Name: _____

1. (25 points) In the space below, draw the linked list described by the following operations:

```
SinglyLinkedList<char> lst;
lst.push_front('a');
lst.push_front('b');
lst.push_front('c');
lst.push_back('d');
lst.push_back('e');
lst.push_back('f');
lst.remove(2);
lst.insert(2, 'g');
```

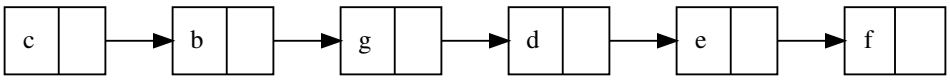


Figure 1: Digraph.

2. (25 points) What is the time complexity of the following operations for singly linked lists with no `head` and `tail` pointers?
   (a) `push_back` an element onto a list of size $n$? $O(n)$
   (b) `push_front` an element onto a list of size $n$? $O(1)$
   (c) `insert` an element into a list of size $n$? $O(n)$
   (d) `remove` an element from a list of size $n$? $O(n)$
   (e) `append` a list of size $n$ onto a list of size $m$? $O(m)$

3. (50 points) Write a function which compares two singly linked lists for equality. It should have the following signature. (You may use a recursive or an iterative solution).

```
bool is_equal(SinglyLinkedListNode<T> *lhs, SinglyLinkedListNode<T> *rhs)
```

**Solution:**

```
bool is_equal(SinglyLinkedListNode<T> *lhs,
              SinglyLinkedListNode<T> *rhs)
{
    // Base case
    if (lhs == nullptr && rhs == nullptr) return true;
    if (lhs == nullptr) return false;
    if (rhs == nullptr) return false;

    // Condition
    if (lhs->value != rhs->value) return false;

    // Recursion
    return is_equal(lhs->next, rhs->next);
}
```