03 Macros, More Intializers in Social Nets

Social Nets

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

typedef struct Person {
    char name[100];
    int age;
    RelStatus relstatus;
    struct Person* friends[5];
```

```
} Person;

typedef struct SocialNet {
    Person members[100];
    int size;
} SocialNet;
```

Social Nets with Macros

```
#define MAX_FRIENDS 5
#define MAX_MEMBERS 100
#define MAX_NAME_LEN 100

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

typedef struct Person {
    char name[MAX_NAME_LEN];
    int age;
    RelStatus relstatus;
    struct Person* friends[MAX_FRIENDS];
} Person;

typedef struct SocialNet {
    Person members[MAX_MEMBERS];
```

```
    int size;
} SocialNet;
```

Intitializer

Name	Age	Rel Status
Alice	24	Not Mentioned
Bob	28	Maried
Charlie	20	Single

```
int main() {
    SocialNet social_net = {
        .members = {
            { "Alice",    24,  NotMentioned},
            { "Bob",      28,  Married},
            { "Charlie",  20,  Single},
        } ,
        .size = 3
    };
    print_network(social_net);
    return 0;
}
```

Print Person

```
void print_person(struct Person p) {
    // TODO (solution at the end of page)
}

void print_network(SocialNet social_net) {
    printf(
        "-----------------------------------------------\n"
        "Name\t\tAge \t Rel Status\n"
        "-----------------------------------------------\n");
    for (int i=0;i <social_net.size; i++) {
        print_person(social_net.members[i]);
    }
    printf("-----------------------------------------------\n");
}
```

Finding a person by name

```
Person* find_person(char* name1, SocialNet *sn) {
    // TODO (solution at the end of page)
}
```

HW: Check Mutual Friends by name

```
bool check_mutual_friends(char *name1, char *name2, SocialNet *sn) {
 // TODO p and q are mutual friends if q is in the friend list of p
 // and p is in the friend list of q
}
```

Full Code with Solutions

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

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

typedef struct Person {
    char name[100];
    int age;
    RelStatus relstatus;
    int count_friends;
    struct Person *friends[5];
} Person;

typedef struct SocialNet {
    Person members[80];
    int size;
} SocialNet;

Person* find_person(char* name1, SocialNet *sn) {
    for(int i = 0; i < sn->size; i++) {
```

```
        if ( strcmp(sn->members[i].name, name1) == 0) {
            return &(sn->members[i]);
        }
    }
    return NULL;
}

void print_person(Person p) {
    char status_string[][15] = {
        "Not Mentioned",
        "Single",
        "Married",
        "Engaged"
    };
    printf("%s\t\t%d\t%s\n", p.name, p.age, status_string[p.relstatus]);
}

void print_network(SocialNet social_net) {
    printf(
        "-----------------------------------------------\n"
        "Name\t\tAge \t Rel Status\n"
        "-----------------------------------------------\n");
    for (int i=0;i <social_net.size; i++) {
        print_person(social_net.members[i]);
    }
    printf("-----------------------------------------------\n");
}

int main()
{
    SocialNet social_net = {
        .members = {
            { "Alice",    24,  NotMentioned},
```

```
            { "Bob",      28,  Married},
            { "Charlie",  20,  Single},

        } ,
        .size = 3
    };
    print_network(social_net);
    print_person(*find_person("Bob", &social_net));
    return 0;
}
```