1.Describe the problem you solved. Include details, such as what the problem required, its constraints, what data structures needed to be used, and what memory needed to be allocated. Do not include any C code in your description.

The problem I solved during mock interview is 148. Sort List. This problem is asking to sort a linked list in ascending order. The number of nodes is between 0 to 5*10^4 and the value of each node is from -10^5 to 10^5. Since the number of node can be 5*10^4, the time complexity of our algorithm can not be O(n^2). The momory needed to be allocated is O(1), so we can only use constant space.

2.Describe the solution that you coded for the interview. How did you solve the problem? What tradeoffs does your solution make? Do not include any C code in your description. (4 points)

Since the constraints of the problem, I choose merge sort algorithm. Merge sort algorithm is to break up the given linked list into sub-linked list until there is only one node. Then merge each node into one linkedlist and sort them in place.

3.Provide the C code for your solution.

Code:

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* merge(struct ListNode* l1, struct ListNode* l2){
    struct ListNode dummyHead;
    struct ListNode* tail = & dummyHead;
    while(l1 != NULL && l2!= NULL){
        if(l1->val < l2->val){
            tail->next =l1;
            l1 = l1->next;
            tail = tail->next;
        }else{
            tail->next =l2;
            l2 = l2->next;
            tail = tail->next;
        }
    }
    if(l1!=NULL) tail->next = l1;
    if(l2!=NULL) tail->next = l2;
    return dummyHead.next;
}

struct ListNode* getMid(struct ListNode* head){
    struct ListNode* slow = head;
    struct ListNode* fast = head;
    struct ListNode* prev = NULL;
    while(fast!=NULL && fast->next!=NULL){
        prev = slow;
        slow = slow->next;
        fast = fast->next->next;
    }
    if(prev != NULL){
        prev -> next = NULL;
    }
    return slow;
}

struct ListNode* sortList(struct ListNode* head) {
    if(head==NULL||head->next ==NULL) return head;
    struct ListNode* mid = getMid(head);
    struct ListNode* left = sortList(head);
    struct ListNode* right = sortList(mid);
    return merge(left,right);
}
```

4.Describe the big O of the solution you wrote and explain why it has that big O. Count the operations (in terms of n); if it is a recursive solution, provide the recurrence equation and its big O.

The time complexity of merge sort is O(n*logn). The reason why it has this big O is that the given linkedlist has n nodes to sort. It takes logn time to split the linkedlist. So the total time complexity is n*logn.

The space complexity is O(logn) where n is the number of nodes in linked list. Since the problem is recursive, we need additional space to store the recursive call stack. The maximum depth of the recursion tree is logn.

Let's denote T(n) as the time required to sort an array of n elements with merge sort. The recursive equation for merge sort is:

$T(n) = 2T(n/2) + O(n)$

5.Is there a better solution to this problem than the one you provided? If so, describe it and explain how and why it is better than the solution you provided. If not, explain how you know this is the optimal solution.

We can also use a bottom up merge sort that split the linked list into the smallest list ad iteratively merge the result to solve the original problem. This appriach is solved iteratively and can be implemented using constant extra space.

6.What do you think you did well in the interview, and what (if anything) do you wish you had done differently?

I solved the coding part quickly but did not spend enough to explain the mechanism behind the algorithm.

7.What are your greatest strengths and assets in coding interviews and environments like them?

The greatest strengths and assets of mine is I can explain the code line by line while coding.

8.What do you think you will need to work on most to prepare for co-op or internship interviews?

I think I need to spend more time on how to explain why each method has its advantage and how to optimize the code, what is the trade off. Also be familiar with the time and space complexity of each method.

9.What is your plan for practicing and improving the aspect of your performance you identified for the future?

The plan is while solving more leetcode problems, I should also know what is the time and space complexity of the algorithm. Explain the problem line by line to myself after I solved the problem. Write comments at beginning of the problem and at each key part of the code.

Get more familiar with some most used searching and sorting algorithms.