#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct contact

{

char name[50];

char email[50];

char numbr[50];

};

struct linkedlist{

int data;

struct linkedlist\* next;

};

struct Node {

int data;

struct Node\* next;

};

struct Node\* mergeSortedLists(struct Node\* list1, struct Node\* list2) {

struct Node\* mergedList = NULL;

struct Node\* tail = NULL;

while (list1 != NULL && list2 != NULL) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Memory allocation error\n");

exit(EXIT\_FAILURE);

}

if (list1->data <= list2->data) {

newNode->data = list1->data;

list1 = list1->next;

} else {

newNode->data = list2->data;

list2 = list2->next;

}

newNode->next = NULL;

if (mergedList == NULL) {

mergedList = newNode;

tail = newNode;

} else {

tail->next = newNode;

tail = newNode;

}

}

// If there are remaining nodes in either list

if (list1 != NULL) {

if (mergedList == NULL) {

mergedList = list1;

} else {

tail->next = list1;

}

} else if (list2 != NULL) {

if (mergedList == NULL) {

mergedList = list2;

} else {

tail->next = list2;

}

}

return mergedList;

}

void printList(struct Node\* head) {

while (head != NULL) {

printf("%d ", head->data);

head = head->next;

}

printf("\n");

}

void insertEnd(struct Node\*\* head, int value) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Memory allocation error\n");

exit(EXIT\_FAILURE);

}

newNode->data = value;

newNode->next = NULL;

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

}

void freeList(struct Node\* head) {

struct Node\* temp;

while (head != NULL) {

temp = head;

head = head->next;

free(temp);

}

}

int main(int argc, char const \*argv[])

{

//exercise 1

struct contact\* addressbook=(struct contact\*) malloc(30 \* sizeof(struct contact) );

int cont\_num=0;

while (1){

printf("choose:\n1.to insert a contact\n2.to delete a contact\n3.to exit.");

int choice;

scanf("%d",&choice);

if (choice == 1){

(cont\_num)++;

addressbook=realloc(addressbook,100\*sizeof(struct contact));

if (addressbook == NULL){

printf("memory not allocted");

break;

}

printf("enter name:");

scanf("%s",addressbook->name);

printf("enter email:");

scanf("%s",addressbook->email);

printf("enter phone number:");

scanf("%s",addressbook->numbr);

printf("contact saved successfully");

}

else if (choice == 2) {

char delnum[50];

printf("enter phone number to be deleted:");

scanf("%s",delnum);

for (int i=0;i<=cont\_num;i++){

if (addressbook->numbr==delnum){

for (int j=i;j<=cont\_num;j++){

strcpy((addressbook)[j].name, (addressbook)[j + 1].name);

strcpy((addressbook)[j].email, (addressbook)[j + 1].email);

strcpy((addressbook)[j].numbr, (addressbook)[j + 1].numbr);

}

}

}

}

else if (choice == 3){

break;

}

}

free(addressbook);

//exercise 3

int count = 1;

struct linkedlist\* head = (struct linkedlist\*)malloc(sizeof(struct linkedlist));

head->next = NULL;

while (1) {

int check = 0;

printf("Enter 1 to enter data in linked list or 0 to exit: ");

scanf("%d", &check);

if (check == 1) {

struct linkedlist\* node = (struct linkedlist\*)malloc(sizeof(struct linkedlist));

printf("Enter number data for linked list: ");

scanf("%d", &node->data);

node->next = head->next;

head->next = node;

count++;

} else {

break;

}

}

int\* array = (int\*)malloc(count \* sizeof(int));

struct linkedlist\* current = head->next;

for (int i = 0; i < count; i++) {

array[i] = current->data;

current = current->next;

}

for (int i = 0; i < count; i++) {

printf("%d ", array[i]);

}

//exercise 4

struct linkedlist\* odd = (struct linkedlist\*)malloc(sizeof(struct linkedlist));

odd->next = NULL;

struct linkedlist\* curr = odd;

for (int j = 0; j <= 50; j++) {

curr->data = j;

curr->next = (struct linkedlist\*)malloc(sizeof(struct linkedlist));

curr = curr->next;

}

curr->next = NULL;

curr = odd;

struct linkedlist\* temp;

while (curr->next != NULL && curr->next->next != NULL) {

temp = curr->next;

curr->next = curr->next->next;

free(temp);

curr = curr->next;

}

curr = odd;

while (curr->next != NULL) {

printf("%d ", curr->data);

curr = curr->next;

}

//exercise 2

struct Node\* list1 = NULL;

struct Node\* list2 = NULL;

insertEnd(&list1, 1);

insertEnd(&list1, 3);

insertEnd(&list1, 5);

insertEnd(&list2, 2);

insertEnd(&list2, 4);

insertEnd(&list2, 6);

printf("List 1: ");

printList(list1);

printf("List 2: ");

printList(list2);

struct Node\* mergedList = mergeSortedLists(list1, list2);

printf("Merged List: ");

printList(mergedList);

freeList(list1);

freeList(list2);

freeList(mergedList);

return 0;

}