A doubly linked list – Testing

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 |  |  |  | Using push\_backN, insert N/2 random numbers first, then N/2 nodes with **one** fixed number. |
| listdblx |  |  |  |
| unique  O(n) | my code |  |  |  | Insert N random numbers. Use quicksort() to save your time. |
| listdblx |  |  |  |
| selection sort  O(n^2) | my code |  |  |  |  |
| listdblx |  |  |  |
| reverse  O(n) | my code |  |  |  | Insert N random numbers. |
| listdblx |  |  |  |
| Shuffle/half  O(n) | my code |  |  |  |
| listdblx |  |  |  |
| push sorted  O(n) | my code |  |  |  |
| listdblx |  |  |  |
| push sortedN  O(n^2) | my code |  |  |  | Using push\_backN, insert N/2 random numbers twice to have N numbers sorted in the list. . |
| listdblx |  |  |  |
| push sortedN  O(n log n) | my code |  |  |  |
| listdblx |  |  |  |
| binary search | my code |  |  |  | Insert N random numbers, sort and search for the last one. |
| listdblx |  |  |  |