For 28 - 32 marks (70% - 80%) All of the above, and:

1. Setting up a `Hacker` with appropriate fields.
2. Modifying a `Client` so that, before it communicates, there is a 10% chance of it disconnecting rather than communicating.
3. Modifying a `Network` so that a record is kept of `Client` and `AccessPoint` history.
4. Modifying a `Network` so that any disconnected device are reconnected by performing another `handshake` between them

For 32 - 36 marks (80% - 90%) All of the above, and:

1. Implementing a means by which `Hacker` can recognise when two handshake packets, belonging to its `targetAccessPoint` and its `targetClient` are in a `Channel`, and thus from where to extract a key.
2. Using an extracted `key` to perform a `handshake` in a `Network` between a new `Client`, owned by `Hacker`, and the `targetAccessPoint`.
3. Modifying the running of a network so it does not just include `networkActivity` but also `hackerActivity`.

For 36 - 40 marks (90% - 100%) All of the above, and:

1. Implementing `HashFunction`, including automatically reading all `keys` from `keys.txt`.
2. Altering all uses of `keys` by the `Client` and the `AccessPoint`, so that they are `hash`ed prior to being added to a `Packet`.
3. Implementing the means for `Hacker` to `crack` a `HandshakePacket` by hashing all possible `keys` from `keys.txt`, and comparing them to the hashed `key` in the `HandshakePacket`.