

# COMPUTER PROGRAMMING : SECTION A

## QUIZ 2

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

Problems are in comments, highlighted.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <stdbool.h>
5
6  typedef enum RelStatus {
7      NotMentioned,
8      Single,
9      Engaged,
10     Married
11 } RelStatus;
12
13 typedef struct Node Node;
14
15 typedef Node* LinkedList;
16
17 typedef struct Person {
18     char name[100];
19     int age;
20     RelStatus relstatus;
21     LinkedList friends;
22 } Person;
23
24 struct Node {
25     struct Person* data;
26     struct Node* next;
27 };
28
29 typedef struct SocialNet {
30     LinkedList members;
31 } SocialNet;
32
33 LinkedList append(Person* p, LinkedList l) {
34     if (l == NULL) {
35         Node* D = (Node *) malloc(sizeof(Node));
36         D->data = p;
37         D->next = NULL;
38         return D;
39     } else {
40         l->next = append(p, l->next);
41     }
42     return l;
43 }
44
45 void print_person(Person* p) {
46     char status_string[15] = {
47         "Not Mentioned",
48         "Single",
49         "Married",
50         "Engaged"
51     };
52     printf("%s\t\t%d\t\t%s\t\t\t",
53         p->name, p->age, status_string[p->relstatus]);
54     LinkedList f = p->friends;
55     while (f != NULL) {
56         printf("%s, ", f->data->name);
57         f = f->next;
58     }
59     printf("\n");
60 }
61
62 void print_network(LinkedList m) {
63     printf(
64         "-----\n"
65         "Name\t\tAge\t\tStatus\t\tFriends\n"
66         "-----\n");
67     while (m != NULL) {
68         print_person(m->data);
69         m = m->next;
70     }
71     printf(
72         "-----\n");
73 }
74
75 Person* find_person(char* name, LinkedList l) {
76     // Either find the person with a particular name
77     // if not found return NULL
78     while (l != NULL) {
79         if (strcmp(l->data->name, name) == 0) {
80             return l->data;
81         }
82         l = l->next;
83     }
84     return NULL;
85 }
86
87 char* person_with_most_friends(LinkedList l) {
88     // Q1: Return the name of the person with most friends
89     // (3 marks)
90 }
91
92 LinkedList delete_by_name(char* name, LinkedList l) {
93     // Q2: Delete the person named 'name' from l (3 marks)
94 }
95
96 bool friends_triangle(LinkedList members) {
97     // Q3: Check if there is a triangle of friends
98     // ie there exists X, Y, Z such that
99     // Y is a friend of X, Z is a friend of Y, X is a friend of Z
100    // Also print all such triplets (4 marks)
101 }
102
103 int main()
104 {
105     SocialNet s = { NULL };
106
107     Person A = {"Alice", 23, Single, NULL};
108     Person B = {"Bob", 26, Engaged, NULL};
109     Person C = {"Charlie", 21, NotMentioned, NULL};
110     Person D = {"Don", 28, Married, NULL};
111
112     s.members = append(&A, s.members);
113     s.members = append(&B, s.members);
114     s.members = append(&C, s.members);
115     s.members = append(&D, s.members);
116
117     A.friends = append(&B, A.friends);
118     A.friends = append(&C, A.friends);
119     B.friends = append(&D, B.friends);
120     C.friends = append(&D, C.friends);
121     D.friends = append(&A, D.friends);
122
123     // should print Alice for the above network
124     printf("%s\n", person_with_most_friends(s.members));
125
126     // For above social network, 'friends_triangle(s.members)'
127     // returns 'true' and prints
128     // -----
129     // Friend Triangles
130     // -----
131     // Alice->Bob->Don->Alice
132     // Alice->Charlie->Don->Alice
133     // Bob->Don->Alice->Bob
134     // Charlie->Don->Alice->Charlie
135     // Don->Alice->Bob->Don
136     // Don->Alice->Charlie->Don
137     // -----
138     friends_triangle(s.members);
139
140 }
141 return 0;

```