This is the third lecture about linked list. In this lecture, we focus on the destroy function. The function deletes every node in a linked list.

This is a way to implement the destroy function.

This function takes only one argument. H is a pointer and it points to the first node of the linked list.

This function does not return anything. Thus, the return type is void.

Inside this function, it first checks whether H’s value is NULL or not. As explained several times already, almost every function related to linked list has to check whether it is NULL or not at the beginning of the function. If H is NULL, then H arrow next does not exist.

Inside this function, another pointer P points to the node after H. . We need to have two pointers. One points to the node that is about to be deleted. One points to the next node.

The next step free H. . This releases the memory occupied by the node.

The next step moves H to the next node. Earlier, P stores the address of the next node. By assigning P’s value to H’s value, H now points to the same node of the linked list as P. .

The program goes back to the top and checks whether H is NULL or not. H points to a valid node and it is not NULL.

Thus, the program enters while and sets P to be the next node after H. .

The next step frees H. . This releases the memory occupied by the node.

The next step moves H to the next node. H points to the same node as P ..

The program goes back to the top and checks whether H is NULL . or not. H points to a valid node and it is not NULL. .

Thus, the program enters while and sets P to be the next node after H. .

The next step frees H. . This releases the memory occupied by the node.

The next step moves H to the next node. H points to the same node as P ..

The program goes back to the top and checks whether H is NULL or not. H points to a valid node and it is not NULL.

Thus, the program enters while. and sets P to be the next node after H. . P points to NULL now. This means it is not a valid node.

The next step frees H. . This releases the memory occupied by the node.

The next step moves H to the next node. H points to the same node as P ..

Please notice that both P and H point to NULL.

The program gets out of while. and the memory of every node in the linked list has been freed.

The next few slides answer several frequently asked questions.

The first question is whether we need two pointers P and H. . Is it possible using only one pointer?

The answer is that we need two pointers. One pointer stores the address of the node that will be freed. After this node is freed, it does not have the next pointer any more. Thus, we must store the next node before freeing the node.

Another common question is whether we can move the definition of P outside while.? The answer is yes. You can move Node asterisk p outside while.

It is important to note that p’s value must be updated inside while. . P is the node after H. .

Another question is whether H needs to be updated inside while. . The answer is yes. After freeing H, we need to update H for the next iteration of while . .

After free H, what is the value of H? H’s value is unchanged. It still stores the address of the node that has already been deleted. Please notice that free H does not set H’s value to NULL.

Another question is whether we can change the order of these three lines. The answer is no. If we move free H first, then h arrow next does not exist and this program will have segmentation fault.

There are six possible orders of these three lines. Only one of the six orders is correct.

Let me explain what is wrong with the other orders.

The first is the correct order: it stores H’s next in P first. Then, free H. . Then, use P’s value to set H’s value.

Consider the second order. In this case, the second and the third lines are swapped. This will free a wrong node. Also, h arrow next is not valid in the next iteration.

Consider the third order. The first and the second lines are swapped. After free H, H arrow next does not exist and cannot be assigned to P. .

In the fourth order, the first line is moved to the bottom. The second line is problematic because P has not been initialized. Thus, the second line assigns an unknown value to H. . As a result, H arrow next is invalid.

Consider the fifth order. This moves the last line to the top. It has the same problem as the fourth case. P has not been initialized. This makes H’s value unknown. H arrow next is invalid.

The sixth order has the same problem. P has not been initialized. This makes H’s value unknown. Free H is invalid.

Thus, the three lines must follow this order. You must not reorder them.