Cameron Pickle

1. Who is your programming partner? Which of you submitted the source code of your program?

Gage Glenn, he is going to submit the code.

2. Have you worked with more than one partner yet? Remember, you are required to switch at least once this semester.

Yes, This is my second partner.

3. In the MyStack class, the stack data structure is implemented using a doubly-linked list. Would it be better to use a singly-linked list instead? Defend your answer.

If your stack is only to support the operations push, pop, and peek then a singly-linked list would be better. It would require less pointers to other nodes. You only need to have the head as the item that is currently on the top of the stack. Every time that you add to the stack push the new item and make it the head with a reference to the item that previously was the head. When popping the stack follow the reference of the head to the next item and set that item to the head

4. Would it be possible to replace the instance of MyLinkedList in the MyStack class with an instance of Java's LinkedList? Why or why not?

Yes, the Java'sLinkedList supports all of the methods that MyLinkedList does and would be able to seamlessly replace MyLinkedList. Java'sLinkedList supports all of the features and many more than MyLinkedList.

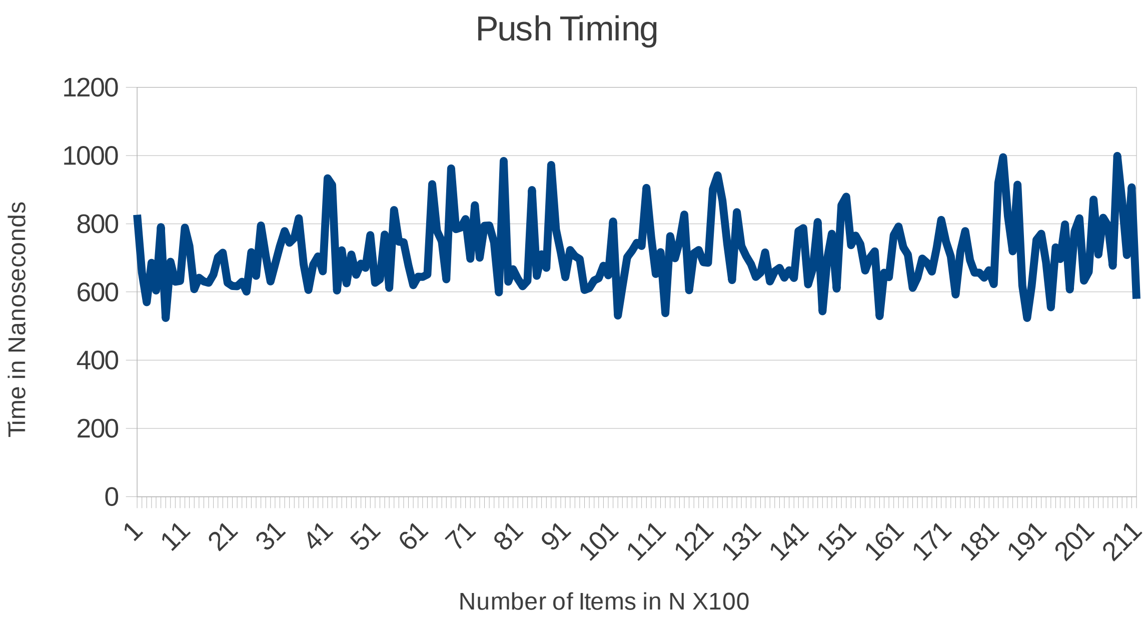
5. Comment on the efficiency of your time spent developing the MyStack class.

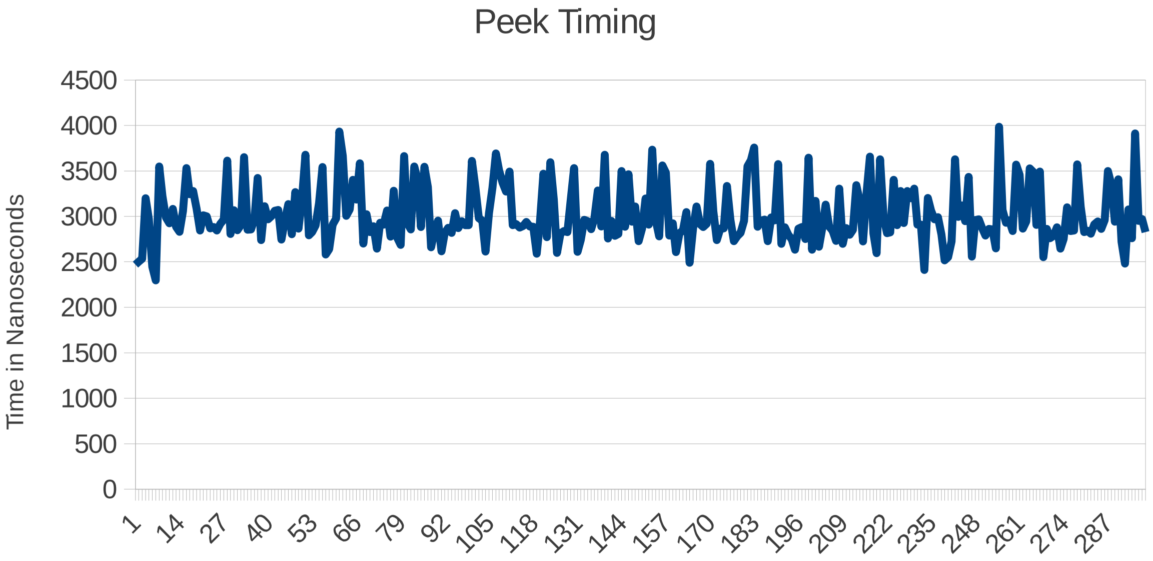
We were very efficient at developing the MyStack.class. It was a simple class that pulls a few of the features already implemented in the MyLinkedList class. It only required about fifteen minutes to complete.

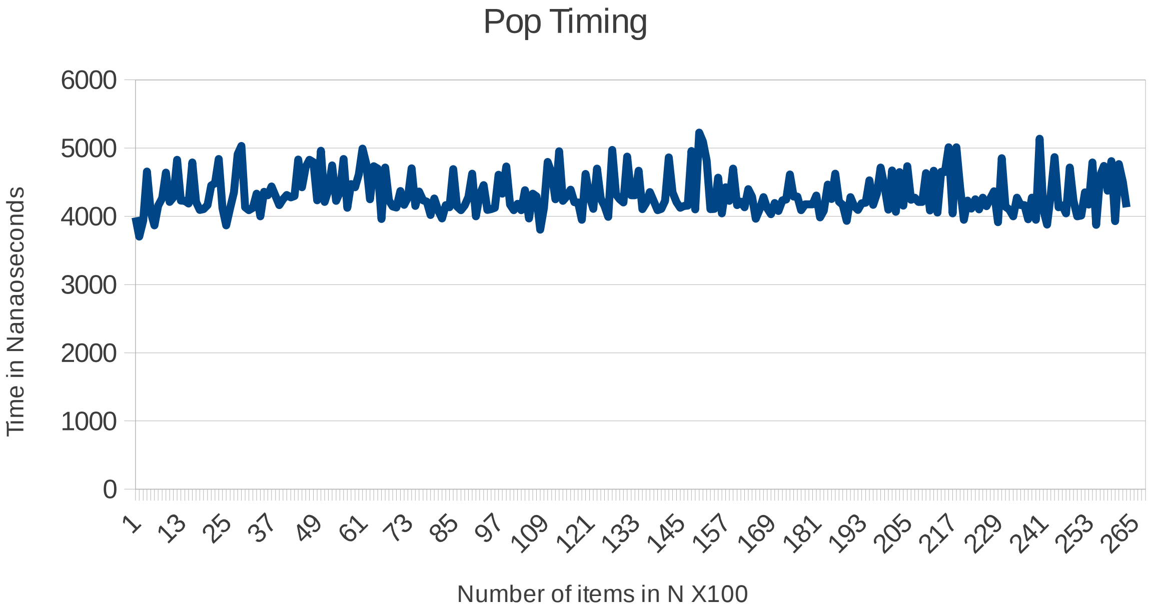
6. Note that the line and column number given by BalancedSymbolChecker indicate the location in a file where an unmatched symbol is detected (i.e., where the closing symbol is expected). Explain how you would also keep track of the line and column number of the unmatched opening symbol. For example, in Class1.java, the unmatched symbol is detected at line 6 and column 1, but the original '(' is located at line 2 and column 24.

You could achieve this by creating a class to store the symbol that you push onto the stack that could also contain stored in the object the reference of where the symbol was located. That way when you reach the location where the closing symbol was expected you may print that out and the reference that is stored in the object of the created class.

7. Collect and plot running times in order to determine if the running times of the MyStack methods push, pop, and peek are O(1) as expected.


Push Timing
1200
1000
800
600
400
200
0
Time in Nanoseconds
1
11
21
31
41
51
61
71
81
91
101
111
121
131
141
151
161
171
181
191
201
211
Number of Items in N X100


Peek Timing
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Time in Nanoseconds
1
14
27
40
53
66
79
92
105
118
131
144
157
170
183
196
209
222
235
248
261
274
287


Pop Timing
6000
5000
4000
3000
2000
1000
0
Time in Nanaoseconds
1
13
25
37
49
61
73
85
97
109
121
133
145
157
169
181
193
205
217
229
241
253
265
Number of items in N X100

Yes, they all have that constant runtime O(1) as expected. There is some noise in the graphs from background tasks but it show that it is a constant rate.

8. How many hours did you spend on this assignment?

We spent 7 hours on this assignment.