Name: Inho Choi **ID:** 180 1080.

Problem 1 Linked List:

Complete the following function so that it deletes the middle Node from a doubly linked list. You may assume that the list passed to this function will have at least 3 Nodes and will have an odd number of Nodes. Write your answers corresponding to the lettered blanks on the right.

```
struct Node {
       int value;
       Node *next;
       Node *prev;
};
```

```
void deleteMiddle(Node * head, Node * tail) {
       Node* h = head;
       Node* t = tail;

       while (h->next != t->prev) {

              h = ___a.___;

              t = ___b.___;

       }

       Node* temp = ___c.___;

        ___d.___ = ___e.___;

        ___f.___ = ___g.___;

       delete ___h.___

}
```

a. h→next

b. t→prev

c. h→next

d. temp→prev→next

e. temp→next

f. temp→next→prev

g. temp→prev

h. temp

Name: Inho Choi **ID:** 1801180

Problem 2:

Below is code segment for a doubly linked list class that stores integers. The first node in the list has nullptr for its prev, and the last node in the list has nullptr for its next. Use the same Node definition provided in the previous problem/

```
class linkedlist {
public:
        ...
        void eraseSecondToLast();
private:
        Node *head;
};
```

The eraseSecondToLast function deletes one node just before the last node in the list with at least three nodes. You may assume that it will be called only for lists with at least three nodes. Below is an incomplete implementation for eraseSecondToLast, complete the implementation with **only the options given**, duplication of choices may occur. You may not include any additional lines or blanks.

```
void LinkedList::eraseSecondToLast() {
        Node *curr = ____head____;          // 1
        while (____curr->next____ != ____nullptr____)   // 2, 3
                ____curr____ = ____curr->next____;      // 4, 5
        Node *temp = ____curr->prev____;    // 6
        ____curr->prev->prev->next____ = ____curr____;  // 7, 8
        ____curr->prev____ = ____curr->prev->prev____;  // 9, 10
        delete ____temp____;                // 11
}
```

A. curr
B. curr->next
C. curr->next->next->next
D. curr->next->next->prev
E. curr->prev
F. curr->prev->next
G. curr->prev->prev
H. curr->prev->prev->next
I. head
J. head->next
K. nullptr
L. temp
M. temp->next->next

Note that the blanks above are for your convenience I will not look at those. **Write the letters corresponding to the filled-in code below**:

1	2	3	4	5	6	7	8	9	10	11
I	B	K	A	B	E	H	A	E	G	L