1. Didn't answer this one

2. (a) Evolutionary algorithms are driven by selection and variation.

   Selection aims to remove members of the population that have a low fitness, and retain individuals with a high fitness in the population. This can be achived through a variety of methods, but usually starts with scoring each individual against the value of an objective function, and then using a weighted random selection to determine which individuals to remove from the population (such as with a roulette wheel, stochastic ranking or a tournament). Essentially, selection ensures that solutions that are further away from minima are likely removed from the population, which directs the candidate solutions towards 'better' areas of the parameter space.

   Variation tries to ensure that each member of the population is different. If a population is homogenous, then there is little point having the population in the first place, but if each member of the population is different (especially if there multiple different 'species' in the population, where individuals from the different species score similarly highly against the objective function, but have different characteristics), then the solution space is likely better explored and the chance of finding a global minima is higher.

   (b) The main operators driving evolutionary algorithms are mutation and sexual reproduction.

   Mutation takes an existing candidate solution and changes it in some structured, but essentially random way. For example, if a candidate is represented as a chromosome with genes represented as floating point numbers, the Rechenburger and Schewfel algorithm mutates each gene by adding a random number to it, the magnitude of which is determined by a companion standard deviation gene (this is mutated alongisde the normal gene, but with a different mechanism).

   Sexually reproduction is essential so that variation between different members of the population can be combined to potentially produce fitter offspring. The exact method by which sexual reproduction is achieved differs depending on the model. When genes are represented as binary numbers (such as in Holland and DeJong's model), cross over is a good approach. If genes are floating point numbers like with Rechenburger and Schewfel's approach, then taking the average of each gene from the parents could be used.

   (c) Genetic programming differs in a few ways from a 'traditional' evolutionary algorithm. Rather than trying to find a set of parameters to minimise and objective function, genetic programming represents candidate solutions as programs and tries to find an optimal program instead.

   Henceforth, three parts of the evolutionary algorithm must be altered to support this goal; generation of solutions, mutation of solutions

and reproduction.

Generation of the tree can be done using either the full or the grow method. Here, a depth n is specified, and a tree of depth n (for the full method) or with a maximal depth of n (for the grow method) will be returned. Both methods select a random function node for the root, but the full method fills children of each node with more function nodes until the depth is reached at which point it uses terminals, while the grow method selects the children from the function nodes and the terminal nodes, which could lead the tree to being less deep.

Mutation is achieved by selecting a random node in a tree and replacing the sub-tree stemming from that node with a randomly generated subtree (generated with the same methods described above).

Reproduction is achieved using cross over, where a random node is picked from two parent trees and the nodes and their sub-trees are swapped to produce two offspring.

(d) Programmers generally work in three phases; sometimes they will develop new code, sometimes they will edit existing code and sometimes they will merge codebases together.

The generation of initial solutions is like the programmer writing some code for the first time without referencing other code.

Programmers editing existing code is akin to mutation changing a candidate solution. This analogy is even better if you consider that the progammer will often write a small portion of new code into a larger program, which is similar to mutation adding a smaller newly-generated subtree into an existing tree.

Finally, programmers combine bits of code (like using libaries) to make new code. This is similar to cross-over in genetic algorithms.

3. Didn't answer this one

4. (a) Scale free networks best capture the small world property. Scale free networks have a power law degree distribution (the probability that a given node would have a given number of incident edges) (graph would go here), which means that most nodes have a few links, but some nodes have lots of links. These nodes are termed hubs.

Scale free networks have a high clustering coeffieient (at least relative to other types of networks such as regular networks, small-world networks, etc) due to their hubs. The clustering coefficient is measured as the number of triangles connected to a node divided by the number of triples it is the center of, and aims to measure the density of the graph.

Scale free networks also have a short average path length (the average longest path from one node to another), which is indicative of the small world property. This is due to there being a high probability of nodes being connected to 'hub' nodes, which in turn, are connected

to many other nodes. Due to the high connectivity between nodes, the diameter of the network (the longest shortest path) is relatively low.

The properties of scale free networks stay more or less constant as the size of the network increases (hence the name 'scale-free') due to the homogenous topology throughout the graph (this is despite having a very heterogeneous distribution of nodes).

(b) The Barabasi-Albert (BA) model was developed in 1999 to produce scale-free networks using preferential attachment. In 2000, Barabasi developed a different model which took into account rewiring and in 2001, Barabasi developed constructs for modelling competition in the network.

The initial version of the algorithm starts with a small set of nodes and proceeds to add nodes to the network, with each new node being connected to m other nodes in the network using the following probability distribution (preferential attachment):

$$\Pi(k_i) = \frac{k_i}{\sum_{j=0}^{N} k_j}$$

This algorithm is very successful at generating sacle-free networks, though it has some flaws:

- The rich get richer (older nodes with lots of connections will get more connections as time goes on, new nodes don't really have a chance at becoming hubs).
- The algorithm is based only on structural features (i.e. only the degree of each node is considered when choosing where to put new links in the network).
- Evolution of the network is linear; existing nodes are not mutated at all (except by linking to new nodes).
- Not resistant to attack; by removing a hub node, whole regions of the graph could be isolated.

Barabasi's second model (2000) incorperated a rewriting strategy in order to mitigate flaw three (linear evolution of the network). Rewiring starts with all of the nodes isolated, and for each timestep does:

- Addd m new links to each node with a probability p. The node at the other end of the link is determined by:

$$\Pi(k_i) = \frac{k_i + 1}{\sum_{j=0}^{N} (k_j + 1)}$$

The $+1$ is required so that isolated nodes are linked.

- Once all of the initial links have been added, then for each node, m links are re-wired with a probability of q. The destination of the re-writing is determined using Π.
- Finally, with a probability of $1-(p+q)$, new nodes are added with m links using the probability distribution Π to determine the link destination.

The competition model (developed in 2001) further mitigates the flaws of the original. New nodes are given a chance to become hubs, and the number of edges incident on a node is not the only thing determining the node's propensity to acquire more edges (and become a hub). The key difference is that when each node is created, it is given a fitness $0 < s \leq 1$ taken from a uniform probability distribution, and Π is modified so that:

$$\Pi(k_i) = \frac{s_i k_i}{\sum_{j=0}^{N}(s_j k_j)}$$

Otherwise, this algorithm is the same as the rewiring one. Once these changes are incorperated, the only disadvantage of the BA model is that it is not resistant to attack if the hub nodes are removed from the graph.

5. Didn't answer this one

6. (a)

$$\sum_{i=0}^{k} \alpha_i y_{n+i} = h \sum_{i=0}^{k} \beta_i f_{n+i}$$

Note that α_k is always one.

The difference between explicit and implicit methods is that with an explicit method $B_k = 0$, while this is not true with implicit methods. This means that if the method is implicit, then the value of y_{n+k} must be found by solving an equation:

$$y_{n+k} = h\beta_k f(x_{n+k}, y_{n+k}) + \sum_{i=0}^{k-1} \beta_i f(x_{n+i}, y_{n+i}) - \sum_{i=0}^{k-1} \alpha_i y_{n+i}$$

(b) Formula for constants is:

$$C_0 = \sum_{i=0}^{k} \alpha_i$$
$$C_1 = \sum_{i=0}^{k} i\alpha_i - \sum_{i=0}^{k} \beta_i$$
$$C_n = \frac{1}{n!}\left(a_1 + \sum_{i=2}^{k} i^n \alpha_i\right) - \frac{1}{(n-1)!}\left(\beta_1 + \sum_{i=2}^{k} i^{n-1}\beta_i\right)$$

The order of convergence is p where $C_0 = 0, \ldots, C_p = 0$, $C_{p+1} \neq 0$. For a method to be consistent, its order must be greater than 1.

To determine the stability of the method, we must find the first characteristic polynomial:

$$P(\zeta) = \sum_{i=0}^{k} \alpha_i \zeta^i$$

For a method to be stable, the roots of this polynomial must not have roots with a modulus greater than one, and all of its roots with a modulus equal to one must be simple.

(c) i.

$$order(a) = \begin{cases} p, & \text{if } C_p = 0 \wedge c_{p+1} \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

Where:

$$C_0 = a + 1$$
$$C_1 = 3 + 2a - 1 = 2a + 2$$
$$C_n = \frac{1}{n!}(3^n + 2^n a) - \frac{1}{(n-1)!}\left(\frac{5}{12} - 2^{n-1}\frac{4}{3} + 3^{n-1}\frac{23}{12}\right)$$

ii. Stability is when the first characteristic polynomial has roots with a modulus less than or equal to one (and roots equal to one are simple).

$$p(\zeta) = \zeta^3 + a\zeta^2$$

In order to make the roots satsify the stability requirements, α must be $-1 \leq \alpha \leq 1$.