* Single or double linked structure? Why?

The single linked structure contains two fields, one field of data and one field containing the address of the next node. The single linked structure might be useful for smaller lists, but the design requires a search to go through the list from beginning every time the index of a node is smaller than the previous.

The double linked structure contains one more field than the single linked structure. This added field contains the address of the previous node in the list.

This requires more memory than the single linked list. If the resources are large enough, the design of the linked list allows for traversal searching. Traversal searching means it is possible to go to the previous node in the search of a specific node. If the list size, the start node and the end node is known, it allows to choose from where to start a search accordingly.

According to the size of the used List, it has been chosen to use a double linked list.

* Circular or non‐circular structure? Why?

The circular structure does not have a null pointer at the end of the list (both ends if double linked). Allowing it to run through the list continuously. This disables the opportunity to add nodes to the start or end of the list.

For the assignment it is chosen not to use circular linked lists. The reason is that an end node and a start node is needed for the assignment.

* Sentinels or no sentinels? Why?

A sentinel node is used to make traversal searches easier. A sentinel node contains no data and is pointed at by the end node and the start node. This is used as an alternative to a null pointer check, giving the opportunity to continue a search, traversal or normal, eliminating steps and saving time in the search.

A sentinel comes at the cost of extra pointers and resource usage.

Given the cost versus the efficiency of a sentinel it is decided to use sentinels in the assignment.