Alexis Benter

Project 1B tDeque COP3530

My tDeque is an dynamic double-ended array that carried similar properties to a stack in which the user is allowed to enter characters, strings, floats, and ints to either side as well as pop them back out from either side.

They can choose to enter from the front or the back of the array by pushing front or pushing back or taking inputs out of the front or the back by popping front or popping back.

The code starts with the setting the initial integer variables to no elements in the array, the minimum length of 8, and the front and back are empty. The queue variable is an array which holds the strings the user inputs in a double ended stack structure. There is then a destructor which allows the user to delete the queue. The size determines the capacity of the array and empty is a Boolean determining wether the array is empty or not.

The array doubles in size if it is full using the growArray method. In this method, there is some exception handling. This is structured as a try-catch-throw set up. The try wraps around the entire contents of the method and catches the exception &e. If there is a bad allocation caught, the program will send the message bad_alloc caught: and then throw e. The array shrinks if it is not already at the minimum size of 8 or less than 1/4 full using the shrinkArray method. In this method, there is some exception handling. This is structured as a try-catch-throw set up. The try wraps around the entire contents of the method and catches the exception &e. If there is a bad allocation caught, the program will send the message bad_alloc caught: and then throw e.

The methods that implement the double ended queue allow the user to push and pop strings from the back or front to modify the array. In pushFront the number of elements is always going to grow by 1 and the new string will be added to the front of the array. The pushBack method also is going to have the array grow by 1 element but place the new string in the back of the array. The popFront method deletes the string at the front of the array and decreases the number of elements by 1. The popBack deletes the string at the back of the array and also decreases the number of elements by 1. If the string is empty, there is nothing to pop. The size is the length of the array. The empty boolean checks wether the array is empty or not. The toStr method returns the queue that the user has created, each string being on a separate line and is used in the main function to print the queue.

I used the Deque_main() file that was sent out and tested the test cases making sure the array was pushing and popping properly. I used this main to make sure all my methods in my tDeque.h worked properly. After testing, one

bug I noticed is that when I push from the back a new line is created between the last push and the push back which is seen while popping.