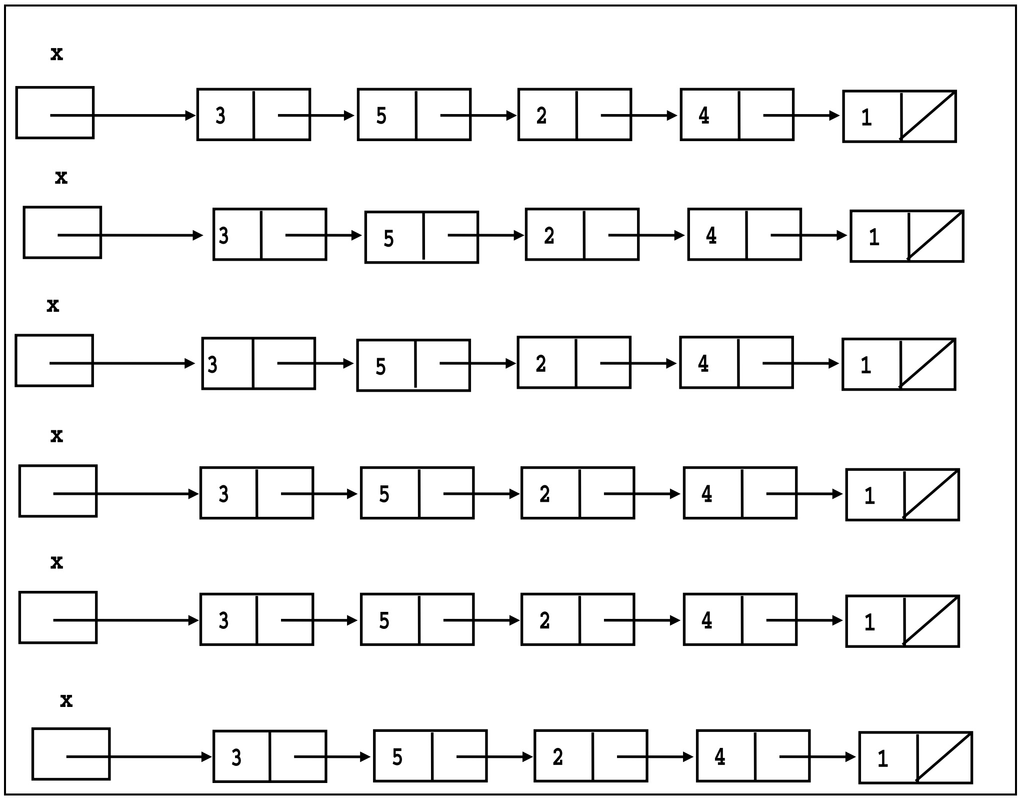
```
template<class T>
class LN {
  public:
    LN ()                              : next(nullptr){}
    LN (const LN<T>& ln)               : value(ln.value), next(ln.next){}
    LN (const T& v, LN<T>* n = nullptr) : value(v), next(n){}

    T      value;
    LN<T>* next;
};
```

Each linked lists is created from the class **LN<int>**. Starting with each list as shown, indicate what state(s) change(s) when the statement to its left is executed. **Cross out** any values that are replaced and **Write in** new boxes or values (text or arrows).

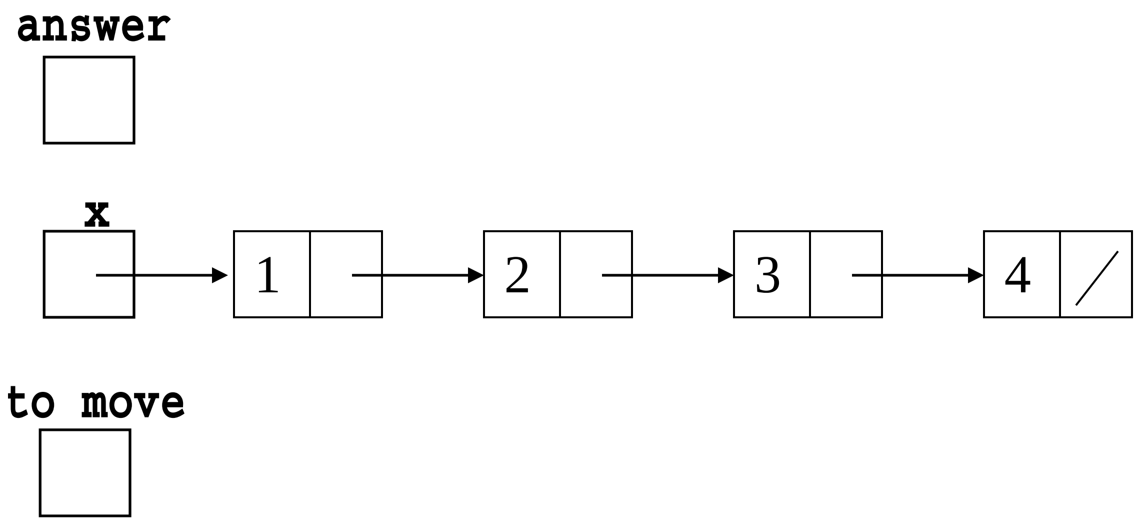
Hint: put a vertical stroke in the variable/ attribute (box) specified on the left side of the **=** which will receive the reference. Put a circle on the tail of the arrow specified on the right hand side of the **=**. Copy the value (reference) by making the vertical-stroked box refer to the object the circle-tailed arrow refers to. I will demonstrate in class.

```
x = x->next;

x->next->value = x->value;

x->next = x->next->next;

x->next->next = x->next;

x->next->next = x;

X = new LN<int>
     (7,x->next->next);
```



Assume **LN<int>* x** and **x** points to a linked list with 4 or 5 values. What changes are made when executing the following code?

```
LN<int>* answer = nullptr;
while (x != nullptr) {
    LN<int>* to_move  = x;
    x                 = x->next;
    to_move->next     = answer;
    answer            = to_move;
}

x = answer;
```


answer
x
1
2
3
4
to_move