| Further Mathematics Year 12 | Working towards expected outcomes<br>Your child is not yet making the expected progress within this course. | Working at expected outcomes<br>Your child is achieving the expected progress for this point within the course. | Working beyond expected outcomes<br>Your child is exceeding the expected progress. |
|---|---|---|---|
| **Autumn Term** | Students working **towards** expected outcomes in Y12 can:<br>• Understand the meaning of an algorithm; trace simple text-based or flow-chart instructions step-by-step<br>• Identify vertices, edges, walks, paths and cycles in basic networks<br>• Translate a real-world situation into a linear programming model by defining decision variables, constraints and an objective function<br>• Shade and interpret the feasible region on a two-variable graph; read off its corner (vertex) points<br>• Build a simple precedence table and draw the corresponding activity network | Students working **at** expected in Y12 can:<br>• Apply bubble-sort and quick-sort to small lists, showing each pass and counting comparisons and swaps<br>• Solve bin-packing problems using first-fit and first-fit-decreasing methods; find and justify a lower bound for optimality<br>• Construct minimum spanning trees in networks using Prim's and Kruskal's algorithms, tracing each edge selection<br>• Use Dijkstra's algorithm to find the shortest path between two nodes in a network<br>• Solve linear programming problems graphically with the objective-line (ruler) and vertex methods, including integer-only cases | Students working **beyond** expected in Y12 can:<br>**In addition to the skills listed under 'Working At' for this topic, students working beyond expected outcomes can:**<br>• Analyse the order of growth of simple algorithms (e.g. why bubble-sort is $O(n^2)$ and how quick-sort improves on this)<br>• Prove the conditions for Eulerian and semi-Eulerian graphs; apply the route-inspection (Chinese postman) algorithm to networks with up to four odd vertices<br>• Conduct sensitivity analysis in linear programming, discussing how changes to constraints or objective coefficients affect the optimum and feasible region<br>• Analyse projects that have multiple critical paths, introduce dummy activities and perform resource levelling to minimise completion time |

| | | | |
|---|---|---|---|
| | | • Perform forward and backward passes in Critical Path Analysis to calculate earliest/latest event times, identify the critical path and compute total float<br><br>• Draw and interpret Gantt (cascade) charts, clearly marking critical activities | |

| **Spring Term** | Students working **towards** expected outcomes in Y12 can: | Students working **at** expected in Y12 can: | Students working **beyond** expected in Y12 can: |
|---|---|---|---|
| | • Define a discrete random variable and construct its probability distribution table<br>• Compute the mean and variance of a discrete random variable using $E[X] = \sum x\, p(x)$ and $Var(X) = E[X^2] - (E[X])^2$<br>• Recognise when to use the Binomial versus Poisson distribution and state their conditions of independence and constant rate<br>• Understand the language of hypothesis testing: null and alternative hypotheses, significance level, critical region and p-value<br>• Perform a one-sample Poisson test by setting up $H_0: \lambda = \lambda_0$, calculating probabilities and comparing to critical values<br>• Calculate expected frequencies for a goodness-of-fit test and combine classes when necessary | • Apply the Poisson distribution to calculate $P(X = k)$, cumulative probabilities, and use Poisson as an approximation to Binomial when $n\,p \leq 10$<br>• Work with linear transformations of random variables: compute $E[aX + b]$ and $Var(aX + b)$<br>• Use additivity of independent Poisson variables to model aggregated counts<br>• Carry out two-tailed Poisson hypothesis tests by halving the significance level and interpreting results in context<br>• Compute the chi-squared statistic $\sum(O–E)^2/E$, determine degrees of freedom and use tables or calculators for significance<br>• Perform contingency-table tests for independence with $(r–1)(c–1)$ degrees of freedom | • Critically evaluate the suitability of the Poisson model in real-world contexts, discussing independence and rate assumptions<br>• Combine multiple Poisson processes and derive the resulting mean and variance<br>• Tackle layered problem-solving questions requiring both Binomial and Poisson reasoning under timed conditions<br>• Estimate distribution parameters (e.g. p in a Binomial, λ in a Poisson) from data and adjust degrees of freedom in chi-squared tests<br>• Interpret borderline p-values, discussing practical versus statistical significance and types of error<br>• Solve multi-step inferential problems that combine Poisson testing and chi-squared analysis in one question |

| | | | |
|---|---|---|---|
| **Summer Term** | Students working **towards** expected outcomes in Y12 can:<br><br>• Recognise whether a graph is planar by spotting crossing edges and, when possible, redraw it without crossings<br><br>• Set up Floyd's algorithm on a small distance matrix and perform the first iteration to update shortest-path entries<br><br>• Identify vertices of odd degree in a network, pair up to four odd nodes, and carry out the basic route-inspection steps to find a closed walk<br><br>• State the Travelling Salesman problem clearly and apply the nearest-neighbour heuristic on a small complete network | Students working **at** expected in Y12 can:<br><br>• Use the route-inspection (Chinese postman) algorithm on networks with up to four odd-degree vertices to find the shortest closed walk<br><br>• Formulate both classical and practical Travelling Salesman problems on complete networks<br><br>• Calculate upper bounds for TSP solutions using MST-doubling and nearest-neighbour heuristics and lower bounds via the residual minimum spanning tree | Students working **beyond** expected in Y12 can:<br><br>• Extend the route-inspection algorithm to handle networks with more than four odd-degree vertices, optimally pairing odd nodes to minimise extra distance<br><br>• Discuss the NP-hard nature of the Travelling Salesman problem and explore heuristic and approximation strategies in depth<br><br>• Analyse the worst-case time complexity of advanced graph algorithms (including Floyd's and TSP heuristics), providing formal order-of-growth arguments<br><br>• Design and evaluate custom heuristics or approximation schemes for TSP and route-inspection, proving theoretical bounds or demonstrating empirical performance |