**Input: For single linked list.**

#include <iostream>

struct Node {

int data;

Node\* next;

};

void append(Node\*\* head\_ref, int new\_data) {

Node\* new\_node = new Node();

Node\* last = \*head\_ref;

new\_node->data = new\_data;

new\_node->next = nullptr;

if (\*head\_ref == nullptr) {

\*head\_ref = new\_node;

return;

}

while (last->next != nullptr) {

last = last->next;

}

last->next = new\_node;

}

void deleteLastOccurrence(Node\*\* head\_ref, int key) {

Node \*temp = \*head\_ref, \*last\_occurrence = nullptr;

Node \*prev = nullptr, \*last\_occurrence\_prev = nullptr;

while (temp != nullptr) {

if (temp->data == key) {

last\_occurrence\_prev = prev;

last\_occurrence = temp;

}

prev = temp;

temp = temp->next;

}

if (last\_occurrence != nullptr) {

if (last\_occurrence\_prev != nullptr) {

last\_occurrence\_prev->next = last\_occurrence->next;

} else {

\*head\_ref = last\_occurrence->next;

}

delete last\_occurrence;

}

}

void printList(Node\* node) {

while (node != nullptr) {

std::cout << node->data << " ";

node = node->next;

}

std::cout << std::endl;

}

int main() {

Node\* head = nullptr;

append(&head, 1);

append(&head, 2);

append(&head, 3);

append(&head, 2);

append(&head, 4);

append(&head, 2);

std::cout << "Original list: ";

printList(head);

int key = 2;

deleteLastOccurrence(&head, key);

std::cout << "List after deleting last occurrence of " << key << ": ";

printList(head);

return 0;

}

