#include<iostream>

#include<string>

using namespace std;

//declare a structure with data and address field

struct Node{

string data;

Node \*next;

};

// functions to perform three operations on the linked lists with head node mentioned

void addToList(Node \*head);

void deleteFromList(Node \*head);

void printList(Node \*head);

int main(){

bool quit = false;

int choice;

Node \*head = new Node;

head->next = NULL;

while (!quit){

cout << "1. add to list" << endl

<< "2. delete from list" << endl

<< "3. print list" << endl

<< "4. quit" << endl;

cin >> choice;

switch(choice){

case 1: addToList(head);

break;

case 2: deleteFromList(head);

break;

case 3: printList(head);

break;

case 4: quit = true;

break;

default:

cout << "That is not a valid input, quitting program";

quit = true;

}

}

}

void deleteFromList(Node \*head){

string deletion;

cout << "Which value do you want to delete from the list? ";

cin >> deletion;

Node \*prev = head;

Node \*current = head->next;

while (current)

{

if (current->data == deletion){

prev->next = current->next;

delete current;

return;

}

prev = current;

current = current->next;

}

if (!current){

cout << "That value is not in the list" << endl;

}

}

void printList(Node \*head){

// head->next, because the head is an empty node in your implementation,

if (!head->next)

{

cout << "Nothing is in the list." << endl;

return;

}

Node \*current;

// set current to head->next, because the head is empty in your implementation:

current = head->next;

while (current)

{

cout << current->data << endl;

current = current->next;

}

}

void addToList(Node \*head){

bool quit = false;

string temp;

Node \*current;

while (!quit){

cout << "Enter a word(quit to stop)";

cin >> temp;

if (temp == "quit"){

quit = true;

}

else{

// Allocate the new node here:

current = new Node;

current->data = temp;

// the new node is inserted after the empty head

// because head is an empty node in this implementation:

current->next = head->next;

head -> next = current;

}

}

return;

}