Esame di Fondamenti di Informatica
Ingegneria Meccanica (A–O)
Appello del 17/7/2001

Compito A

Esercizio 1

Un elaboratore adotta per i **numeri interi** una rappresentazione in complemento a due su un byte e per i **numeri reali** una rappresentazione in virgola mobile con un byte per la mantissa normalizzata in segno/modulo (si usi il primo bit della mantissa per rappresentare il segno) ed un byte per l'esponente in complemento a due.
Si consideri l'espressione:

(27 + 13)/8 - 2.46

Indicare il risultato dell'espressione ottenuto eseguendo il calcolo con l'elaboratore dato. Mostrare i passaggi intermedi eseguiti dall'elaboratore con particolare riferimento alle operazioni ed alla rappresentazione interna in binario, nonché i relativi eventuali errori compiuti. Scrivere il risultato finale in codifica decimale.

Esercizio 2

Dato il seguente programma C:

```
#include <stdio.h>

const int n=4;

 int F(int V[], int i) {
  if(i>V[n-i]) V[i]+=F(V, --i);
  return i;
}

main() {
  int V[n];
  int i;

  for(i=0; i<n; i++) V[i]=i;
  printf("%d\n", F(V, i));
  for(i=0; i<4; i++) printf("%d\t", V[i]);
  printf("\n");
}
```

Scrivere l'output prodotto dal programma ed illustrare la dinamica dei record di attivazione sullo stack. Motivare opportunamente le risposte.

Esercizio 3

Il circolo tennis "Sei un dritto" mantiene le informazioni sui propri affiliati in un file binario "ISCRITTI.DAT" all'interno del quale sono memorizzati, per ogni iscritto, il cognome, un punteggio in classifica (rappresentato da un numero intero) ed una variabile intera che vale 0 se e solo se l'affiliato non ha pagato la retta annua. Si supponga che non esistano omonimie tra gli iscritti e che ci siano un massimo di 100 iscritti al club.
Gli iscritti al circolo giocano partite tra di loro. I risultati delle partite sono memorizzati all'interno di un file di testo, chiamato "PARTITE.TXT", che contiene, per ogni linea, il cognome del giocatore che ha vinto la partita ed il cognome del giocatore che ha perso la partita.
Ogni partita vinta aumenta il numero di punti in classifica del vincitore: per la precisione, il vincitore di una partita guadagna due (2) punti se ha sconfitto un giocatore che aveva più punti in classifica di lui, ed uno (1) solo altrimenti. Al contrario, il giocatore che perde la partita ottiene la diminuzione del suo punteggio in classifica di un (1) punto solamente se è stato sconfitto da un giocatore con meno punti in classifica di lui. Ad esempio, se Rossi (con 54 punti in classifica) batte Bianchi (con 67 punti in classifica), Rossi guadagna 2 punti mentre Bianchi ne perde 1. Se, invece, è Bianchi a battere Rossi, Bianchi guadagna 1 punto mentre Rossi non ne perde alcuno.
Si scriva un programma C per la gestione del tennis club "Sei un dritto" che:

1. Legga dal file "ISCRITTI.DAT" le informazioni sugli affiliati e le memorizzi all'interno di un vettore `V`.
2. Aggiorni il vettore `V` con le informazioni contenute all'interno del file "PARTITE.TXT". A tal scopo si supponga l'esistenza di una funzione `trova_iscritto` che, dato un vettore di iscritti ed una stringa rappresentante il cognome da cercare, restituisca l'indice corrispondente al giocatore affiliato con tale cognome.
3. Stampi a video il cognome ed il punteggio in classifica del giocatore con il punteggio più alto tra quelli che hanno pagato la retta annua.
4. Aggiorni le informazioni all'interno del file "ISCRITTI.DAT".
5. Si scriva il codice della funzione `trova_iscritto` (a tal scopo si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche).

Esercizio 4

Illustrare il concetto di funzione in C.

Soluzione Esercizio 1

Rappresentazione **27**:
$(27)_{10} = (11011)_2$
rappr. interna = **0001 1011**

Rappresentazione **13**:
$(13)_{10} = (1101)_2$
rappr. interna = **0000 1101**

Rappresentazione **8**:
$(8)_{10} = (1000)_2$
rappr. interna = **0000 1000**

Rappresentazione **2.46**:
$(2.46)_{10} = (10.011101\ldots)_2$
Errore di troncamento!
Normalizzando: 0.10011101×2^2
byte esponente = **0000 0010**
byte mantissa = **0001 1101**

Risultato **27 + 13**:
0001 1011 +
0000 1101 =
$0010\ 1000 = (40)_{10}$

Risultato **40 / 8**: $101000 / 2^3 = 101 = (5)_{10}$

Risultato **5 – 2.46**:
Normalizzazione del risultato: 0.101×2^3
Incolonnamento del numero più piccolo: 0.01001110×2^3
Errore di incolonnamento!
$0.1010\ 0000 \times 2^3$ –
$0.0100\ 1110 \times 2^3$ =
$0.0101\ 0010 \times 2^3$
Normalizzando: 0.10100100×2^2
Errore di cancellazione!
byte esponente = **0000 0010**
byte mantissa = **0010 0100**
risultato: $(10.100100)_2 = (2.5625)_{10}$
risultato atteso: $(2.54)_{10}$

Soluzione Esercizio 2

Il risultato del programma è il seguente:

3
0 1 4 5

Infatti, la prima invocazione della funzione `F` modifica l'ultimo (`i`=3) valore del vettore `V`, la seconda invocazione modifica il penultimo (`i`=2) valore del vettore `V`, mentre la terza invocazione non modifica alcun valore, limitandosi a terminare la ricorsione.

Soluzione Esercizio 3

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

typedef struct {
  char cognome[30];
  int punti;
  int retta;
} iscritto;

int trova_iscritto(iscritto V[], char cognome[]);

main() {
  int N=0, i, k, punti=-1;
  iscritto V[100];
  FILE *fp=fopen("ISCRITTI.DAT", "rb");
  while(!feof(fp)) {
    fread(&(V[N]), sizeof(iscritto), 1, fp);
    N++;
  }
/* in alternativa:
  N=fread(V, sizeof(iscritto), 100, fp); */
  fclose(fp); /* fine domanda 1. */
  fp=fopen("PARTITE.TXT", "rt");
  while(!feof(fp)) {
    char c1[30], c2[30];
    int j1, j2;
    fscanf(fp, "%s%s", c1, c2);
    j1=trova_iscritto(V, c1); j2=trova_iscritto(V, c2);
    if(V[j1].punti<V[j2].punti) {
      V[j1].punti+=2; V[j2].punti-=1;
    }
    else V[j1].punti+=1;
  }
  fclose(fp); /* fine domanda 2. */
  for(i=0; i<N; i++)
    if(V[i].retta && (V[i].punti>punti)) {
      punti=V[i].punti;
      k=i;
    }
  printf("%s %d", V[k].cognome, V[k].punti) /* fine domanda 3. */
  fp=fopen("ISCRITTI.DAT", "wb");
  fwrite(V, sizeof(iscritto), N, fp);
  fclose(fp); /* fine domanda 4. */
}

int trova_iscritto(iscritto V[], char cognome[]) {
  int i=0;
  while(strcmp(V[i].cognome, cognome)) i++;
  return i;
} /* fine domanda 5. */
```

Esame di Fondamenti di Informatica
Ingegneria Meccanica (A–O)
Appello del 17/7/2001

Compito B

Esercizio 1

Un elaboratore adotta per i **numeri interi** una rappresentazione in complemento a due su un byte e per i **numeri reali** una rappresentazione in virgola mobile con un byte per la mantissa normalizzata in segno/modulo (si usi il primo bit della mantissa per rappresentare il segno) ed un byte per l'esponente in complemento a due.
Si consideri l'espressione:

(40 – 37)*4 + 4.92

Indicare il risultato dell'espressione ottenuto eseguendo il calcolo con l'elaboratore dato. Mostrare i passaggi intermedi eseguiti dall'elaboratore con particolare riferimento alle operazioni ed alla rappresentazione interna in binario, nonché i relativi eventuali errori compiuti. Scrivere il risultato finale in codifica decimale.

Esercizio 2

Dato il seguente programma C:

```
#include <stdio.h>

const int n=4;

 int F(int V[], int i) {
  if(i<V[i]) V[i]+=F(V, ++i);
  return i;
}

main() {
  int V[n];
  int i;

  for(i=n; i>0; i--) V[i-1]=n-i;
  printf("%d\n", F(V, i));
  for(i=0; i<4; i++) printf("%d\t", V[i]);
  printf("\n");
}
```

Scrivere l'output prodotto dal programma ed illustrare la dinamica dei record di attivazione sullo stack. Motivare opportunamente le risposte.

Esercizio 3

Il circolo tennis "Sei un dritto" mantiene le informazioni sui propri affiliati in un file di testo "ISCRITTI.TXT" all'interno del quale sono memorizzati, per ogni linea, il cognome, un punteggio in classifica (rappresentato da un numero intero) ed una variabile intera che vale 0 se e solo se l'affiliato non ha pagato la retta annua. Si supponga che non esistano omonimie tra gli iscritti e che ci siano un massimo di 200 iscritti al club.
Gli iscritti al circolo giocano partite tra di loro. I risultati delle partite sono memorizzati all'interno di un file binario, chiamato "PARTITE.DAT", che contiene, per ogni partita, il cognome del giocatore che ha vinto la partita ed il cognome del giocatore che ha perso la partita.
Ogni partita vinta aumenta il numero di punti in classifica del vincitore: per la precisione, il vincitore di una partita guadagna due (2) punti se ha sconfitto un giocatore che aveva più punti in classifica di lui, ed uno (1) solo altrimenti. Al contrario, il giocatore che perde la partita ottiene la diminuzione del suo punteggio in classifica di un (1) punto solamente se è stato sconfitto da un giocatore con meno punti in classifica di lui. Ad esempio, se Rossi (con 54 punti in classifica) batte Bianchi (con 67 punti in classifica), Rossi guadagna 2 punti mentre Bianchi ne perde 1. Se, invece, è Bianche a battere Rossi, Bianche guadagna 1 punto mentre Rossi non ne perde alcuno.
Si scriva un programma C per la gestione del tennis club "Sei un dritto" che:

1. Legga dal file "ISCRITTI.TXT" le informazioni sugli affiliati e le memorizzi all'interno di un vettore `V`.
2. Aggiorni il vettore `V` con le informazioni contenute all'interno del file "PARTITE.DAT". A tal scopo si supponga l'esistenza di una funzione `trova_iscritto` che, dato un vettore di iscritti ed una stringa rappresentante il cognome da cercare, restituisca l'indice corrispondente al giocatore affiliato con tale cognome.
3. Stampi a video il cognome ed il punteggio in classifica del giocatore con il punteggio più basso tra quelli che non hanno pagato la retta annua. A tal scopo si supponga che il punteggio massimo ottenibile da un giocatore sia di 1000 punti.
4. Aggiorni le informazioni all'interno del file "ISCRITTI.TXT".
5. Si scriva il codice della funzione `trova_iscritto` (a tal scopo si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche).

Esercizio 4

Illustrare il concetto di programmazione strutturata.

Soluzione Esercizio 1

Rappresentazione **40**:
$(40)_{10} = (101000)_2$
rappr. interna = **0010 1000**

Rappresentazione **– 37**:
$(37)_{10} = (100101)_2$
rappr. interna = 0010 0101
complementando:
rappr. interna = **1101 1011**

Rappresentazione **4**:
$(4)_{10} = (100)_2$
rappr. interna = **0000 0100**

Rappresentazione **4.92**:
$(4.92)_{10} = (100.11101\ldots)_2$
Errore di troncamento!
Normalizzando: 0.10011101×2^3
byte esponente = **0000 0011**
byte mantissa = **0001 1101**

Risultato **40 – 37**:
0010 1000 +
1101 1011 =
$(1)0000\ 0011 = (3)_{10}$

Risultato **3 * 4**: $11 / 2^2 = 1100 = (12)_{10}$

Risultato **12 + 4.92**:
Normalizzazione del risultato: 0.1100×2^4
Incolonnamento del numero più piccolo: 0.01001110×2^4
Errore di incolonnamento!
$0.1100\ 0000 \times 2^4$ +
$0.0100\ 1110 \times 2^4$ =
$1.0000\ 1110 \times 2^4$
Normalizzando: 0.10001111×2^5
Nessun errore di troncamento!
byte esponente = **0000 0101**
byte mantissa = **0000 0111**
risultato: $(10000.111)_2 = (16.875)_{10}$
risultato atteso: $(16.92)_{10}$

Soluzione Esercizio 2

Il risultato del programma è il seguente:

1
3 4 3 0

Infatti, la prima invocazione della funzione `F` modifica il secondo (`i`=1) valore del vettore `V`, la seconda invocazione modifica il terzo (`i`=2) valore del vettore `V`, mentre la terza invocazione non modifica alcun valore, limitandosi a terminare la ricorsione.

Soluzione Esercizio 3

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

typedef struct {
  char cognome[30];
  int punti;
  int retta;
} iscritto;
typedef struct { char c1[30]; char c2[30]; } partita;

int trova_iscritto(iscritto V[], char cognome[]);

main() {
  int N=0, i, k, punti=1000;
  iscritto V[200];
  FILE *fp=fopen("ISCRITTI.TXT", "rt");
  while(!feof(fp)) {
    fscanf(fp, "%s%d%d", V[N].cognome, &V[N].punti, &V[N].retta);
    N++;
  }
  fclose(fp); /* fine domanda 1. */
  fp=fopen("PARTITE.DAT", "rb");
  while(!feof(fp)) {
    partita P;
    int j1, j2;
    fread(&P, sizeof(partita), 1, fp);
    j1=trova_iscritto(V, P.c1); j2=trova_iscritto(V, P.c2);
    if(V[j1].punti<V[j2].punti) {
      V[j1].punti+=2; V[j2].punti-=1;
    }
    else V[j1].punti+=1;
  }
  fclose(fp); /* fine domanda 2. */
  for(i=0; i<N; i++)
    if(!V[i].retta && (V[i].punti<punti)) {
      punti=V[i].punti;
      k=i;
    }
  printf("%s %d", V[k].cognome, V[k].punti) /* fine domanda 3. */
  fp=fopen("ISCRITTI.TXT", "wt");
  for(i=0; i<N; i++)
    fprintf(fp, "%s %d %d", V[i].cognome, V[i].punti, V[i].retta);
  fclose(fp); /* fine domanda 4. */
}

int trova_iscritto(iscritto V[], char cognome[]) {
  int i=0;
  while(strcmp(V[i].cognome, cognome)) i++;
  return i;
} /* fine domanda 5. */
```