm is an effective method for a non-trivial subset of problem instances. Explains the principles of the brute-force search or greedy algorithm design patterns, utilising appropriate examples. | Designs and applies algorithms including use of iteration and writes these in pseudocode with minimal errors.Designs modular algorithms.Applies a given algorithm design pattern to design an algorithm to solve a problem. | Designs algorithms using iteration and recursion for problems that have a structure that does not allow for the direct application of one of the studied algorithms. Explains the attributes required of problems for one of the algorithm design patterns to be applied. | Designs algorithms using iteration, recursion and non-trivial functions for problems that have a structure that does not allow for the direct application of one of the studied algorithms. Selects suitable algorithm design patterns for solving information problems and applies the design patterns to design algorithms and find solutions.  |
| Names and states correctly the computational applications of most of the specified graph algorithms.States informally the input types of the specified graph algorithms. | Explains informally how some of the specified graph algorithms perform their computation and writes the approximate pseudocode for these. | Selects and explains graph algorithms.Selects a suitable graph algorithm to apply to solve a complex problem.States precisely the input types of the specified graph algorithms. | Executes, without error, any of the specified graph algorithms using manual techniques for complex graphs. | Justifies a selection of a suitable graph algorithm for solving a complex problem based on the properties and limitations of the algorithm.Explains in precise terms why any of the specified graph algorithms are not valid for some classes of graph or graphs with certain properties. |
| Identifies the first few nodes visited by either the breadth-first or depth-first search algorithm when applied to a decision tree. | Executes the breadth-first or depth-first search algorithm on a decision tree.  | Applies a given graph search method to a decision tree to solve a planning problem. | Selects a suitable graph search method and applies it to a decision tree to solve a planning problem. | Evaluates the relative advantages of different graph search methods for solving a planning problem. |
| Implements simple algorithms with sequential, conditional and iterative elements. | Implements simple iterative algorithms that utilise collection ADTs. | Implements graph traversal algorithms and simple recursive algorithms.Applies an implementation of a simple iterative algorithm that utilises ADTs to solve a particular problem instance. | Implements shortest-path graph algorithms.Applies an implementation of a graph traversal algorithm or simple recursive algorithm to solve particular problem instances. | Efficiently implements shortest-path graph algorithms and applies them to solve particular problem instances. |
| Limited and unstructured arguments given for correctness of graph algorithms. | Describes an argument for the correctness of a graph algorithm that considers only the correctness of a specific example. | Demonstrates the correctness of specified graph algorithms using induction and contradiction for some input cases. | Describes an argument for the correctness of one of the specified graph algorithms that considers the general case of the problem, but not all steps in the chain of argument are explained. | Describes a valid argument for the correctness of at least one of the specified graph algorithms using either the induction or contradiction methods. |

KEY to marking scale based on the Outcome contributing 50 marks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Very Low 1–10 | Low 11–20 | Medium 21–30 | High 31–40 | Very High 41–50 |