HW1 Report

Blake Williams

brw020@uark.edu

2/11/2024

010974718

For this project it was required to implement a linked list and a queue to assist in the guiding interface. In the linked list there were multiple functions to implement. The first was the constructor which simply constructs the nodes in a linked list. It has 3 parameters including value: data stored inside the node, next: pointer to the next node, and prev: pointer to the previous node. Prev is not used as a singly linked list was implemented instead of a doubly. The linked list destructor simply sets the next pointer to null before it is deleted. Linked list constructor initializes a pointer called root to null indicating a empty linked list. The LinkedList Destructor iterates through the linked list, updates a pointer called current set to the next node, then deletes it. Finally it sets the root pointer to null. The LinkedList Insert function works by inserting a new node with a provided value, and using root as the next node. Then it updates root to point to a new node and returns the newNode. The Linked List find function takes in a value and iterates through the list comparing the value of each node. If a match is found, it returns a pointer to that node. Otherwise it returns null. Linked List removal function takes in a value and iterates through the list, and keeping tack of the current node ‘current’ and the previous node ‘prev’. If a node containing the value is found, it updates the pointers to bypass the node then deletes it. It returns the node that was deleted if it is found, otherwise null is returned. Finally the Linked List size function iterates through the list and increment a variable called count representing the number of nodes in the list. For the queue the QueueNode Constructor and Destructor operate similar to the Linked List Constructor and Destructor. Same with the Queue Constructor and Destructor. What changes are the Queue Empty, Pop, and Push Functions. First the Queue Empty function simply checks if the queue is empty. It returns true if head pointer is nullptr else it returns false. The pop function checks if the queue is empty and throws and exception if it is. Otherwise it retrieves the value from the front node, updates the head pointer to the next node, and deletes the old front node. If the queue becomes empty after popping, it sets the tail pointer to nullptr. Finally the Queue Push works by creating a new node with the inputted value. If the queue is empty it sets both head and tail pointers to the new node. Otherwise, it updates the next pointer to the current tail pointer then updates the tail to be a new node. A screenshot of a computer

Description automatically generated