Esercizio 13 (5 punti)

Le seguenti dichiarazioni di tipo permettono la rappresentazione dei dati necessari per l'implementazione di una piccola rete sociale che può essere descritta come un insieme di persone e di relazioni di amicizia. Il numero massimo di utenti nell'intera rete sociale è definito con la direttiva al pre-processore C: `MAX_UTENTI`.

Ogni persona è descritta da un `nome`, un `cognome`, un `nickname` e un numero intero positivo che la identifica univocamente (`id`).
Le relazioni di amicizia di ciascuna persona sono descritte tramite un *array* (`amici[...]`) contenente gli identificatori di tutti i suoi amici e un attributo `num_amici` che tiene traccia del numero di amici presenti effettivamente nell'*array.* Ogni utente può avere al più `MAX_AMICI`.

Un utente del sistema è dunque descritto da un tipo di dato (`t_utente`) che aggrega le informazioni di una persona e le sue relazioni di amicizia; mentre, la rete sociale è descritta come un tipo di dato *array* in cui ogni cella contiene le informazioni di un utente.

```
#define MAX_UTENTI 200
#define MAX_AMICI  200
#define LUN_NOME   50

typedef struct {
                char nome[LUN_NOME], cognome[LUN_NOME];
                char nickname[LUN_NOME];
                unsigned int id;                // identificatore univoco
                unsigned int num_amici;         // numero di amici
                unsigned int amici[MAX_AMICI];  // array con gli id degli amici
        } t_utente;

typedef t_utente t_rete_sociale[MAX_UTENTI];    // array di utenti
```

Si codifichi in C la funzione

```
int cerca_utente(unsigned int id, t_rete_sociale rete, unsigned int num_utenti);
```

che, dato l'identificatore univoco di un utente (`id`), una rete sociale (`rete`) e il numero di utenti effettivamente presenti nella rete sociale (`num_utenti`), restituisca il valore intero che rappresenta la posizione dell'utente nell'*array* `rete`, oppure `-1` se l'utente non è presente nell'*array*.

Soluzione

```
int cerca_utente(unsigned int id, t_rete_sociale rete, unsigned int num_utenti){

  if (num_utenti > MAX_UTENTI)  // condizione di errore:
    return -1;                  // il valore del 3zo parametro e' troppo grande!

  unsigned int i = 0;
  while (i < num_utenti && rete[i].id != id)
       i++;

  if (i == num_utenti)          // se l'utente non e' presente nell'array
    i = -1;

  return i;
}
```

Esercizio 14 (8 punti)

Dati gli identificatori di due utenti e la rete sociale a cui appartengono, codificare in C la funzione

```
void amici_comuni(unsigned int id1,
                  unsigned int id2,
                  t_rete_sociale rete , unsigned int num_utenti);
```

che stampa a video l'elenco degli amici **in comune** dei due utenti `id1, id2`.
Stampare un amico per riga e fornire nome, cognome e *nickname* per ciascun amico trovato.
Dove appropriato, è consigliato il ri-uso della funzione `cerca_utente(...)`, come dichiarata nell'esercizio precedente.

Soluzione

```
void amici_comuni(unsigned int id1,
                  unsigned int id2,                        // identificatori utenti
                  t_rete_sociale rete, unsigned int num_utenti)  // rete sociale
{
  int i = 0;                        // indice nell'array dell arete sociale
  int pos1 = pos2 = -1;             // inizializzazione indici di posizione degli
                                    // utenti con gli identificatori dati

  // ricerca posizione utenti conoscendo gli id
  while ( i < num_utenti && (pos1 < 0 || pos2 < 0)) {
    if (rete[i].id == id1) pos1 = i;
    if (rete[i].id == id2) pos2 = i;
    i++;
  }
  if (pos1 == -1) {
    printf("L'utente con id: %d, non esiste nella rete sociale!\n\n", id1);
    return;
  }
  if (pos2 == -1) {
    printf("L'utente con id: %d, non esiste nella rete sociale!\n\n", id2);
    return;
  }

  printf("Gli amici in comune dei due utenti sono:\n\n");
  int amico, flag_almeno_uno = 0;
            // ricerca degli id degli amici di utente1 fra gli amici di utente2
  for (i = 0; i < rete[pos1].num_amici; i++) {
    for (int j = 0; j < rete[pos2].num_amici; j++) {

      if (rete[pos1].amici[i] == rete[pos2].amici[j]) {
                                    // ottieni indice amico e stampa info amico
        amico = cerca_utente(rete[pos1].amici[i], rete, num_utenti);
        if (amico != -1) {
          printf("%s %s, %d anni\n", rete[amico].nome,
                                     rete[amico].cognome,
                                     rete[amico].nickname);
          flag_almeno_uno = 1;
        } // end if
      } // end if

    } // end for
  } // end for
                    // se non é stato trovato nessun amico in comune, stampa info
  if (flag_almeno_uno == 0)
    printf("\n I due utenti non hanno amici in comune.\n");
}
```

Esercizio 15 (6 punti)

Assumiamo ora che la rete sociale non sia più modellizzata con un tipo di dato "*array* di utenti", ma con un tipo di dato "lista dinamica di utenti", come segue.

```
#define MAX_UTENTI 200
#define MAX_AMICI  200
#define LUN_NOME   50

typedef struct {
                char nome[LUN_NOME], cognome[LUN_NOME];
                char nickname[LUN_NOME];
                unsigned int id;                  // identificatore univoco
                unsigned int num_amici;           // numero di amici
                unsigned int amici[MAX_AMICI];    // array con gli id degli amici
        } t_utente;

typedef struct t_struct_nodo {
                 t_utente info;
                 struct t_struct_nodo* next;
        } t_nodo;

typedef t_nodo* lista;
```

Definire un sottoprogramma che abbia un parametro di tipo `lista` e che restituisca il numero di amici dell'utente più popolare della rete sociale.

Soluzione

```
unsigned int num_amici_utente_piu_popolare(lista rete) {
     unsigned int massimo = 0;

     while (rete != NULL) {
       if ( rete->info.num_amici > massimo )
          massimo =  rete->info.num_amici;

       rete = rete->next;
     } // end while
     return massimo;
}
```

Esercizio 4 (5 punti)

Si assuma che la rete sociale sia rappresentanta con il tipo di dato "lista dinamica di utenti" specificato nell'esercizio 3.
Si scriva un sottoprogramma `... stampa_popolari(...)` che abbia un parametro di tipo `lista` e che stampi a schermo il *nickname* dell'utente più popolare (o i *nickname* degli utenti più popolari, se ve né più di uno).
Il sottoprogramma deve far uso di un ulteriore sottoprogramma **ricorsivo** `...stampa_con_max_amici(...)` che abbia come parametri la lista che rappresenta la rete sociale e il numero massimo di amici che un utente ha nella rete stessa.

Soluzione

```
void stampa_con_max_amici(lista rete, unsigned int massimo) {
    if (rete == NULL) return;
    if (rete->info.num_amici == max) {
      printf("%s\n", rete->info.nickname);
    }
    stampa_con_max_amici(rete->next, massimo);
}

void stampa_popolari(lista rete) {
    unsigned int max = num_amici_utente_piu_popolare(rete);
    stampa_con_max_amici(rete, massimo);
}
```