Computer Programming — Quiz 2 Solutions

Instructor: Girish Varma • Course Code: CS0.101 • IIIT Hyderabad

Problems are in comments.

```c
#include <stdio.h>
#include <stdlib.h>

#include <string.h>
// defines strcmp(first_str, second_str);
// if `first_str` is equal to `second_str`
// returs 0. Otherwise returns a nonzero value

#include <stdbool.h>
// defines type `bool`and macros `true` and `false`

typedef enum RelStatus {
    NotMentioned,
    Single,
    Engaged,
    Married
} RelStatus;

typedef struct Node Node;

typedef Node* LinkedList;

typedef struct Person {
    char name[100];
    int age;
    RelStatus relstatus;
    LinkedList friends;
} Person;

struct Node {
    struct Person* data;
    struct Node* next;
};

typedef struct SocialNet {
    LinkedList members;
} SocialNet;

LinkedList append(Person* p, LinkedList l) {
    if (l == NULL) {
        Node* D = (Node *) malloc(sizeof(Node));
        D->data = p;
        D->next = NULL;
        return D;
    } else {
        l->next = append(p, l->next);
    }
    return l;
}

int size(LinkedList l) {
    return l==NULL? 0: 1+ size(l->next);
}


Person* find_person_by_name(char* name, LinkedList l) {
    // Q1: Return the pointer to the Person with name
    // given by argument `name` in the LinkedList `l`
    // (10 marks)

    // Solution:
    while (l != NULL) {
        if (strcmp(l->data->name, name) == 0) {
            return l->data;
        }
        l = l->next;
    }
    return NULL;
}

bool common_single_friend(char* name1, char* name2,
                    SocialNet* s) {
    // Q2: Check if the Persons with name = name1
    // and name =  name2 has a common friend who
    // is Single. Return `true` or `false`
    //  (10 marks)

    // Solution:
    Person* person1 = find_person_by_name(name1, s->members);
    Person* person2 = find_person_by_name(name2, s->members);

    if (person1 == NULL || person2 == NULL) {
        return false;
    }
    LinkedList friends1 = person1->friends;
    while (friends1 != NULL) {
        Person* friend1 = friends1->data;
        if (friend1->relstatus == Single) {
            LinkedList friends2 = person2->friends;
            while (friends2 != NULL) {
                if (friends2->data == friend1) {
                    return true;
                }
                friends2 = friends2->next;
            }
        }
        friends1 = friends1->next;
    }
    return false;
}

char* most_popular_person(SocialNet* s) {
    // Q3: Return the name of the person who is in the
    // friends list of most number of people
    // (15 marks)

    // Solution:

    int max_friends = -1;
    char* most_popular = NULL;

    LinkedList members = s->members;
    while (members != NULL) {
        Person* current = members->data;
        int friend_count = 0;

        LinkedList all_members = s->members;
        while (all_members != NULL) {
            LinkedList friends = all_members->data->friends;
            while (friends != NULL) {
                if (friends->data == current) {
                    friend_count++;
                    break;
                }
                friends = friends->next;
            }
            all_members = all_members->next;
        }
        if (friend_count > max_friends) {
            max_friends = friend_count;
            most_popular = current->name;
        }
        members = members->next;
    }
    return most_popular;
}

bool all_members_with_only_two_young_friends(
                SocialNet* s, int age_upper) {
    // Q4: Check if all members in the social
    // network `s` have exactly two friends
    // whose age is <= `age_upper`.
    // Return `true` or `false. (15 marks)

    // Solution:
    LinkedList members = s->members;
    while (members != NULL) {
        int young_friends_count = 0;
        LinkedList friends = members->data->friends;
        while (friends != NULL) {
            if (friends->data->age <= age_upper) {
                young_friends_count++;
            }
            friends = friends->next;
        }
        if (young_friends_count != 2) {
            return false;
        }
        members = members->next;
    }
    return true;
}

int main() {
    SocialNet* s = (SocialNet*) malloc(sizeof(SocialNet));
    // Sample data setup
    Person alice = {"Alice", 25, Single, NULL};
    Person bob = {"Bob", 30, Married, NULL};
    Person charlie = {"Charlie", 22, Single, NULL};
    Person david = {"David", 35, Engaged, NULL};
    Person bender = {"Bender", 28, Single, NULL};
    // Set up friends
    alice.friends = append(&bob, alice.friends);
    alice.friends = append(&charlie, alice.friends);
    bob.friends = append(&alice, bob.friends);
    bob.friends = append(&david, bob.friends);
    charlie.friends = append(&alice, charlie.friends);
    charlie.friends = append(&bender, charlie.friends);
    david.friends = append(&bob, david.friends);
    david.friends = append(&bender, david.friends);
    bender.friends = append(&charlie, bender.friends);
    bender.friends = append(&david, bender.friends);

    // Create social network
    SocialNet network = {NULL};
    network.members = append(&alice, network.members);
    network.members = append(&bob, network.members);
    network.members = append(&charlie, network.members);
    network.members = append(&david, network.members);
    network.members = append(&bender, network.members);

    // Q1: find_person_by_name
    printf("Q1 Example 1: %s\n", find_person_by_name(
        "Alice", network.members)->name);
    printf("Q1 Example 2: %s\n", find_person_by_name(
        "David", network.members)->name);
    printf("Q1 Example 3: %s\n", find_person_by_name(
        "Frank", network.members) == NULL ? "NULL" : "Not NULL");

    // Q2: common_single_friend
    printf("Q2 Example 1: %s\n", common_single_friend(
        "Alice", "Bob", &network) ? "true" : "false");
    printf("Q2 Example 2: %s\n", common_single_friend(
        "Bob", "David", &network) ? "true" : "false");
    printf("Q2 Example 3: %s\n", common_single_friend(
        "Charlie", "Eve", &network) ? "true" : "false");
    printf("Q2 Example 4: %s\n", common_single_friend(
        "Charlie", "David", &network) ? "true" : "false");

    // Q3: most_popular_person
    printf("Q3 Result: %s\n", most_popular_person(&network));

    // Q4: all_members_with_only_two_young_friends
    printf("Q4 Example 1 (age_upper = 25): %s\n",
        all_members_with_only_two_young_friends(&network, 25) ? "true" : "false");
    printf("Q4 Example 2 (age_upper = 30): %s\n",
        all_members_with_only_two_young_friends(&network, 30) ? "true" : "false");
    printf("Q4 Example 3 (age_upper = 35): %s\n",
        all_members_with_only_two_young_friends(&network, 35) ? "true" : "false");

    return 0;
}
```