|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Description** | **Implication** | **Type** |
| **1** | User must be able to choose between a binary tree generator and the educational/quiz mode. | Program form split into two, with each half as a clickable section to begin each mode. |  |
| **2** | Generator screen displays input text boxes, drop down/selection menu and instructions. | Graphical libraries such as SDL should be used to display these input types. |  |
| **3** | User can enter text into the input text boxes. | Text boxes should respond to keyboard input. Text entered should be validated upon submission. |  |
| **4** | User must be able to select traversal type for generator. | Form must have radio buttons or drop down menu. |  |
| **5** | Upon submission/validation of data, a traversal tree will appear beneath the form elements. | Program must calculate/sort elements entered by the user into the nodes, also calculating how many nodes/branches will be required and their position on the screen. Must also sort depending upon chosen traversal type.  Submit button will be included. |  |
| **6** | User should be able to exit the generator to return to the selection screen. | Back button that returns the user to the selection screen upon click. |  |
| **7** | The educational/quiz mode generate randomised node values and an empty tree. | Program must randomly generate a value for each node for the user to guess. |  |
| **8** | Display the possible value for the nodes. | There will be a row of possible values for the nodes that the user can drag-and-drop or select the values for a specific node. |  |
| **9** | User makes a guess of the node value by selecting the value and the node on the tree. | The guess will be compared to that of the internal node value. If the value is correct (for example if the node is ‘E’ and the user selects ‘E’) then the value of that node will be filled in (the colour of the node my change to reflect this). This could be done either by dragging and dropping or by entering text into the node itself. |  |
| **10** | The program should have a difficulty option. | A certain amount of nodes will be pre-filled with values depending on the selected difficult, or possible incremental difficulty level. |  |
| **11** | User should be given a score that is saved to a text file. | Correct choices on higher difficulty levels should result in a higher score, with incorrect choices reducing the score. The score should be visible and updated on the form. |  |
| **12** | User should be able to exit the quiz/educational mode to return to the selection screen. | Back button that returns the user to the selection screen upon click. |  |
| **13** | Software must be compatible with University computers. | The software will be developed using University computers and resources for the most part, which will ensure that the software is compatible with the University software and operating systems. All testing will be done on the University campus. |  |
| **14** | The developed software must use at least one data structure. | For this software, the group has decided to use Rooted Binary Trees as a basis for the program. The program will generate a binary tree based on two factors: data entered by the user for the contents of the leaves within the tree, and a traversal type that sets the order in which the data will be generated, such as in-order, pre-order and post-order. The program will display this graphically onto the screen, and so may use SDL libraries or other similar options. |  |
| **15** | The developed software must use at least one sorting/searching algorithm | As the program will be organising user entered data depending on a traversal type, it is natural that one or many of the sorting algorithms will be used within the program. The program will incorporate “search trees” as well, which will allow for the sorting of leaves based on their value (so the lowest value nodes are to the left of the tree, either as a numerical value or a letter of the alphabet). |  |