**Example of Marking Grid**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Criteria – run solution | Dictionary 40-50 | BSTree 50-60 | AVLTree 60+ |
| 1 |  | Load file into Data Structure |  |  |  |
| 2 |  | Edit unique word information |  |  |  |
| 3 |  | Display number of unique words |  |  |  |
| 4 |  | Remove a unique word |  |  |  |
| 5 |  | Display text concordance |  |  |  |
| 6 |  | Search & Display the Info of a unique word (A: using full word, B: using prefix of a word) |  |  |  |
| 7 |  | Search most common unique word and all the unique words that occur more than a specified number of times |  |  |  |
| 8 |  | Display the unique words in decreasing number of occurrences |  |  |  |
| 9 |  | Display Collocation |  |  |  |

Credit for

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classes** | | Complete | | WordTree – all reusable |
| Working | |  | |  |
| **GUI** Usability – intuitive? | | Menu | | Validation |
| Search – Algorithm Efficiency | |  | |  |
| Testing | | BlackBox / Strategy | | NUnit |
| Refactored | |  | |  |
|  | |  | |  |
| **Classes Word / Location** | Attrib, Types (private/public) | | Locations – list / string | |
|  | Properties | | Iterator | |
|  |  | |  | |
| IComparable | Substring matching | |  | |
|  |  | |  | |
| **Tree** – Generic | Public/private | | Remove balance | |
| All Recursive |  | |  | |
| FindWord |  | |  | |
| FindMostCommonWord | Substring matching | |  | |
|  |  | |  | |
| **Maintainable** code | Presentation (Indentation ect) | | Var Names (Meaningful) | |
|  |  | |  | |
| Extras (Beyond spec – in scope) | Addition of an Help Function | |  | |

Strengths of your submission

This is an excellent project which uses most of the key concept developed in the unit. Good use of the AVL data structure (of class Word). Proper definition of the classes Word and Location. Good use of generics and IComparable interface. As extra there is the possibility to browse the text file. GUI is excellent, elegant and very usable. Search of the word information is generally done efficiently (except for occurrences, see below).

Weaknesses of your submission

Search by occurrences search the entire tree which is inefficient when the tree contains many nodes. It would have been better to search the words by occurrences by keeping the words (eg sorted by occurrences) in some other opportune data structure where you have used the efficient binary search (or similar).

No testing of the code is presented

Clicking on Edit without selecting a word produces an exception not managed in the code

Grade = 88