Social Network

Name	Age	Rel Status	Friends
Alice	24	Single	Diestel, Eve
Bob	28	Maried	Alice
Charlie	20	Single	Diestel
Diestel	27	Not Mentioned	Alice, Eve, Charlie
Eve	25	Engaged	Diestel, Alice

Define a Person (Profile)

```
struct Person {
    char name[100];
    int age;
    int rel_status;
};
```

Implemeting Rel Status as int, requires us to keep in mind the mapping between Single, Maried, Not Mentioned, Engaged and integers.

{{<hint info>}}
Can we specify this in code??
{{</hint>}}

Enums

```
typedef enum Weekday {
    Sunday,
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday
} Weekday;
```

```
Weekday today = Wednesday;
printf("Day %d",today+1);
printf("Size of enum variable = %d bytes",
        sizeof(today));
```

Enums : Changing default values

```
typedef enum Weekday {
    Sunday = 1,
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday,
    Saturday
} Weekday;
```

```
Weekday today = Wednesday;
printf("Day %d",today+1);
```

Enums : interchangable with int

```c
#include "stdio.h"

typedef enum Weekday {
    Sunday = 5,
    Monday = 3,
    Tuesday,
    Wednesday = 2,
    Thursday,
    Friday,
    Saturday
} Weekday;

int main() {
    Weekday today = Wednesday;;
    printf("Day %d\n",today+1);
    printf("Size of enum variable = %d bytes",
             sizeof(today));
    return 0 ;
}
```

Define a Person (Profile)

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

struct Person {
    char name[100];
    int age;
    enum RelStatus status;
};
```

Social Nets

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

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

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

Intitializer

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

```c
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");
}
```

Full Code for Social Net

HomeWork: change the code to print all the names of friends of the person in 4th column

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

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


typedef struct Person {
        char name[100];
        int age;
        RelStatus status;
        int friends[10];
        int friends_size;
} Person;

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


void print_person(Person* p) {
        char status_string[4][30] = {
                { "Not Mentioned"},
                { "Single" },
                { "Engaged"},
                { "Married" }
        };
        printf("%s\t%d\t%s\n", p->name, p->age, status_string[p->status]);
}

void print_socialnet(SocialNet *s) {
        printf("-------------------------------\n");
        printf("Name\tAge\tRelationship Status\tFriends\n");
        printf("-------------------------------\n");
        for (int i = 0; i < s->size; i++ ) {
                // HomeWork: change the code to print all the names of friends of the person in 4th column
                print_person(&(s->members[i]));
        }
        printf("-------------------------------\n");
}

int main() {
        SocialNet s = {
                .members = {
                        { "Ramu", 19, Single, {1,2}, 2},
```

Commandline Argument

An easier way to take input from the user in shell.

Commandline Argument

```c
#include <stdio.h>
int main(int argc, char* argv[]) {
    printf("The number of arguments is %d\n", argc);

    for (int i = 0; i < argc; i++) {
        printf("%d Argument: %s\n", i, argv[i]);
    }
    return 0;
}
```

Problem

Write a program that takes the First Name Last Name Age as commandline arguments and prints it as follows

First Name: <first arg>

Last name : <sec arg>

Age : <third arg>

Solution

```c
#include "stdio.h"

int main(int argc, char* argv[]) {

    if (argc != 4) {
        printf("Incorrect number of arguments provided.\n");
        return 0;
    }

    printf("First Name:\t%s\n", argv[1]);
    printf("Last Name :\t%s\n", argv[2]);
    printf("Age       :\t%s\n", argv[3]);
    return 0;
 }
```