A doubly linked list – Testing

On my honour, I pledge that I have neither received nor provided improper

assistance in the completion of this assignment. Signed: Keonho Lim

Test your functionalities properly and seriously. If your implementation and timing should work properly and correctly. If your timing does not match with your code or does not work, you will not get a full credit or even get a penalty for your implementation.

* Use the proper time unit such that we can immediately understand the amount of time taken. If it takes less than a second, you may just write **"instant**". Therefore, use instant, year, day, hours and sec appropriately.Follow the instructions if necessary.
* Fill the timing **ONLY if your code works** for that functionality. Leave it as blank including if it does not work**.**
* Fill the dark grey blanks with an estimated time, based on your actual timing and **ONLY if your code works**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | | 10,000 | 100,000 | 500,000 | Instructions |
| Pop\_all  O(n) | my code | **0.000533 sec** | **0.005613 sec** | **0.025571 sec** | Using push\_backN, insert N/2 random numbers first, then N/2 nodes with **one** fixed number. |
| listdblx | **0.000526 sec** | **0.005563 sec** | **0.025107 sec** |
| unique  O(n) | my code | **0.000377 sec** | **0.002611 sec** | **0.010615 sec** | Insert N random numbers. Use quicksort() to save your time. |
| listdblx | **0.000438 sec** | **0.00319 sec** | **0.012137 sec** |
| selection sort  O(n^2) | my code | **0.362334 sec** | **36.0251 sec** |  |  |
| listdblx | **0.209791 sec** | **21.0716 sec** |  |
| reverse  O(n) | my code | **0.000264 sec** | **0.001973 sec** | **0.008579 sec** | Insert N random numbers. |
| listdblx | **0.000171 sec** | **0.00178 sec** | **0.043156 sec** |
| Shuffle/half  O(n) | my code | **0.147721 sec** | **14.9664 sec** | **3.82105 sec** |
| listdblx | **0.000112 sec** | **0.001724 sec** | **0.006751 sec** |
| push sorted  O(n) | my code | **0.000134 sec** | **0.001451 sec** | **0.004842 sec** |
| listdblx | **0.000108 sec** | **0.001232 sec** | **0.005501 sec** |
| push sortedN  O(n^2) | my code | **0.160431 sec** | **35.1256 sec** |  | Using push\_backN, insert N/2 random numbers twice to have N numbers sorted in the list. . |
| listdblx | **0.113868 sec** | **25.3537 sec** |  |
| push sortedN  O(n log n) | my code |  |  |  |
| listdblx | **0.001502 sec** | **0.018022 sec** | **0.080801 sec** |
| binary search | my code | **0.00015 sec** | **0.000969 sec** | **0.004635 sec** | Insert N random numbers, sort and search for the last one. |
| listdblx | **0.000126 sec** | **0.00114 sec** | **0.005462 sec** |