CSCI 3333 Practice Quiz STL

Problem 1. Select the worst-case running time for each of the following operations in a dynamic-array-based list (i.e. a vector) containing \( n \) elements.

- Accessing an element in the middle of the list (via \( \text{operator=} \)). \( \Theta(1) \) or \( \Theta(n) \)
- Adding an element to the front of the list (via \( \text{push\_front} \)). \( \Theta(1) \) or \( \Theta(n) \)
- Adding an element to the back of the list (via \( \text{push\_back} \)). \( \Theta(1) \) or \( \Theta(n) \)
- Removing an element in the middle of the list (via \( \text{erase} \)). \( \Theta(1) \) or \( \Theta(n) \)

Problem 2. Complete the following template function that removes all elements of a vector \( V \) except the first element:

```cpp
template <typename T>
void nearly_clear(vector<T> &V)
{
    while (V.size() > 1)
    {
        V.pop_back();
    }
}
```

Let \( n \) be the length of \( L \).
The running time of \( \text{nearly\_clear} \) is: \( \Theta(1) \) or \( \Theta(n) \) or \( \Theta(n^2) \)

Problem 3. Select the worst-case running time for each of the following operations in a doubly-linked-list-based list (i.e. a list) containing \( n \) elements.

- Accessing an element in the middle of the list (via \( \text{operator=} \)). \( \Theta(1) \) or \( \Theta(n) \)
- Adding an element to the front of the list (via \( \text{push\_front} \)). \( \Theta(1) \) or \( \Theta(n) \)
- Removing an element from the back of the list (via \( \text{pop\_back} \)). \( \Theta(1) \) or \( \Theta(n) \)
- Adding an element into the middle of the list (via \( \text{insert} \)). \( \Theta(1) \) or \( \Theta(n) \)
Problem 4. Complete the following template function that replaces each element of a list \( L \) with two copies of the element:

```cpp
template <typename T>
void duplicate(list<T> &L)
{
    list<T> C = L;
    L.clear();
    while (C.size() > 0)
    {
        L.push_back(C._____);
        L.push_back(C._____);
        C._____;  // Hint: C.pop_front()
    }
}
```

For instance, the following tests should pass:

```cpp
list<int> L{1, 2, 3};
duplicate(L);
test(L.size() == 6);
list<int>::iterator it = L.begin();
test(*it == 1);
++it;
test(*it == 1);
++it;
test(*it == 2);
++it;
test(*it == 2);
++it;
test(*it == 3);
++it;
test(*it == 3);
++it;
```

Let \( n \) be the length of \( L \).
The running time of `duplicate` is:  \( \square \Theta(1) \quad \square \Theta(n) \quad \square \Theta(n^2) \)
Problem 5. Complete the following function that replaces every element of a list \( L \) with the sum of the elements of \( L \):

```cpp
void replace_with_sum(list<int> _____L)
{
    if (L.size() == 0)
        return;
    int sum = 0;
    list<int>::iterator it = L.begin();
    while (it != L.end())
    {
        sum += _____;
        ++it;
    }
    do
    {
        _____;
        _____ = sum;
    }
    while (it != L.begin());
}
```

For instance, the following tests should pass:

```cpp
list<int> L{1, 2, 3};
replace_with_sum(L);
test(L.size() == 3);
list<int>::iterator it = L.begin();
test(*it == 6);
++it;
test(*it == 6);
++it;
test(*it == 6);
```

Let \( n \) be the length of \( L \).
The running time of `replace_with_sum` is:  □ \( \Theta(1) \)  □ \( \Theta(n) \)  □ \( \Theta(n^2) \)