**Name:** Safyan Anwar

**Roll no:** S24-018

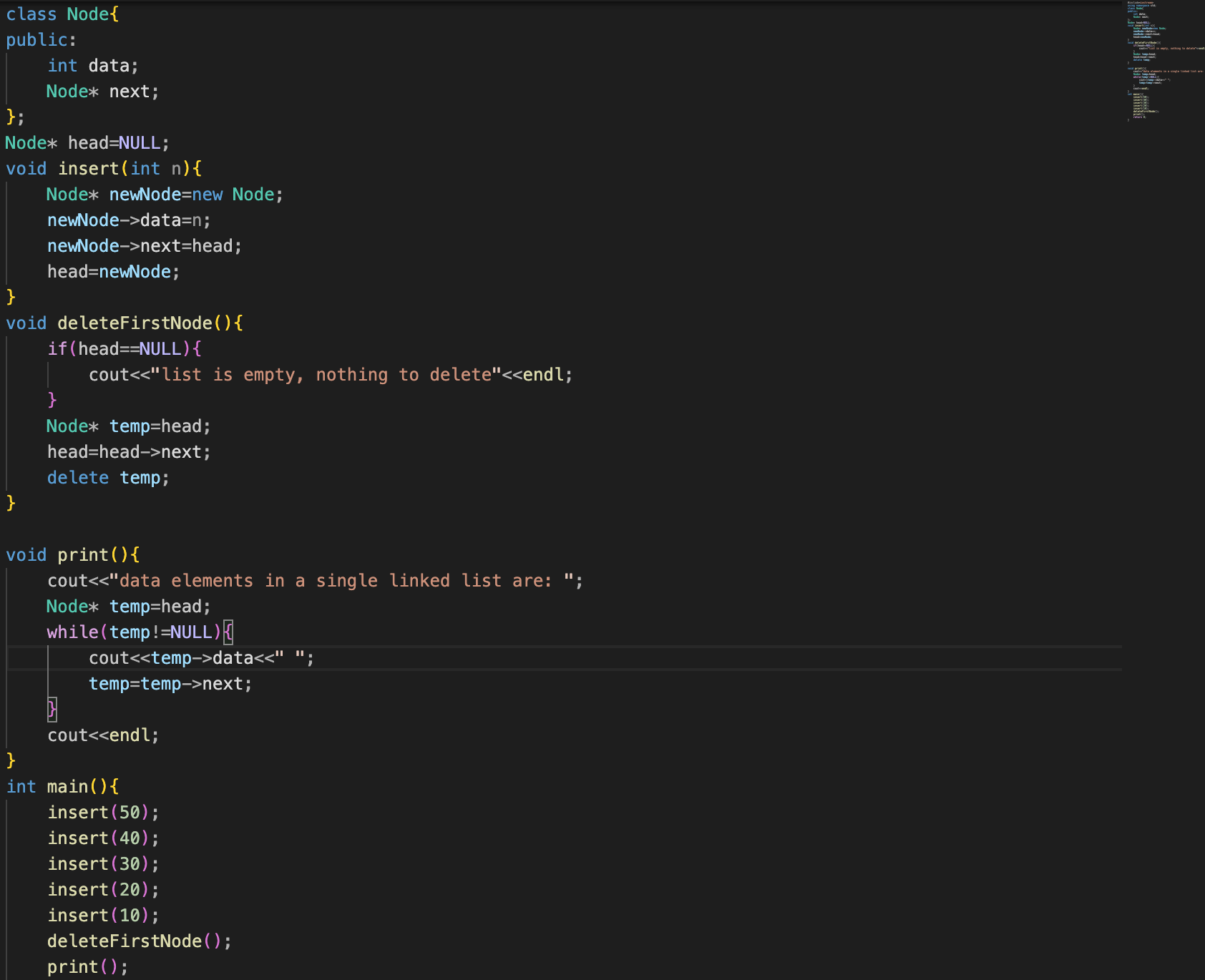
**Section:** 3A-BSSE

**Assignment no:** 6

**Submitted to:** Sir. Rasikh

**LAB TASK:**

**Code: “**delete\_at\_first**”:**

****

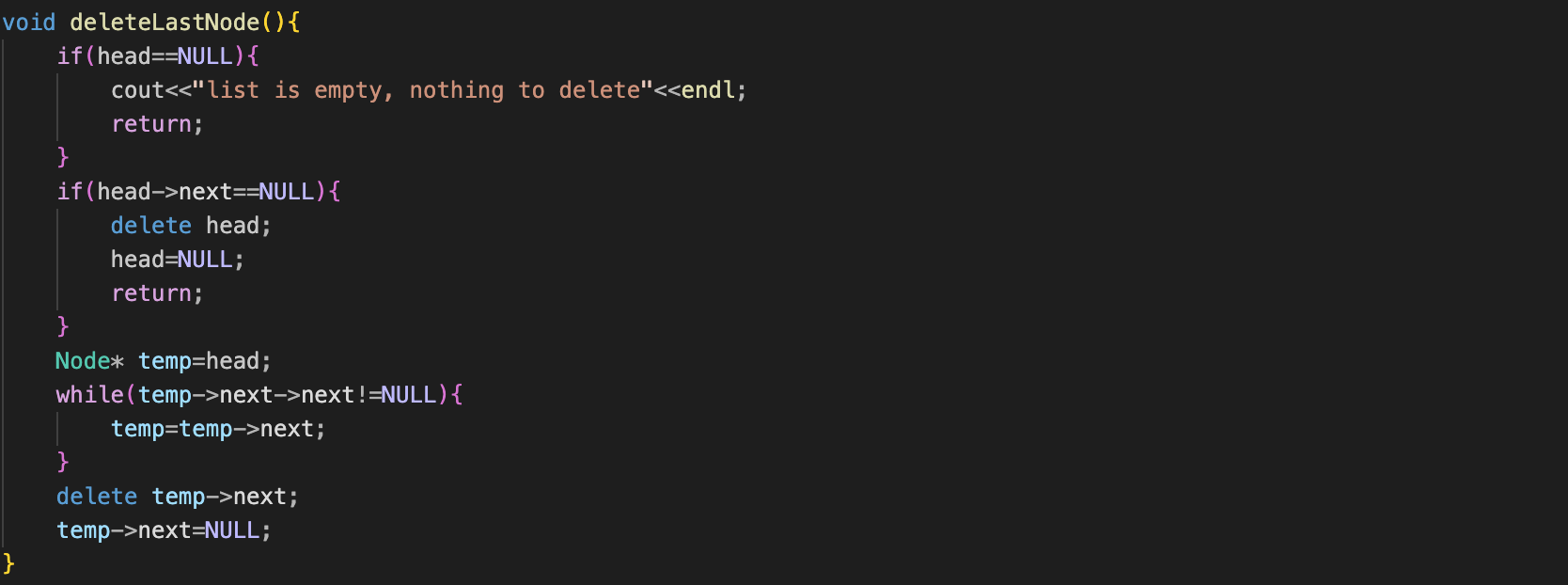
**Output:**

****

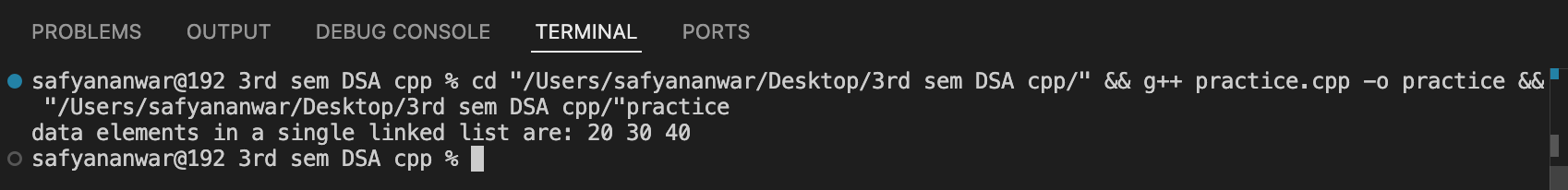
**Explain:**

To delete the first node, we checked that the list is not empty. If the list is empty, a message will be printed on the screen. But the list is not empty. To delete the first node, we will create a temp pointer, store the memory address of the first node in it, then update the head, and then delete the temp, and then call it in the main function.

**Code: “**delete\_at\_last**”:**

****

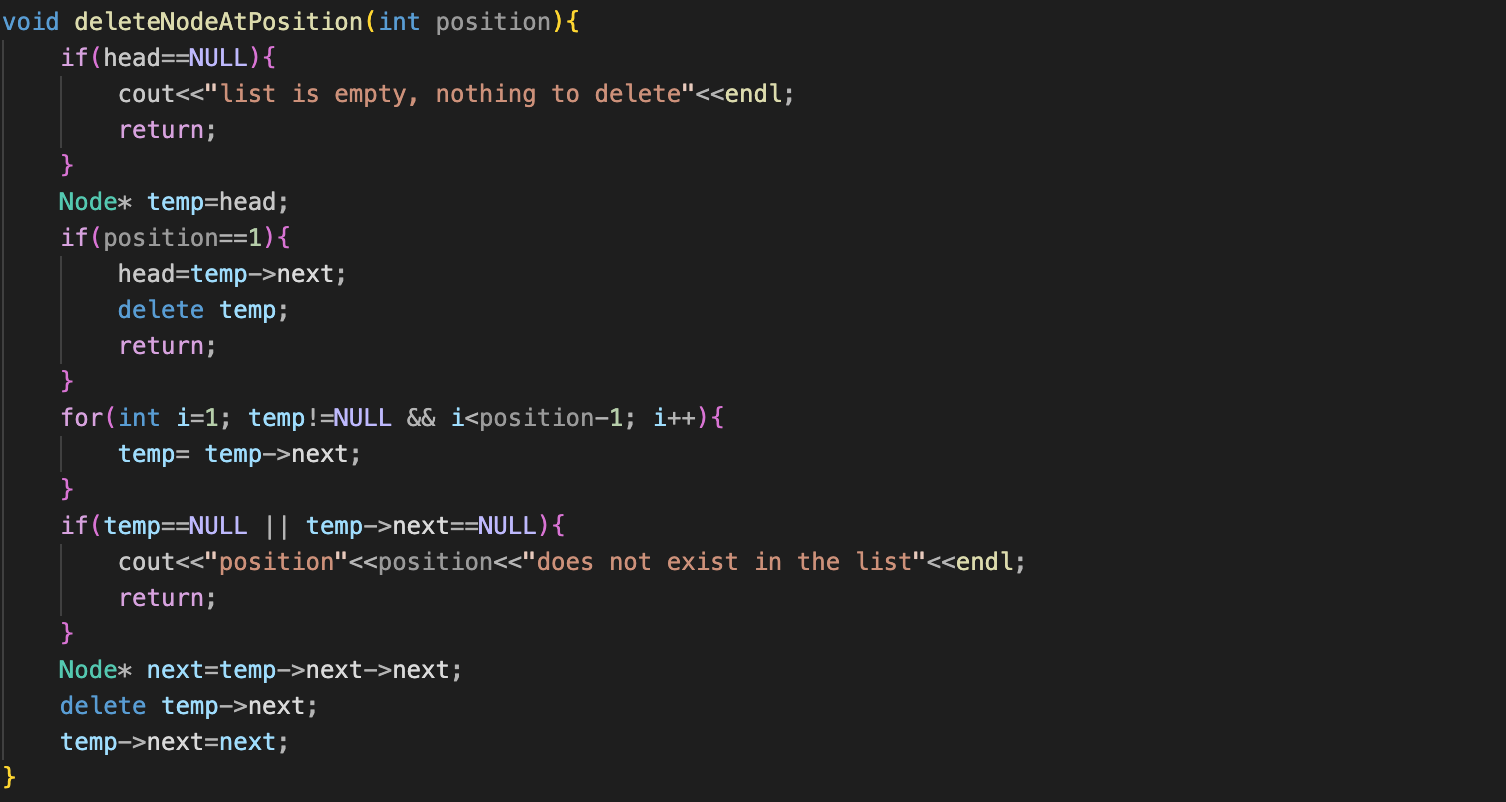
**Output:**

****

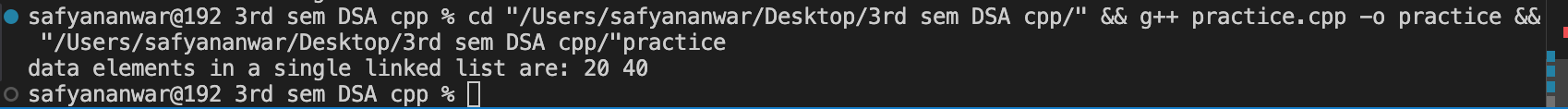
**Explain:**

To delete the last node, we checked that the list is not empty. If the list is empty, a message will be printed on the screen. If there is only one node in the list it delete that node and sets head to NULL, effectively making the list empty. If there are multiple nodes, it initializes a temporary pointer temp to head and traverses the list until it reached the second to last node. Once it finds the second to last node, it delete the last node and sets it to NULL, effectively removing the last node from the list and then call it in the main function.

**Code: “**delete\_at\_specificlocation**”:**

****

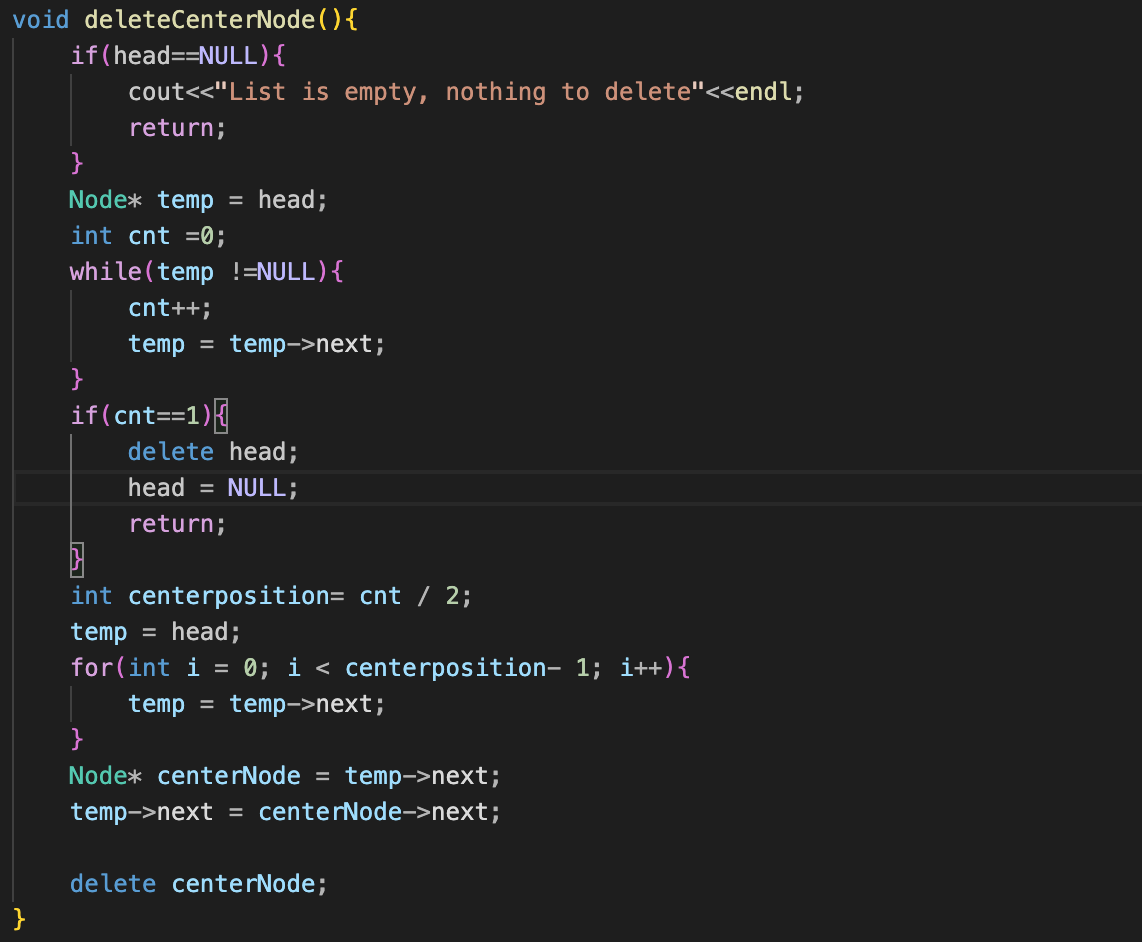
**Output:**

****

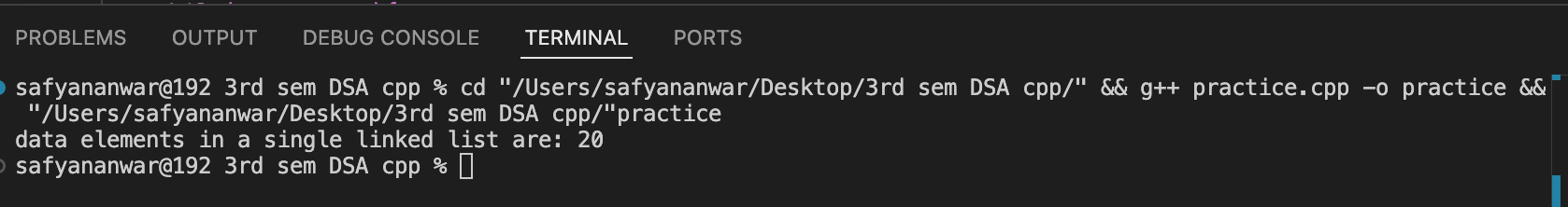
**Explain:**

To delete the specific position node, we checked that the list is not empty. If the list is empty, a message will be printed on the screen. If the position is 1 the node to delete is the first node in the list the function adjusts the head pointer to point to the second node. The current head node is then deleted using delete temp. After the loop, it checked if temp is NULL or if temp->next is NULL. If either is true it mean the specified position does not exist in the list and it print an error message. If the position is valid it stores the pointer to the node that is to be deleted and updates the next pointer of temp to skip over the node being deleted. Finally, it deleted the node and then call it in the main function.

**Code: “**delete\_at\_center**”:**

****

**Output:**

****

**Explain:**

To delete the center position node, we checked that the list is not empty. If the list is empty, a message will be printed on the screen. Counting the total number of nodes. Determining the position of the center node. Traversing to the node just before the center node. Updating the pointers to by the center node and delete it.