On my honor, I pledge that I have neither recieved nor provided improper assistance in the completion of this assignment. -21900112 김태민



# A doubly linked list – Self Testing

Test your functionalities properly and seriously.
Mark Pass or Fail for each operation. 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.

| Step | Operations | Point | Testing | comments |
|---|---|---|---|---|
| 1 | find, more, less | 0.5 | P/F | Check the code.<br>find() use one while loop, but not if<br>Use push command to test find(). |
| 2 | push commands<br>push() | 0.5 | P/F | Check the code.<br>push() must use find() and insert(), not more than 3~4 lines of code<br>Don't add a new node if the position x is not found |
| 3 | pop commands<br>pop_all* | 0.5 | P/F | Test it with over 100,000 samples.<br>Make sure O(n), not O(n^2)<br>Use testing method described below |
| 4 | half() and show()* | 0.5 | P/F | Check the correctness the middle node.<br>half() is used in shuffle() and show()<br>Test it with both odd & even number sequences.<br>Record the timing of half() for 20 million samples displayed using show HEAD/TAIL on exiting.<br>Method 1: 0.096726 Method 2: 0.059277 .<br>Method 2 is faster than Method 1 by about 63.176 % |
| 5 | **swap_pairs** | **0.0** | **PASS** | Check the code. It must go through the list once, not twice nor more.<br>Test it with both odd & even number sequences. |
| 6 | sorted() | 0.5 | P/F | It is checked by other operations. |
| 7 | push_sorted() | 0.5 | P/F | Check it with unsorted, ascending and descending ordered lists. Make sure duplicated ones included such as 3 5 5 7 9 9 9. Use "reverse" menu option.<br>Test it with over 100,000 samples. pushN to generate samples and push_sorted 100001.<br>Make sure O(n), not O(n^2)<br>Additionally, use testing method described below |
| 8 | unique()* | 0.5 | P/F | Test it with over 100,000 samples.<br>Make sure O(n), not O(n^2)<br>Use testing method described below |
| 9 | **reverse()** | **0.0** | **PASS** | Test it with over 100,000 samples.<br>Make sure O(n), not O(n^2) |
| 10 | **randomize()** | **0.0** | **PASS** | Test it with over 100,000 samples.<br>Make sure O(n), not O(n^2)<br>The commands sort & quicksort uses randomize(). |
| 11 | shuffle()* | 0.5 | P/F | Check the exactness.<br>Test it with both odd & even number sequences. |

| | | | | |
|---|---|---|---|---|
| | Total | 4.0 | 0.8 + 0.2 | Extra 0.1 p per step for a proper testing<br>Extra 0.2 if you get them all right |

**Test Hint 1: pop_all()**
To test pop_all(), you may need to generate a sequence that has a consecutive numbers of a certain value. You may use "push back N" command option with a negative N provided.

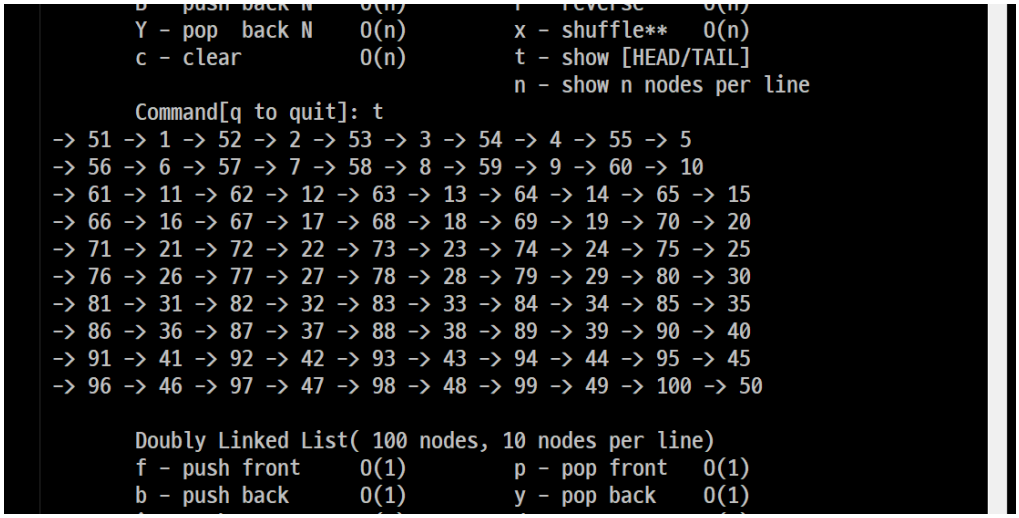
For example, make a sequence with ten thousands and another ten thousands of 7 samples:

- select "push_back_N" and enter 10,000 for random samples
- select "push_back_N" and enter -10000, then enter "7" for a value.
- run "pop-all" 7.

**Test Hint 2: unique()**
To test unique(), you may also need to generate a sorted sequence with consecutive numbers of a certain value. You may use "push back N" command option with a negative N provided.

For example, make a sequence with thirty thousands for each 1, 5, and 7, and ten thousands for 9, respectively and run unique().

**Test Hint 3: show(), pop(), push_sorted(), push_backN(), ...**
Make a sequence of numbers from 1 to 100 as shown below in a fewer steps possible. Then you may need to use all kinds of commands you implemented so far.



# Find, more, less

```
pNode find(pList p, int val) {
    DPRINT(cout << ">find val=" << val << endl;);

    pNode curr = begin(p);
    while(curr!=end(p)&&curr->data!=val) curr=curr->next;

    return curr;
}
#endif
```

While without if statement

```
pset9 — ./driver — ./driver — driver — 80×24
        u - unique*           O(n)        w - swap pairs   O(n)
        t - show [HEAD/TAIL]              n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: n
        Enter number of nodes to show per line: 10
FRONT   1       2       3       4       5       6

        Doubly Linked List(nodes:6, show:ALL,10)
        f - push front        O(1)        p - pop front    O(1)
        b - push back         O(1)        y - pop back     O(1)
        B - push back N       O(n)        Y - pop back N   O(n)
        i - push              O(n)        d - pop          O(n)
        z - push sorted*      O(n)        e - pop all*     O(n)

        s - sorted?           O(n)        r - reverse      O(n)
        x - perfect shuffle*  O(n)        a - randomize    O(n)
        u - unique*           O(n)        w - swap pairs   O(n)
        t - show [HEAD/TAIL]              n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: i
        Enter a number to push: 99
        Choose a position node: 4
FRONT   1       2       3       99      4       5       6
```

Testing find() with push command(i)

## Push commands, push()

```
void push(pList p, int val, int x) {
    DPRINT(cout << ">push val=" << val << endl;);
    pNode y=find(p,x);
    if(y->next==nullptr) return;
    insert(y,val);
    DPRINT(cout << "<push\n";);
}
```

Push() using both find()&insert(), implemented within 3 lines.
By using find(), we can get the target node that has same parameter val.
According to find(), if position is not found, it will return last node. And since last node's next is nullptr, push() doesn't add a new node and just returns without any actions.

## Pop commands, pop_all()

```
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: B
        Enter N nodes to push back(-N for a value)?: -100000
        Enter a value to push back?: 10
        pushing [90000]=10
        cpu: 0.006619 sec
FRONT   10      10      10      10      10
        ...  10  ...
        10      10      10      10      10

        Doubly Linked List(nodes:100000, show:HEAD/TAIL,5)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
        z - push sorted*     O(n)       e - pop all*    O(n)

        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: e
        Enter a number to pop all: 10
        cpu: 0.006611 sec

        The list is empty.

        Doubly Linked List(nodes:0, show:HEAD/TAIL,5)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
```

Testing with 100,000 samples

By using pop_all, we can get rid of every number that is same as input number. To fix the bug, I temporarily saved the node's next address before the erase().

```
temp=c->next;
erase(p,c);
c=temp;
```

Since erase() deletes c node, I set the c as saved node address. By this way, the bug doesn't occur anymore.

# Half() and show()

```
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]:
Retry -         Command[q to quit]: t
FRONT   1       2       3       4
       [5]      6       7       8
        9

        Doubly Linked List(nodes:9, show:ALL,4)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
        z - push sorted*     O(n)       e - pop all*    O(n)

        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [HEAD/TAIL]            n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: t
FRONT   1       2       3       4
        ... [5] ...
        6       7       8       9

        Doubly Linked List(nodes:9, show:HEAD/TAIL,4)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
        z - push sorted*     O(n)       e - pop all*    O(n)
```

The length of list is 9, and 4 nodes are shown per list in HEAD/TAIL menu.
We can see that it works properly.
What about the case when length of list is even number?

```
        z - push sorted*     O(n)       e - pop all*    O(n)

        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: t
FRONT   1       2       3       4
        5      [6]      7       8
        9       10

        Doubly Linked List(nodes:10, show:ALL,4)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
        z - push sorted*     O(n)       e - pop all*    O(n)

        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [HEAD/TAIL]            n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: t
FRONT   1       2       3       4
        ... [6] ...
        7       8       9       10

        Doubly Linked List(nodes:10, show:HEAD/TAIL,4)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
```

When list length is even number, the half node should be the first node of second half node.
We can see that it also work as intended.

There are two ways to implement half(). Both methods' time complexity is O(n), but the second method is faster.

First method uses list's size and goes through halfway using for loop.
Let me test it for 20,000,000 samples.

```
        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: B
        Enter N nodes to push back(-N for a value)?: -20000000
        Enter a value to push back?: 1
        pushing [19990000]=1
        cpu: 0.355146 sec
FRONT   1       1       1       1       1
        ...  1  ...
        1       1       1       1       1

        Doubly Linked List(nodes:20000000, show:HEAD/TAIL,5)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
        z - push sorted*     O(n)       e - pop all*    O(n)

        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: q
        Congratulatios! It's half-time: cpu: 0.096726 sec

        Cleared...

        Joyful Coding~~
→  pset9
```

First method(for loop): 0.096726 sec

```
        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: B
        Enter N nodes to push back(-N for a value)?: -20000000
        Enter a value to push back?: 1
        pushing [19990000]=1
        cpu: 0.351403 sec
FRONT   1       1       1       1       1
        ...  1  ...
        1       1       1       1       1

        Doubly Linked List(nodes:20000000, show:HEAD/TAIL,5)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
        z - push sorted*     O(n)       e - pop all*    O(n)

        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: q
        Congratulatios! It's half-time: cpu: 0.059277 sec

        Cleared...

        Joyful Coding~~
→  pset9
```

Second method(rabbit and turtle): 0.059277 sec

So the second method is about 1.63 times faster than the first method.

# sorted()

checked by other operations ☺

# push_sorted()

```
        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [HEAD/TAIL]            n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: t
FRONT   17      40      13      20      24
        ...  31  ...
        27      44      19      24      9

        Doubly Linked List(nodes:50, show:HEAD/TAIL,5)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
        z - push sorted*     O(n)       e - pop all*    O(n)

        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: z
        The operation works in sorted list only.
FRONT   17      40      13      20      24
        ...  31  ...
        27      44      19      24      9

        Doubly Linked List(nodes:50, show:HEAD/TAIL,5)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
```

If we try push_sorted() with usorted list, it is not possible.

```
        z - push sorted*     O(n)       e - pop all*    O(n)

        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: s
        Sorted in ascending order
FRONT   3       5       5       7       9
        9       9       9

        Doubly Linked List(nodes:8, show:HEAD/TAIL,5)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
        i - push             O(n)       d - pop         O(n)
        z - push sorted*     O(n)       e - pop all*    O(n)

        s - sorted?          O(n)       r - reverse     O(n)
        x - perfect shuffle* O(n)       a - randomize   O(n)
        u - unique*          O(n)       w - swap pairs  O(n)
        t - show [ALL]                  n - n nodes per line
        c - clear       O(n)
        Command[q to quit]: z
        Enter a number to push: 5
        cpu: 2.1e-05 sec
FRONT   3       5       5       5       7
        9       9       9       9

        Doubly Linked List(nodes:9, show:HEAD/TAIL,5)
        f - push front       O(1)       p - pop front   O(1)
        b - push back        O(1)       y - pop back    O(1)
        B - push back N      O(n)       Y - pop back N  O(n)
```

Since the list is sorted in ascending order, we can try push_sorted(). We can see that 5 is properly inserted.

```
        s – sorted?          O(n)       r – reverse     O(n)
        x – perfect shuffle* O(n)       a – randomize   O(n)
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: s
        Sorted in descending order
FRONT   9       9       9       9       7
        5       5       3

        Doubly Linked List(nodes:8, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
        B – push back N      O(n)       Y – pop back N  O(n)
        i – push             O(n)       d – pop         O(n)
        z – push sorted*     O(n)       e – pop all*    O(n)

        s – sorted?          O(n)       r – reverse     O(n)
        x – perfect shuffle* O(n)       a – randomize   O(n)
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: z
        Enter a number to push: 8
        cpu: 1.6e-05 sec
FRONT   9       9       9       9       8
        7       5       5       3

        Doubly Linked List(nodes:9, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
        B – push back N      O(n)       Y – pop back N  O(n)
        i – push             O(n)       d – pop         O(n)
```

Since the list is sorted in descending order, we can try push_sorted(). We can see that 8 is inserted properly.

```
        s – sorted?          O(n)       r – reverse     O(n)
        x – perfect shuffle* O(n)       a – randomize   O(n)
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: s
        Sorted in ascending order
FRONT   1       2       2       2       2
        ...  2  ...
        2       2       2       2       2

        Doubly Linked List(nodes:100000, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
        B – push back N      O(n)       Y – pop back N  O(n)
        i – push             O(n)       d – pop         O(n)
        z – push sorted*     O(n)       e – pop all*    O(n)

        s – sorted?          O(n)       r – reverse     O(n)
        x – perfect shuffle* O(n)       a – randomize   O(n)
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: z
        Enter a number to push: 3
        cpu: 0.001376 sec
FRONT   1       2       2       2       2
        ...  2  ...
        2       2       2       2       3

        Doubly Linked List(nodes:100001, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
```

```
        s – sorted?          O(n)       r – reverse     O(n)
        x – perfect shuffle* O(n)       a – randomize   O(n)
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: s
        Sorted in descending order
FRONT   3       2       2       2       2
        ...  2  ...
        2       2       2       2       2

        Doubly Linked List(nodes:100000, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
        B – push back N      O(n)       Y – pop back N  O(n)
        i – push             O(n)       d – pop         O(n)
        z – push sorted*     O(n)       e – pop all*    O(n)

        s – sorted?          O(n)       r – reverse     O(n)
        x – perfect shuffle* O(n)       a – randomize   O(n)
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: z
        Enter a number to push: 1
        cpu: 0.003615 sec
FRONT   3       2       2       2       2
        ...  2  ...
        2       2       2       2       1

        Doubly Linked List(nodes:100001, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
```

Result of testing with 100,000 samples.

# unique()

```
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: z
        Enter a number to push: 1
        cpu: 0.003615 sec
FRONT   3       2       2       2       2
        ...  2  ...
        2       2       2       2       1

        Doubly Linked List(nodes:100001, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
        B – push back N      O(n)       Y – pop back N  O(n)
        i – push             O(n)       d – pop         O(n)
        z – push sorted*     O(n)       e – pop all*    O(n)

        s – sorted?          O(n)       r – reverse     O(n)
        x – perfect shuffle* O(n)       a – randomize   O(n)
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: u
        cpu: 0.003851 sec
FRONT   3       2       1

        Doubly Linked List(nodes:3, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
        B – push back N      O(n)       Y – pop back N  O(n)
        i – push             O(n)       d – pop         O(n)
        z – push sorted*     O(n)       e – pop all*    O(n)

        s – sorted?          O(n)       r – reverse     O(n)
```

Testing with one ‘3’, 99,999 ‘2’s, one ‘1’.

We can see that all the ‘2’s are wiped out except one remainder. The existing bug is fixed by the same method of pop_all().

```
        z – push sorted*     O(n)       e – pop all*    O(n)

        s – sorted?          O(n)       r – reverse     O(n)
        x – perfect shuffle* O(n)       a – randomize   O(n)
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: b
        Enter a number to push: 9
FRONT   3       5       5       7       9
        9       9       9

        Doubly Linked List(nodes:8, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
        B – push back N      O(n)       Y – pop back N  O(n)
        i – push             O(n)       d – pop         O(n)
        z – push sorted*     O(n)       e – pop all*    O(n)

        s – sorted?          O(n)       r – reverse     O(n)
        x – perfect shuffle* O(n)       a – randomize   O(n)
        u – unique*          O(n)       w – swap pairs  O(n)
        t – show [ALL]                  n – n nodes per line
        c – clear       O(n)
        Command[q to quit]: u
        cpu: 1.8e-05 sec
FRONT   3       5       7       9

        Doubly Linked List(nodes:4, show:HEAD/TAIL,5)
        f – push front       O(1)       p – pop front   O(1)
        b – push back        O(1)       y – pop back    O(1)
        B – push back N      O(n)       Y – pop back N  O(n)
        i – push             O(n)       d – pop         O(n)
        z – push sorted*     O(n)       e – pop all*    O(n)
```

We can see that all duplicate number is wiped out.

# 11. shuffle()

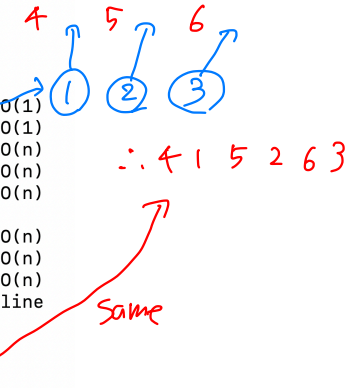
```
b – push back        O(1)       y – pop back    O(1)
B – push back N      O(n)       Y – pop back N  O(n)
i – push             O(n)       d – pop         O(n)
z – push sorted*     O(n)       e – pop all*    O(n)

s – sorted?          O(n)       r – reverse     O(n)
x – perfect shuffle* O(n)       a – randomize   O(n)
u – unique*          O(n)       w – swap pairs  O(n)
t – show [ALL]                  n – n nodes per line
c – clear       O(n)
Command[q to quit]: n
Enter number of nodes to show per line: 8
FRONT  1       2       3       4       5       6

Doubly Linked List(nodes:6, show:HEAD/TAIL,8)
f – push front       O(1)       p – pop front   O(1)
b – push back        O(1)       y – pop back    O(1)
B – push back N      O(n)       Y – pop back N  O(n)
i – push             O(n)       d – pop         O(n)
z – push sorted*     O(n)       e – pop all*    O(n)

s – sorted?          O(n)       r – reverse     O(n)
x – perfect shuffle* O(n)       a – randomize   O(n)
u – unique*          O(n)       w – swap pairs  O(n)
t – show [ALL]                  n – n nodes per line
c – clear       O(n)
Command[q to quit]: x
cpu: 1.8e-05 sec
FRONT  4       1       5       2       6       3

Doubly Linked List(nodes:6, show:HEAD/TAIL,8)
f – push front       O(1)       p – pop front   O(1)
b – push back        O(1)       y – pop back    O(1)
B – push back N      O(n)       Y – pop back N  O(n)
```




When list length is even number

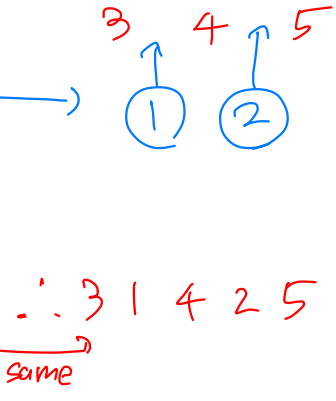
```
B – push back N      O(n)       Y – pop back N  O(n)
i – push             O(n)       d – pop         O(n)
z – push sorted*     O(n)       e – pop all*    O(n)

s – sorted?          O(n)       r – reverse     O(n)
x – perfect shuffle* O(n)       a – randomize   O(n)
u – unique*          O(n)       w – swap pairs  O(n)
t – show [ALL]                  n – n nodes per line
c – clear       O(n)
Command[q to quit]: y
FRONT  1       2       3       4       5

Doubly Linked List(nodes:5, show:HEAD/TAIL,8)
f – push front       O(1)       p – pop front   O(1)
b – push back        O(1)       y – pop back    O(1)
B – push back N      O(n)       Y – pop back N  O(n)
i – push             O(n)       d – pop         O(n)
z – push sorted*     O(n)       e – pop all*    O(n)

s – sorted?          O(n)       r – reverse     O(n)
x – perfect shuffle* O(n)       a – randomize   O(n)
u – unique*          O(n)       w – swap pairs  O(n)
t – show [ALL]                  n – n nodes per line
c – clear       O(n)
Command[q to quit]: x
cpu: 8e-06 sec
FRONT  3       1       4       2       5

Doubly Linked List(nodes:5, show:HEAD/TAIL,8)
f – push front       O(1)       p – pop front   O(1)
b – push back        O(1)       y – pop back    O(1)
B – push back N      O(n)       Y – pop back N  O(n)
i – push             O(n)       d – pop         O(n)
z – push sorted*     O(n)       e – pop all*    O(n)
```




When list length is even number