B. Khyathi Sri
2023102065

Computer Programming : Section B

Quiz 2

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

Problems are in comments, highlighted.

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

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;
}

void print_person(Person* p) {
    char status_string[][15] = {
        "Not Mentioned", "Single", "Married", "Engaged"
    };
    printf("%s\t\t%d\t%s\t\t\t",
           p->name, p->age, status_string[p->relstatus]);
    LinkedList f = p->friends;
    while (f != NULL) {
        printf("%s, ", f->data->name);
        f = f->next;
    }
    printf("\n");
}

void print_network(LinkedList m) {
    printf(
"-----------------------------------------------\n"
"Name\t\tAge\tStatus\t\t\tFriends\n"
"-----------------------------------------------\n");
    while (m != NULL) {
        print_person(m->data);
        m = m->next;
    }
    printf(
"-----------------------------------------------\n");
}

Person* find_person(char* name, LinkedList l) {
    // Either find the person with a particular name
    // if not found return NULL
    while(l!= NULL) {
        if (strcmp(l->data->name, name) == 0) {
            return l->data;
        }
        l = l->next;
    }
    return NULL;
}

int popularity(char* name, LinkedList l) {
    // Q1: Return the number of people who has the person
    // named `name` amoung their friends. (3 marks)
}

LinkedList filterby_age(LinkedList l, int lower, int upper) {
    // Q2: Return the link list of people in l with age
    // between lower and upper (3 marks)
}

bool transitive_friendship(LinkedList members) {
    // Q3: check if the friendship relation is transitive
    // ie for any X,Y, Z, if Y is a friend of X and
    // Z is a friend of Y then Z is a friend of X
    // Also print all the links that violates transitivity
    // (4 marks)
}S

int main()
{
    SocialNet s = { NULL };
    Person A = {"Alice", 23, Single, NULL};
    Person B ={"Bob", 26, Engaged, NULL};
    Person C = {"Charlie", 21, NotMentioned, NULL};
    Person D ={"Don", 28, Married, NULL};

    s.members = append(&A, s.members);
    s.members = append(&B, s.members);
    s.members = append(&C, s.members);
    s.members = append(&D, s.members);

    A.friends = append(&B, A.friends);
    A.friends = append(&C, A.friends);
    B.friends = append(&D, B.friends);
    C.friends = append(&D, C.friends);
    D.friends = append(&A, D.friends);

    // prints
    // -----------------------------------------------
    // Name         Age     Status          Friends
    // -----------------------------------------------
    // Bob          26      Married         Don,
    // Don          28      Engaged         Alice,
    // -----------------------------------------------
    print_network(filterby_age(s.members, 24, 28));

    // For the above social network,
    // `transitive_friendship(s.members)`
    // returns false and prints
    // ----------------------------
    // Links that are not Transitive
    // ----------------------------
    // Alice->Bob->Don, but there is no Alice->Don
    // Alice->Charlie->Don, but there is no Alice->Don
    // Bob->Don->Alice, but there is no Bob->Alice
    // Charlie->Don->Alice, but there is no Charlie->Alice
    // Don->Alice->Bob, but there is no Don->Bob
    // Don->Alice->Charlie, but there is no Don->Charlie
    // ------------------------
    transitive_friendship(s.members);

    return 0;
}
```