Esame di Fondamenti di Informatica
Ingegneria Meccanica (A–O)
Appello del 7/12/2001

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 (non si esegua la divisione intera):

3.7 – 79/7

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 e confrontarlo con il risultato atteso.

Esercizio 2

Sia dato il seguente programma C:

```
#include <stdio.h>
#define m 4

int f(int *V, int n, int i) {
  if(2*i%n) return f(V+1, --n, --i);
  else return V[n-i]--;
}

main() {
  int j, V[]={0, 1, 2, 3};

  printf("%d\n", f(V, m, 3));
  for(j=0; j<m; j++) printf("%d\t", V[j]);
  printf("\n");
}
```

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

Esercizio 3

L'Ufficio Magazzino della Cooperativa Babbi Natale mantiene le informazioni sui giocattoli da consegnare ai bambini all'interno di un file binario, chiamato "REGALI.DAT". Per ogni tipo di giocattolo vengono memorizzati un codice identificativo, un nome (di al più 30 caratteri), il numero di pezzi attualmente presenti in magazzino ed un costo (totale) di rifornimento. Si supponga che il numero di tipi di giocattoli contenuti nel file sia presente all'inizio del file stesso come numero intero e che non esistano giocattoli con lo stesso nome.
Ogni Santo Natale, le richieste dei bambini giungono codificate all'interno di un file di testo chiamato "LETTERE.TXT". Ogni linea di questo file contiene il nome del bambino ed il nome del giocattolo richiesto. Si supponga che, per ogni tipo di giocattolo, la quantità presente in magazzino sia sufficiente ad esaudire tutte le richieste dei bambini.
Si scriva un programma C per gestire l'eventuale riordino dei giocattoli da parte dell'Ufficio Magazzino della Cooperativa Babbi Natale che:

1. Legga dal file "REGALI.DAT" le informazioni sui giocattoli e le memorizzi all'interno di un vettore `V` opportunamente dimensionato.
2. Per ogni riga del file "LETTERE.TXT", aggiorni il vettore `V` diminuendo la disponibilità in magazzino del giocattolo richiesto. A tal scopo si supponga l'esistenza di una procedura `aggiorna` che, dato un vettore di giocattoli ed una stringa che rappresenta il nome del giocattolo da cercare, aggiorni il vettore in maniera opportuna.
3. Calcoli e stampi il costo totale per il rifornimento di tutti i giocattoli esauriti (ovvero per cui la disponibilità è nulla).
4. Lette da tastiera le informazioni su un nuovo tipo di giocattolo, provveda ad aggiornare in maniera opportuna il file "REGALI.DAT".
5. Si scriva il codice della funzione `aggiorna` (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

Si illustri il concetto di file system.

Soluzione Esercizio 1

Rappresentazione **79**:
$(79)_{10} = (1001111)_2$
rappr. interna = **0100 1111**

Rappresentazione **7**:
$(7)_{10} = (111)_2$
rappr. interna = **0000 0111**

Risultato **79 / 7**:

1001111	111
1001 –	01011.0100
111	
01011 –	
111	
01001 –	
111	
01000 –	
111	
00100	

Errore di troncamento!
Risultato = $1011.0100 = (11.25)_{10}$

Rappresentazione **3.7**:
$(3.7)_{10} = (11.101100)_2$
Errore di troncamento!
Normalizzando: 0.11101100×2^2
byte esponente = **0000 0010**
byte mantissa = **0110 1100**

Risultato **3.7 – 11.25**:
Normalizzazione del risultato: 0.10110100×2^4
Incolonnamento del numero più piccolo: 0.00111011×2^4
Nessun errore di incolonnamento!
$0.1011\ 0100 \times 2^4 –$
$\underline{0.0011\ 1011} \times 2^4 =$
$0.0111\ 1001 \times 2^4$
Normalizzando: $– 0.11110010 \times 2^3$
Errore di cancellazione!
byte esponente = **0000 0011**
byte mantissa = **1111 0010**
risultato: $(– 111.1001)_2 = (– 7.5625)_{10}$
risultato atteso: $(– 7.5857…)_{10}$

Soluzione Esercizio 2

Il risultato del programma è il seguente:

3
0 1 2 2

Infatti, alla prima invocazione della funzione `f` la condizione non è verificata (2*3 % 4 = 2) quindi viene invocata la funzione `f` con parametri `V`+1, 3, 2; alla seconda invocazione della funzione `f` la condizione non è ancora verificata (2*2 % 3 = 1) quindi viene invocata la funzione `f` con parametri `V`+2, 2, 1; alla terza invocazione della funzione `f` la condizione è verificata (2*1 % 2 = 0) quindi viene restituito il quarto valore del vettore (`V`+2+2-1), che viene anche in seguito decrementato, e la ricorsione ha termine. Il valore restituito dalla funzione è quindi il valore originale (3), mentre i valori del vettore sono quelli indicati.

Soluzione Esercizio 3

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

typedef struct {
  int codice;
  char nome[30];
  int disp;
  int costo;
} giocattolo;

void aggiorna(giocattolo V[],char s[]);

main() {
  int N, i, totale=0;
  giocattolo *V, g;
  FILE *fp=fopen("REGALI.DAT", "rb");

  fread(&N, sizeof(int), 1, fp);
  V=(giocattolo *)malloc(N*sizeof(giocattolo));
  fread(V, sizeof(giocattolo), N, fp);
  fclose(fp); /* fine domanda 1. */
  fp=fopen("LETTERE.TXT", "r");
  while(!feof(fp)) {
    char nome[30], bimbo[30];

    fscanf(fp, "%s %s", nome, bimbo);
    aggiorna(V, nome);
  }
  fclose(fp); /* fine domanda 2. */
  for(i=0; i<N; i++)
    if(V[i].disp<=0) totale+=V[i].costo;
  printf("%d\n", totale); /* fine domanda 3. */
  fp=fopen("REGALI.DAT", "ab");
  scanf("%d %s %d %d", &g.codice, g.nome, &g.disp, &g.costo);
  fwrite(&g, sizeof(giocattolo), 1, fp);
  fclose(fp); /* fine domanda 4. */
  free(V);
}

void aggiorna(giocattolo V[],char s[]) {
  int i=0;

  while(strcmp(V[i].nome, s)) i++;
  V[i].disp--;
} /* fine domanda 5. */
```

Esame di Fondamenti di Informatica
Ingegneria Meccanica (A–O)
Appello del 7/12/2001

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 (non si esegua la divisione intera):

62/11 – 29.57

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 e confrontarlo con il risultato atteso.

Esercizio 2

Sia dato il seguente programma C:

```
#include <stdio.h>
#define m 4

int f(int *V, int n, int i) {
  if(2*i%n) return f(V+1, --n, ++i);
  else return V[n-i]++;
}

main() {
  int j, V[]={3, 2, 1, 0};

  printf("%d\n", f(V, m, 1));
  for(j=0; j<m; j++) printf("%d\t", V[j]);
  printf("\n");
}
```

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

Esercizio 3

L'Ufficio Acquisti della Cooperativa Babbi Natale mantiene le informazioni sui giocattoli da consegnare ai bambini all'interno di un file di testo, chiamato "REGALI.TXT". Per ogni linea sono memorizzate le informazioni su un tipo di giocattolo, precisamente: un codice identificativo, un nome (di al più 30 caratteri), il numero di pezzi richiesti in regalo ed un costo per singolo pezzo. Si supponga che il numero di tipi di giocattoli contenuti nel file sia presente all'inizio del file stesso come numero intero.
Le richieste dei bambini giungono, ogni Santo Natale, codificate all'interno di un file binario chiamato "LETTERE.DAT". Ogni record di questo file contiene il nome del giocattolo richiesto e il nome del bambino che lo ha richiesto.
Si scriva un programma C per gestire gli ordini dei giocattoli da parte dell'Ufficio Acquisti della Cooperativa Babbi Natale che:

1. Legga dal file "REGALI.TXT" le informazioni sui giocattoli e le memorizzi all'interno di un vettore `V` opportunamente dimensionato.
2. Per ogni riga del file "LETTERE.DAT", aggiorni il vettore `V` aumentando il numero di giocattoli richiesti di quel tipo. A tal scopo si supponga l'esistenza di una procedura `aggiorna` che, dato un vettore di giocattoli ed una stringa che rappresenta il nome del giocattolo da cercare, aggiorni il vettore in maniera opportuna.
3. Calcoli e stampi il costo totale per l'acquisto di tutti i giocattoli richiesti.
4. Lette da tastiera le informazioni su una nuova richiesta di regalo, provveda ad aggiornare in maniera opportuna il file "LETTERE.DAT".
5. Si scriva il codice della funzione `aggiorna` (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

Si illustri il funzionamento del processore comandi MS-DOS.

Soluzione Esercizio 1

Rappresentazione **62**:
$(62)_{10} = (111110)_2$
rappr. interna = **0011 1110**

Rappresentazione **11**:
$(11)_{10} = (1011)_2$
rappr. interna = **0000 1011**

Risultato **62 / 11**:

```
111110  | 1011
1111 –  | ------
1011    | 101.10100
010010 –|
  1011  |
  01110 –
   1011 |
   01100 –
    1011|
    00100
```

Errore di troncamento!
Risultato = $101.10100 = (5.625)_{10}$

Rappresentazione **29.57**:
$(29.57)_{10} = (11101.100)_2$
Errore di troncamento!
Normalizzando: 0.11101100×2^5
byte esponente = **0000 0101**
byte mantissa = **0110 1100**

Risultato **5.625 – 29.57**:
Normalizzazione del risultato: 0.10110100×2^3
Incolonnamento del numero più piccolo: 0.00101101×2^5
Nessun errore di incolonnamento!
$0.1110\ 1100 \times 2^5 –$
$\underline{0.0010\ 1101} \times 2^5 =$
$0.1011\ 1111 \times 2^5$
Normalizzando: $– 0.10111111 \times 2^5$
Nessun errore di cancellazione!
byte esponente = **0000 0101**
byte mantissa = **1011 1111**
risultato: $(– 111101.1)_2 = (– 23.875)_{10}$
risultato atteso: $(– 23.933…)_{10}$

Soluzione Esercizio 2

Il risultato del programma è il seguente:

2
3 3 1 0

Infatti, alla prima invocazione della funzione `f` la condizione non è verificata (2*1 % 4 = 2) quindi viene invocata la funzione `f` con parametri `V`+1, 3, 2; alla seconda invocazione della funzione `f` la condizione non è ancora verificata (2*2 % 3 = 1) quindi viene invocata la funzione `f` con parametri `V`+1, 2, 3; alla terza invocazione della funzione `f` la condizione è verificata (2*3 % 2 = 0) quindi viene restituito il secondo valore del vettore (`V`+2+2-3), che viene anche in seguito incrementato, e la ricorsione ha termine. Il valore restituito dalla funzione è quindi il valore originale (2), mentre i valori del vettore sono quelli indicati.

Soluzione Esercizio 3

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

typedef struct {
  int codice;
  char nome[30];
  int richiesti;
  int costo;
} giocattolo;

void aggiorna(giocattolo V[],char s[]);

main() {
  int N, i, totale=0;
  giocattolo *V;
  char nome[30], bimbo[30];
  FILE *fp=fopen("REGALI.DAT", "rt");

  fscanf(fp, "%d", &N);
  V=(giocattolo *)malloc(N*sizeof(giocattolo));
  for(i=0; i<N; i++) {
    fscanf(fp, "%d %s %d", &V[i].codice, V[i].nome, &V[i].costo);
    V[i].richiesti=0;
  }
  fclose(fp); /* fine domanda 1. */
  fp=fopen("LETTERE.DAT", "rb");
  while(!feof(fp)) {
    fread(nome, sizeof(char), 30, fp);
    fread(bimbo, sizeof(char), 30, fp);
    aggiorna(V, nome);
  }
  fclose(fp); /* fine domanda 2. */
  for(i=0; i<N; i++)
    totale+=V[i].costo*V[i].richiesti;
  printf("%d\n", totale); /* fine domanda 3. */
  fp=fopen("LETTERE.DAT", "ab");
  scanf("%s %s", nome, bimbo);
  fwrite(nome, sizeof(char), 30, fp);
  fwrite(bimbo, sizeof(char), 30, fp);
  fclose(fp); /* fine domanda 4. */
  free(V);
}

void aggiorna(giocattolo V[],char s[]) {
  int i=0;

  while(strcmp(V[i].nome, s)) i++;
  V[i].richiesti++;
} /* fine domanda 5. */
```