

## Testing Strategy

1. When the linked list is empty, test `is_empty()`, `size()`, `print()` functions. The `print()` function will be used for other tests, so I have to make sure it is correct.
2. Test `push_front()` function. I push\_front 10000 items with two purposes: (1) check if the function will work for both empty and not empty lists; (2) check if there is a memory leak which may lead the program to crash.
3. Now the list is not empty, test the `is_empty()`, `size()`, and `print()` functions again.
4. Test the `clear()` function, and reset current list.
5. Test `push_back()` function. I push\_back 10000 items with two purposes: (1) check if the function will work for both empty and not empty lists; (2) check if there is a memory leak which may lead the program to crash.
6. Test `item_at()`, `T& operator[](int i)`, and `T const& operator[](int i) const`. To test this, I first added some characteristics other than 'A' to both the beginning and end of the list. Then, I tested items at beginning (0), items in the middle (1000) and items exceed the index (20000, which throws an exception and terminates the program, so I commented it out.). Similar idea to test the `operator[]` overload. However, I was unable to test the `const& operator[]` because I don't know how to call it.
7. Test `contains()`. I used the functions provided by the professor.
8. Test `insert()`. I inserted three items at the beginning, middle, and exceed the end.
9. Test `replace()`. Replace the item at position 4 and use `item_at()` to check if it's correct or not.
10. Test `remove()`. I remove three items at the beginning, middle, and exceed the list as well. And use the `print()` function to check the correctness.
11. Test `pop_front()` and `pop_back()`. I popped 4000 items for each function, and use `size()` to check if it is correct or have a crash.
12. Test `iterator` and `const_iterator`. To test these, I first push some characters other than 'A' to the beginning and end of the list. The `const_iterator` is a copy of the regular `iterator`. Other than that, the testing strategy for both `iterator` are the same as following:

(1) I test the `*`, `begin()` and `end()` function. I assign the begin of list to a iterator, and use `*` check if it's correct or not. Same strategy for `end()`. However, because the `end()` is a pointer passed the last item in the list which would be `NULL`, the program crashed, so I commented this test out.

(2) Test `->()` which is same as `*`.

(3) Test immutable: assign a character to the iterator. The `const_iterator` will crash the program, so I commented it out.

(4) Test `=()` function. I defined a new iterator `iter1`, and copy the previous iterator to the `iter1`, and use `*iter1` to see if it is correct.

(5) Test increment (pre and post) function. `operator++(int)` should return the value of the original one, and `++operator()` should return the value of next item.

(6) Test `==()` and `!=()` function. I defined two new lists, one regular and the other one constant. I pushed the same items to the new lists. Then I defined to iterators to the beginning of the new iterators. Similarly, I check if the beginning of the new iterator is the same or different than the previous one.