Unit D1 – Decision Mathematics

The examination

The examination will consist of one 1½ hour paper. The paper will contain about seven questions with varying mark allocations per question which will be stated on the paper. All questions may be attempted.

Candidates are expected to know any other formulae which might be required by the specification and which are not included in the booklet, *Mathematical Formulae including Statistical Formulae and Tables*, which will be provided for use with the paper. Questions will be set in SI units and other units in common usage.

Candidates are expected to have available a calculator with at least the following keys: +, −, ×, ÷, π, x^2, $\sqrt{x}$, $1/x$, x^y and memory. Calculators with a facility for symbolic algebra, differentiation and/or integration are not permitted.

Preamble

Candidates should be familiar with the terms defined in the glossary attached to this specification. Candidates should show clearly how an algorithm has been applied. Matrix representation will be required but matrix manipulation is not required. Candidates will be required to model and interpret situations, including cross-checking between models and reality.

SPECIFICATION	NOTES
1. Algorithms	
The general ideas of algorithms.	The order of algorithm is not expected.
Implementation of an algorithm given by a flow chart or text.	
Candidates should be familiar with (i) bin packing, (ii) bubble sort, (iii) quick sort, (iv) binary search.	When using the quick sort algorithm, the pivot should be chosen as the 'number' at the mid-point of the list.
2. Algorithms on graphs	
The minimum spanning tree (minimum connector) problem. Prim's and Kruskal's (Greedy) algorithm.	Matrix representation for Prim's algorithm is expected. Candidates will be expected to draw a network from a given matrix and also to write down the matrix associated with a network.
Dijkstra's algorithm for finding the shortest path.	
Planar and non-planar graphs. Planarity algorithm for graphs with a Hamiltonian cycle.	Candidates should know that K_5 and $K_{3,3}$ are non-planar. Kuratowski's theorem is not required.

3. The route inspection problem

Algorithm for finding the shortest route around a network, travelling along every edge at least once and ending at the start vertex. The network will have up to four odd nodes.	Also known as the 'Chinese postman' problem. Candidates will be expected to consider all possible pairings of odd nodes. The application of Floyd's algorithm to the odd nodes is not required.

4. Critical path analysis

Modelling of a project by an activity network, including the use of dummies.	A precedence table will be given. Activity on edge will be used.
Algorithm for finding the critical path. Earliest and latest event times. Earliest and latest start and finish times for activities. Total float. Gantt (cascade) charts. Scheduling.	

5. Linear programming

Formulation of problems as linear programs.	
Graphical solution of two variable problems using ruler and vertex methods. Consideration of problems where solutions must have integer values.	
The Simplex algorithm and tableau for maximising problems.	Problems will be restricted to those with a maximum of three variables and three constraints, in addition to non-negativity conditions.
The use and meaning of slack variables.	

6. Matchings

Use of bipartite graphs for modelling matchings. Complete matchings and maximal matchings.	Candidates will be required to use the maximum matching algorithm to improve a matching by finding alternating paths. No consideration of assignment is required.
Algorithm for obtaining a maximum matching.	

7. Flows in networks

Algorithm for finding a maximum flow. Cuts and their capacity.	Vertex restrictions are not required. Only networks with directed edges will be considered. Only problems with upper capacities will be set.
Use of max flow – min cut theorem to verify that a flow is a maximum flow.	
Multiple sources and sinks.	

Glossary for D1

1. Algorithms

$[x]$ is the smallest integer which is greater than or equal to x, for example $[10.5] = 11$ and $[11] = 11$. In a list containing N names the 'middle' name has position $[\frac{1}{2}(N + 1)]$ so that if $N = 20$, this is $[10.5] = 11$ and if $N = 21$ it is $[11] = 11$.

2. Algorithms on graphs

A **graph** G consists of points (**vertices** or **nodes**) which are connected by lines (**edges** or **arcs**).

A **subgraph** of G is a graph, each of whose vertices belongs to G and each of whose edges belongs to G.

If a graph has a number associated with each edge (usually called its **weight**) then the graph is called a **weighted graph** or **network**.

The **degree** or **valency** of a vertex is the number of edges incident to it. A vertex is **odd** (**even**) if it has **odd** (**even**) degree.

A **path** is a finite sequence of edges, such that the end vertex of one edge in the sequence is the start vertex of the next, and in which no vertex appears more then once.

A **cycle** (**circuit**) is a closed path, ie the end vertex of the last edge is the start vertex of the first edge.

A cycle that passes through every vertex of a graph is called a **Hamiltonian cycle** and a graph in which a Hamiltonian cycle exists is said to be **Hamiltonian**.

Two vertices are **connected** if there is a path between them. A graph is **connected** if all its vertices are connected.

If the edges of a graph have a direction associated with them they are known as **directed edges** and the graph is known as a **digraph**.

A **tree** is a connected graph with no cycles.

A **spanning tree** of a graph G is a subgraph which includes all the vertices of G and is also a tree.

A **minimum spanning tree** (MST) is a spanning tree such that the total length of its edges is as small as possible. (This is sometimes called a **minimum connector**.)

A graph in which each of the n vertices is connected to every other vertex is called a **complete graph**. (The notation K_n will be used for such a graph with n vertices.)

A graph G is **planar** if it can be drawn in a plane in such a way that no two edges meet each other except at a vertex to which they are both incident.

4. Critical path analysis

The **total float** $F(i, j)$ of activity (i, j) is defined to be $F(i, j) = l_j - e_i -$ duration (i, j), where e_i is the earliest time for event i and l_j is the latest time for event j.

5. Linear programming

The **simplex tableau** for the linear programming problem:

Maximise $P = 14x + 12y + 13z$,
Subject to $4x + 5y + 3z \leq 16$,
$5x + 4y + 6z \leq 24$,

will be written as

Basic variable	x	y	z	r	s	Value
r	4	5	3	1	0	16
s	5	4	6	0	1	24
P	−14	−12	−13	0	0	0

where r and s are slack variables.

6. Matchings

A **bipartite graph** consists of two sets of vertices X and Y. The edges only join vertices in X to vertices in Y, not vertices within a set. (If there are r vertices in X and s vertices in Y then this graph is $K_{r,s}$.)

A **matching** is the pairing of some or all of the elements of one set, X, with elements of the second set, Y. If every member of X is paired to a member of Y the matching is said to be a **complete matching**.

7. Flows in networks

A **cut**, in a network with source S and sink T, is a set of arcs (edges) whose removal separates the network into two parts X and Y, where X contains at least S and Y contains at least T. The **capacity of a cut** is the sum of the capacities of those arcs in the cut which are directed from X to Y.

If a network has several sources S_1 , S_2 , . . ., then these can be connected to a single **supersource** S. The capacity of the edge joining S to S_1 is the sum of the capacities of the edges leaving S_1 .

If a network has several sinks T_1 , T_2 , . . ., then these can be connected to a **supersink** T. The capacity of the edge joining T_1 to T is the sum of the capacities of the edges entering T_1.