Algorithms

CMPT 435

 to program a few elementary data structures and imple Develop and test a Node class. Use it to develop a stack, 	ement a few sor	ting a	lgori	hme		
• Develop and test a Node class. Use it to develop a stack,						
a queue. You must implement these yourself; you may in features of the language or its libraries for the stack or may use language libraries for other things, like file ope	ck, then to develop 1 y not use any built- ck or queue. (You operations, etc.)			[5 points]		
 Download the the text file magicitems.txt from our v Read it line-by-line into array. 	web site.					
 Check each line of the array to see if it's a palindrome, is and capitalization. Print it out only if it is a palindrome To check whether or not a given string is a palindrom character by character (ignoring spaces and capitalize each character on a stack and enqueue each character When every character is on the stack and in the que and dequeue the queue one character at a time. If the then the string is a palindrome. 	gnoring spaces me, take it ization) and pus cer in a queue. ue, pop the stac ney always matc	sh sk h	[5 p	ooints	5]	
 Develop your own implementation of selection sort, in merge sort, and quick sort. Then sort magicitems.txt printing the number of comparisons each time. Be sure array before each sort. To do that, write your own O(n) based on the Knuth shuffle (also known as the Fisher-Y not known as the Rosanna shuffle, which is different). 	sertion sort, using each sor to shuffle the shuffle routine ates shuffle, bu	t,	[15 p	oints	5]	
• Document all code (with line numbers and explanation results (in a table) in a LaTeX document. Make multiple each sort and average the results. For the code, explain how each works, as if you were teaching it to someone line numbers for pedantic clarity. For the sort results, n asymptotic running time of <i>each</i> sort and explain why it	s) and your test runs for the good parts else, referencing ote the t is that way.	of g	[50 p	oints	5]	
 Your code must not be in a package; that just makes it harder for me to separate structure from presentation, be professionally uniquely yours (show some personality), and demonstration 	compile and ter formatted yet ate best practic	st es	[-∞	if not	:]	
 Linked lists are described in the 3rd edition of our text i Stacks and queues are described in the 3rd edition of our chapter 10, starting on page 1110 1000. Insertion sort, merge sort, and quick sort are described sections 2.1, 2.3, and 7.1 respectively. 	n chapter 10.2, Ir text in the beg I in the 3 rd edition	starti ginnin on of	ing or ng of our te	n EC. ext in		
Make many commits to GitHub. I do not want to see one	R A C	E	С	А	R	
nassive "everything" commit when I review your code. [It's $-\infty$ if you do that.) Commit early and often. And make sure your commit messages are descriptive and amusing.	head			**		
	 may use language libraries for other things, like file operation of the series of the seri	 may use language libraries for other things, like file operations, etc.) Download the the text file magicitems.txt from our web site. Read it line-by-line into array. Check each line of the array to see if it's a palindrome, ignoring spaces and capitalization. Print it out only if it is a palindrome. To check whether or not a given string is a palindrome, take it character by character (ignoring spaces and capitalization) and puse each character on a stack and enqueue each character in a queue. When every character is on the stack and in the queue, pop the stac and dequeue the queue one character at a time. If they always mate then the string is a palindrome. Develop your own implementation of selection sort, insertion sort, merge sort, and quick sort. Then sort magicitems.txt using each sor printing the number of comparisons each time. Be sure to shuffle the array before each sort. To do that, write your own 0(n) shuffle routine based on the Knuth shuffle (also known as the Fisher-Yates shuffle, but not known as the Rosanna shuffle, which is different). Document all code (with line numbers and explanations) and your results (in a table) in a LaTeX document. Make multiple test runs for each sort and average the results. For the code, explain the good parts thow each works, as if you were teaching it to someone else, referencing line numbers for pedantic clarity. For the sort results, note the asymptotic running time of <i>each</i> sort and explain why it is that way. Your code must not be in a package; that just makes it harder for me to compile and test separate structure from presentation, be professionally formatted yet uniquely yours (show some personality), and demonstrate best practice. Linked lists are described in the 3rd edition of our text in the beg chapter 10, starting on page 1110 1000. Insertion sort, merge sort, and quick sort are described in the 3rd editio sections 2.1, 2.3, and 7.1 respectively. <td> may use language libraries for other things, like file operations, etc.) Download the the text file magicitems.txt from our web site. Read it line-by-line into array. Check each line of the array to see if it's a palindrome, ignoring spaces and capitalization. Print it out only if it is a palindrome, take it character by character (ignoring spaces and capitalization) and push each character on a stack and enqueue each character in a queue. When every character is on the stack and in the queue, pop the stack and dequeue the queue one character at a time. If they always match then the string is a palindrome. Develop your own implementation of selection sort, insertion sort, merge sort, and quick sort. Then sort magicitems.txt using each sort, printing the number of comparisons each time. Be sure to shuffle the array before each sort. To do that, write your own 0(n) shuffle routine based on the Knuth shuffle (also known as the Fisher-Yates shuffle, but not known as the Rosanna shuffle, which is different). Document all code (with line numbers and explanations) and your results (in a table) in a LaTeX document. Make multiple test runs for each sort and average the results. For the code, explain the good parts of how each works, as if you were teaching it to someone else, referencing line numbers for pedantic clarity. For the sort results, note the asymptotic running time of <i>each</i> sort and explain why it is that way. Your code must not be in a package; that just makes it harder for me to compile and test separate structure from presentation, be professionally formatted yet uniquely yours (show some personality), and demonstrate best practices Linked lists are described in the 3rd edition of our text in the beginnin chapter 10, starting on page 1110 1000. Insertion sort, merge sort, and quick sort are described in the 3rd edition of sections 2.1, 2.3, and 7.1 respectively. </td><td> may use language libraries for other things, like file operations, etc.) Download the the text file magicitems.txt from our web site. Read it line-by-line into array. Check each line of the array to see if it's a palindrome, ignoring spaces and capitalization. Print it out only if it is a palindrome. To check whether or not a given string is a palindrome, take it character by character (ignoring spaces and capitalization) and push each character on a stack and enqueue each character in a queue. When every character is on the stack and in the queue, pop the stack and dequeue the queue one character at a time. If they always match then the string is a palindrome. Develop your own implementation of selection sort, insertion sort, merge sort, and quick sort. Then sort magicitems.txt using each sort, printing the number of comparisons each time. Be sure to shuffle the array before each sort. To do that, write your own O(<i>n</i>) shuffle routine based on the Knuth shuffle (also known as the Fisher-Yates shuffle, but not known as the Rosanna shuffle, which is different). Document all code (with line numbers and explanations) and your results (in a table) in a LaTeX document. Make multiple test runs for each sort and average the results. For the code, explain the good parts of how each works, as if you were teaching it to someone else, referencing line numbers for pedantic clarity. For the sort results, note the asymptotic running time of <i>each</i> sort and explain why it is that way. Your code must Inke lists are described in the 3rd edition of our text in chapter 10.2, starting or Stacks and queues are described in the 3rd edition of our text in the beginning of chapter 10, starting on page 1110 1000. Insertion sort, merge sort, and quick sort are described in the 3rd edition of our text in the beginning of chapter 10, starting on page 110 000. Insertion sort, merge sort, and quick sort are described in the 3rd ed</td><td> may use language libraries for other things, like file operations, etc.) Download the the text file magicitems.txt from our web site. Read it line-by-line into array. Check each line of the array to see if it's a palindrome, ignoring spaces and capitalization. Print it out only if it is a palindrome. To check whether or not a given string is a palindrome, take it character by character (ignoring spaces and capitalization) and push each character on a stack and enqueue each character in a queue. When every character is on the stack and in the queue, pop the stack and dequeue the queue one character at a time. If they always match then the string is a palindrome. Develop your own implementation of selection sort, insertion sort, printing the number of comparisons each time. Be sure to shuffle the array before each sort. To do that, write your own O(n) shuffle routine based on the Knuth shuffle (also known as the Fisher-Yates shuffle, but not known as the Rosanna shuffle, which is different). Document all code (with line numbers and explanations) and your results (in a table) in a LaTeX document. Make multiple test runs for each sort and average the results. For the code, explain the good parts of how each works, as if you were teaching it to someone else, referencing line numbers for pedantic clarity. For the sort results, note the asymptotic running time of <i>each</i> sort and explain why it is that way. Your code must not be in a package; that just makes it harder for me to compile and test separate structure from presentation, be professionally formatted yet uniquely yours (show some personality), and demonstrate best practices Linked lists are described in the 3rd edition of our text in the beginning of chapter 10, starting on page 1110 1000. Insertion sort, merge sort, and quick sort are described in the 3rd edition of our text in sections 2.1, 2.3, and 7.1 respectively. </td>	 may use language libraries for other things, like file operations, etc.) Download the the text file magicitems.txt from our web site. Read it line-by-line into array. Check each line of the array to see if it's a palindrome, ignoring spaces and capitalization. Print it out only if it is a palindrome, take it character by character (ignoring spaces and capitalization) and push each character on a stack and enqueue each character in a queue. When every character is on the stack and in the queue, pop the stack and dequeue the queue one character at a time. If they always match then the string is a palindrome. Develop your own implementation of selection sort, insertion sort, merge sort, and quick sort. Then sort magicitems.txt using each sort, printing the number of comparisons each time. Be sure to shuffle the array before each sort. To do that, write your own 0(n) shuffle routine based on the Knuth shuffle (also known as the Fisher-Yates shuffle, but not known as the Rosanna shuffle, which is different). Document all code (with line numbers and explanations) and your results (in a table) in a LaTeX document. Make multiple test runs for each sort and average the results. For the code, explain the good parts of how each works, as if you were teaching it to someone else, referencing line numbers for pedantic clarity. For the sort results, note the asymptotic running time of <i>each</i> sort and explain why it is that way. Your code must not be in a package; that just makes it harder for me to compile and test separate structure from presentation, be professionally formatted yet uniquely yours (show some personality), and demonstrate best practices Linked lists are described in the 3rd edition of our text in the beginnin chapter 10, starting on page 1110 1000. Insertion sort, merge sort, and quick sort are described in the 3rd edition of sections 2.1, 2.3, and 7.1 respectively. 	 may use language libraries for other things, like file operations, etc.) Download the the text file magicitems.txt from our web site. Read it line-by-line into array. Check each line of the array to see if it's a palindrome, ignoring spaces and capitalization. Print it out only if it is a palindrome. To check whether or not a given string is a palindrome, take it character by character (ignoring spaces and capitalization) and push each character on a stack and enqueue each character in a queue. When every character is on the stack and in the queue, pop the stack and dequeue the queue one character at a time. If they always match then the string is a palindrome. Develop your own implementation of selection sort, insertion sort, merge sort, and quick sort. Then sort magicitems.txt using each sort, printing the number of comparisons each time. Be sure to shuffle the array before each sort. To do that, write your own O(<i>n</i>) shuffle routine based on the Knuth shuffle (also known as the Fisher-Yates shuffle, but not known as the Rosanna shuffle, which is different). Document all code (with line numbers and explanations) and your results (in a table) in a LaTeX document. Make multiple test runs for each sort and average the results. For the code, explain the good parts of how each works, as if you were teaching it to someone else, referencing line numbers for pedantic clarity. For the sort results, note the asymptotic running time of <i>each</i> sort and explain why it is that way. Your code must Inke lists are described in the 3rd edition of our text in chapter 10.2, starting or Stacks and queues are described in the 3rd edition of our text in the beginning of chapter 10, starting on page 1110 1000. Insertion sort, merge sort, and quick sort are described in the 3rd edition of our text in the beginning of chapter 10, starting on page 110 000. Insertion sort, merge sort, and quick sort are described in the 3rd ed	 may use language libraries for other things, like file operations, etc.) Download the the text file magicitems.txt from our web site. Read it line-by-line into array. Check each line of the array to see if it's a palindrome, ignoring spaces and capitalization. Print it out only if it is a palindrome. To check whether or not a given string is a palindrome, take it character by character (ignoring spaces and capitalization) and push each character on a stack and enqueue each character in a queue. When every character is on the stack and in the queue, pop the stack and dequeue the queue one character at a time. If they always match then the string is a palindrome. Develop your own implementation of selection sort, insertion sort, printing the number of comparisons each time. Be sure to shuffle the array before each sort. To do that, write your own O(n) shuffle routine based on the Knuth shuffle (also known as the Fisher-Yates shuffle, but not known as the Rosanna shuffle, which is different). Document all code (with line numbers and explanations) and your results (in a table) in a LaTeX document. Make multiple test runs for each sort and average the results. For the code, explain the good parts of how each works, as if you were teaching it to someone else, referencing line numbers for pedantic clarity. For the sort results, note the asymptotic running time of <i>each</i> sort and explain why it is that way. Your code must not be in a package; that just makes it harder for me to compile and test separate structure from presentation, be professionally formatted yet uniquely yours (show some personality), and demonstrate best practices Linked lists are described in the 3rd edition of our text in the beginning of chapter 10, starting on page 1110 1000. Insertion sort, merge sort, and quick sort are described in the 3rd edition of our text in sections 2.1, 2.3, and 7.1 respectively. 	