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 `operator[]`).
  - \(\Theta(1)\)
  - \(\Theta(n)\)

- Adding an element to the back of the list (via `push_back`).
  - \(\Theta(1)\)
  - \(\Theta(n)\)

- Removing an element from the back of the list (via `pop_back`).
  - \(\Theta(1)\)
  - \(\Theta(n)\)

- Removing an element in the middle of the list (via `erase`).
  - \(\Theta(1)\)
  - \(\Theta(n)\)

**Problem 2.** 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 `operator[]`).
  - \(\Theta(1)\)
  - \(\Theta(n)\)

- Adding an element to the front of the list (via `push_front`).
  - \(\Theta(1)\)
  - \(\Theta(n)\)

- Removing an element from the back of the list (via `pop_back`).
  - \(\Theta(1)\)
  - \(\Theta(n)\)

- Adding an element into the middle of the list (via `insert`).
  - \(\Theta(1)\)
  - \(\Theta(n)\)
Problem 3. 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 \texttt{duplicate} is: \( \square \Theta(1) \quad \square \Theta(n) \quad \square \Theta(n^2) \).
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._____;  // Remove one element from C
    }
}
```

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: \( \Theta(1) \) \( \Theta(n) \) \( \Theta(n^2) \)