9

Name + SID: Colin Morris-Moncada 006279654

Big-Oh Algorithm Complexities – Vector vs Linked List

To be Completed and Submitted for HW3
(15 pts)

Task 1: (12 pts) Given the implementations of the Vector and Linked List abstract data types we studied in the lectures, indicate the algorithm time-complexity (in terms of numbers of significant operations) for each member function listed. (Functions parameters are omitted; consider them implicit).

Provide your answers by placing 'X' into the applicable cell. Leave blank those cells with labels that do not apply.

If there is a best case and worst case for a function, indicate both. If a function does not exist for one of Vector and List, indicate this with NA in the corresponding cells.

	Vector<T>			List<T>	
	O(1)	O(N)		O(1)	O(N)
::size()	X			X	
::empty()	X			X	
::front()	X			X	
::back()	X			X	
::push_front()		X		X	
::push_back()	X			X	
::pop_front()		X		X	
::pop_back()	X			X	
::operator []	X				X
::begin()	X			X	
::end()	X			X	
::find() *		X			X

*: regardless of whether our data structure has been implemented to include this function answer based of the most natural implementation you can think of for this task.

$\rightarrow$ <T>

Task 2: (3 pts) Explain in your own words the algorithm (= sequence of steps) you would use in order to implement a function Vector<T>::pop_front() and a function List<T>::operator [](int i). Although these functions are not provided in the Standard Template Library (STL) for vector and list, we could certainly create them. In your explanations above, state the likely (and good) reason why the STL does not include these functions.

[continue on next page]