Esame di Fondamenti di Informatica
Ingegneria Meccanica (A–O)
Appello del 19/9/2001

Compito A

Esercizio 1

Un elaboratore adotta per i **numeri interi** una rappresentazione 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:

12.46 – 137/5

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>

int n=3;
int G(int k);

int F(int n) {
  return G(--n)-n;
}

int G(int k) {
  if(k-- > 0) return F(k);
  else return k;
}

main() {
  printf("%d %d\n", F(n), n);
}
```

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

Esercizio 3

I corsi di laurea della modernissima Università "La Sorbolona" non prevedono un unico piano di studi, ma ogni studente deve, all'atto dell'iscrizione, costruire da sé il proprio piano di studi selezionando l'insieme di esami che intende sostenere tra quelli istituiti dall'Università.
Ogni corso è caratterizzato da un numero di crediti formativi e la somma dei crediti formativi di tutti i corsi inseriti all'interno di ogni piano di studio deve essere non inferiore a 100. Non esiste, invece, alcun vincolo sul numero di esami da sostenere.
Il presidente della commissione per l'approvazione dei piani di studio dell'Università "La Sorbolona", prof. Croma-Pellata, deve controllare tutti i piani di studio presentati dagli studenti e vuole automatizzare la procedura di controllo tramite un elaboratore. A tal scopo, ha già provveduto a creare un file binario, chiamato "CORSI.DAT" che contiene le informazioni su tutti i corsi offerti dall'Università. In ogni record del file "CORSI.DAT" sono contenuti il codice del corso (un numero intero), il numero di crediti formativi ed il codice di un altro corso propedeutico al primo (0 se il corso non prevede propedeuticità). Ovviamente, nei piani di studio la propedeuticità deve essere rispettata, quindi, ad esempio, se l'esame 4137 è propedeutico all'esame 372, ogni piano di studi che presenti l'esame 372 deve contenere anche l'esame 4137, prima di esso.
La segreteria studenti fa pervenire alla commissione i piani di studio degli iscritti tramite un file di testo, chiamato "PIANI.TXT". Per ogni studente viene riportato il nome, il cognome, il numero di matricola (un semplice intero), il numero di corsi contenuti nel piano e, di seguito, i codici dei corsi scelti (si supponga che non vengano mai forniti codici errati: gli studenti de "La Sorbolona" non sbagliano mai!).
Si scriva un programma C per la commissione per l'approvazione dei piani di studio dell'Università "La Sorbolona" che:

1. Per ogni studente, legga dal file "PIANI.TXT" le informazioni sui corsi scelti e provveda a controllarne la correttezza, stampando a video il nome, il cognome ed il numero di matricola degli studenti con un piano di studi non corretto. A tal scopo, i codici degli esami di ogni studente devono essere memorizzati in un vettore `V` che deve essere opportunamente dimensionato studente per studente. Si supponga l'esistenza di una funzione `trova_corso` che, dato il codice di un corso, restituisca il numero di crediti formativi e l'eventuale codice del corso propedeutico e di una funzione `controlla_prop` che, dato un vettore di corsi, il codice di un corso ed un intero rappresentante la posizione del corso all'interno del piano di studi, restituisca un valore diverso da zero se e solo se la propedeuticità è rispettata.
2. Lette da input le informazioni su un nuovo piano di studi, aggiorni il file "PIANI.TXT" in maniera opportuna.
3. Stampi in output il numero di matricola dello studente che ha presentato il piano di studi col maggior "peso" in crediti formativi.
4. Si scriva il codice della funzione `trova_corso`. Le informazioni sui corsi non devono essere memorizzate in strutture interne, ma vanno lette volta per volta dal file "CORSI.DAT".
5. Si scriva il codice della funzione `controlla_prop` che deve restituire un valore diverso da zero se e solo se il corso passato per argomento si trova all'interno del vettore in una posizione precedente a quella indicata.

Esercizio 4

Illustrare i diversi tipi di indirizzamento offerti dalle istruzioni in linguaggio macchina.

Soluzione Esercizio 1

Rappresentazione **137**:
$(137)_{10} = (10001001)_2$
rappr. interna = **1000 1001**

Rappresentazione **5**:
$(5)_{10} = (101)_2$
rappr. interna = **0000 0101**

Risultato **137 / 5**:

```
10001001 | 101
1000 –   | 011011
 101     |
 0111 –  |
  101    |
  01000 – |
   101   |
   0111 – |
    101   |
    010   |
```

Quoziente = 0001 1011 = $(27)_{10}$
Resto = 10 = $(2)_{10}$

Rappresentazione **12.46**:
$(12.46)_{10} = (1100.0111\ldots)_2$
Errore di troncamento!
Normalizzando: 0.11000111×2^4
byte esponente = **0000 0100**
byte mantissa = **0100 0111**

Risultato **12.46 – 27**:
Normalizzazione del risultato: 0.11011×2^5
Incolonnamento del numero più piccolo: 0.01100011×2^5
Errore di incolonnamento!
$0.1101\ 1000 \times 2^5 –$
$\underline{0.0110\ 0011} \times 2^5 =$
$0.0111\ 0101 \times 2^5$
Normalizzando: $– 0.11101010 \times 2^4$
Errore di cancellazione!
byte esponente = **0000 0100**
byte mantissa = **1110 1010**
risultato: $(– 1110.101)_2 = (– 14.625)_{10}$
risultato atteso: $(– 14.54)_{10}$

Soluzione Esercizio 2

Il risultato del programma è il seguente:

-3 3

Infatti, la prima invocazione della funzione F modifica il valore del parametro n (n=2), la prima invocazione della funzione G modifica il valore del parametro k (k=1) e richiama la funzione F, la quale modifica nuovamente il valore del parametro n (n=0), mentre la seconda invocazione della funzione G termina la ricorsione restituendo il valore modificato del parametro k (k=–1). La variabile esterna n non viene modificata da alcuna funzione.

Soluzione Esercizio 3

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

typedef struct { int codice, crediti, prop; } corso;
int trova_corso(int codice, int *prop);
int controlla_prop(int *V, int codice, int posiz);

main() {
  FILE *fp=fopen("PIANI.TXT", "rt");
  char nome[15], cognome[30];
  int i, m, num, codice, best, max=0;
  while(!feof(fp)) {
    int *V, crediti=0, prop;
    fscanf(fp, "%s %s %d %d", nome, cognome, &m, &num);
    V=(int *)malloc(num*sizeof(int));
    for(i=0; i<num; i++) {
      fscanf(fp, "%d", &(V[i]));
      crediti+=trova_corso(V[i], &prop);
      if(prop && !controlla_prop(V, prop, i))
        printf("%s %s %d (propedeuticità non rispettata)", nome, cognome, m);
    }
    if(crediti<100)
      printf("%s %s %d (crediti insufficienti)\n", nome, cognome, m);
    if(crediti>max) {
      best=m;
      max=crediti;
    }
    free(V);
  }
  fclose(fp); /* fine domanda 1. */
  fp=fopen("PIANI.TXT", "at");
  scanf("%s %s %d %d", nome, cognome, &m, &num);
  fprintf(fp, "%s %s %d %d\n", nome, cognome, m, num);
  for(i=0; i<num; i++) {
    scanf("%d", &codice);
    fprintf(fp, "%d\n", codice);
  }
  fclose(fp); /* fine domanda 2. */
  printf("%d\n", best); /* fine domanda 3. */
}

int trova_corso(int codice, int *prop) {
  FILE *fp=fopen("CORSI.DAT", "rb");
  corso c;
  while(1) {
    fread(&c, sizeof(corso), 1, fp);
    if(c.codice==codice) {
      *prop=c.prop;
      fclose(fp);
      return c.crediti;
    }
  }
} /* fine domanda 4. */

int controlla_prop(int *corsi, int codice, int posiz) {
  int i;
  for(i=0; i<posiz; i++) if(corsi[i]==codice) return 1;
  return 0;
} /* fine domanda 5. */
```

Esame di Fondamenti di Informatica
Ingegneria Meccanica (A–O)
Appello del 19/9/2001

Compito B

Esercizio 1

Un elaboratore adotta per i **numeri interi** una rappresentazione 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:

155/6 – 39.72

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>

int n=-3;
int G(int k);

int F(int n) {
  return G(++n)+n;
}

int G(int k) {
  if(k++ < 0) return F(k);
  else return k;
}

main() {
  printf("%d %d\n", F(n), n);
}
```

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

Esercizio 3

I corsi di laurea della modernissima Università "La Sorbolona" non prevedono un unico piano di studi, ma ogni studente deve, all'atto dell'iscrizione, costruire da sé il proprio piano di studi selezionando l'insieme di esami che intende sostenere tra quelli istituiti dall'Università.
Ogni corso è caratterizzato da un numero di crediti formativi e la somma dei crediti formativi di tutti i corsi inseriti all'interno di ogni piano di studio deve essere non inferiore a 150, ma il numero massimo di esami è 30.
Il presidente della commissione per l'approvazione dei piani di studio dell'Università "La Sorbolona", prof. Croma-Pellata, deve controllare tutti i piani di studio presentati dagli studenti e vuole automatizzare la procedura di controllo tramite un elaboratore. A tal scopo, ha già provveduto a creare un file binario, chiamato "CORSI.DAT" che contiene le informazioni su tutti i corsi offerti dall'Università. In ogni record del file "CORSI.DAT" sono contenuti il codice del corso (un numero intero), il numero di crediti formativi ed il codice di un altro corso abbinato al primo (0 se il corso non prevede abbinamenti). Ovviamente, nei piani di studio l'abbinamento deve essere rispettato, quindi, ad esempio, se l'esame 4137 è abbinato all'esame 372, ogni piano di studi che presenti l'esame 372 deve contenere anche l'esame 4137.
La segreteria studenti fa pervenire alla commissione i piani di studio degli iscritti tramite un altro file di testo, chiamato "PIANI.TXT". Per ogni studente viene riportato il nome, il cognome, il numero di matricola (un semplice intero), il numero di corsi contenuti nel piano e, di seguito, i codici dei corsi scelti (si supponga che non vengano mai forniti codici errati: gli studenti de "La Sorbolona" non sbagliano mai!).
Si scriva un programma C per la commissione per l'approvazione dei piani di studio dell'Università "La Sorbolona" che:

1. Per ogni studente, legga dal file "PIANI.TXT" le informazioni sui corsi scelti e provveda a controllarne la correttezza, stampando a video il nome, il cognome ed il numero di matricola degli studenti con un piano di studi non corretto. A tal scopo, i codici degli esami devono essere memorizzati in un vettore `V` che deve essere opportunamente dimensionato studente per studente. Si supponga l'esistenza di una funzione `trova_corso` che, dato il codice di un corso, restituisca il numero di crediti formativi e l'eventuale codice del corso abbinato e di una funzione `controlla_abb` che, dato un vettore di corsi, un intero che rappresenta il numero di elementi contenuti nel vettore ed il codice di un corso, restituisca un valore diverso da zero se e solo se il corso è contenuto nel vettore.
2. Lette da input le informazioni su un nuovo corso dell'Università, aggiorni il file "CORSI.DAT" in maniera opportuna.
3. Stampi in output il numero di matricola dello studente che ha presentato il piano di studi col minor numero di esami.
4. Si scriva il codice della funzione `trova_corso`. Le informazioni sui corsi non devono essere memorizzate in strutture interne, ma vanno lette volta per volta dal file "CORSI.DAT".
5. Si scriva il codice della funzione `controlla_abb` che deve restituire un valore diverso da zero se e solo se il corso passato per argomento si trova all'interno del vettore.

Esercizio 4

Descrivere i registri tipicamente presenti nella CPU.

Soluzione Esercizio 1

Rappresentazione **155**:
$(155)_{10} = (10011011)_2$
rappr. interna = **1001 1011**

Rappresentazione 6:
$(6)_{10} = (110)_2$
rappr. interna = **0000 0110**

Risultato **155 / 6**:

```
10011011 | 110
1001 –   | 011001
 110
 0111 –
  110
  001011 –
    110
    0101
```

Quoziente = 0001 1001 = $(25)_{10}$
Resto = 101 = $(5)_{10}$

Rappresentazione **39.72**:
$(39.72)_{10} = (100111.10\ldots)_2$
Errore di troncamento!
Normalizzando: 0.10011110×2^6
byte esponente = **0000 0110**
byte mantissa = **0001 1110**

Risultato **25 – 39.72**:
Normalizzazione del risultato: 0.11001×2^5
Incolonnamento del numero più piccolo: 0.01100100×2^6
Nessun errore di incolonnamento!
$0.1001\ 1110 \times 2^6 –$
$\underline{0.0110\ 0100} \times 2^6 =$
$0.0011\ 1010 \times 2^6$
Normalizzando: $– 0.11101000 \times 2^4$
Errore di cancellazione!
byte esponente = **0000 0100**
byte mantissa = **1110 1000**
risultato: $(– 1110.1)_2 = (– 14.5)_{10}$
risultato atteso: $(– 14.72)_{10}$

Soluzione Esercizio 2

Il risultato del programma è il seguente:

-1 -3

Infatti, la prima invocazione della funzione F modifica il valore del parametro n (n=–2), la prima invocazione della funzione G modifica il valore del parametro k (k=–1) e richiama la funzione F, la quale modifica nuovamente il valore del parametro n (n=0), mentre la seconda invocazione della funzione G termina la ricorsione restituendo il valore modificato del parametro k (k=1). La variabile esterna n non viene modificata da alcuna funzione.

Soluzione Esercizio 3

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

typedef struct { int codice, crediti, abb; } corso;
int trova_corso(int codice, int *abb);
int controlla_abb(int *V, int n, int codice);

main() {
  FILE *fp=fopen("PIANI.TXT", "rt");
  int best, min=30;
  corso c;
  while(!feof(fp)) {
    int *V, i, m, num, crediti=0, abb;
    char nome[15], cognome[30];
    fscanf(fp, "%s %s %d %d", nome, cognome, &m, &num);
    V=(int *)malloc(num*sizeof(int));
    for(i=0; i<num; i++) fscanf(fp, "%d", &(V[i]));
    if(num>30) {
      printf("%s %s %d (troppi esami)\n", nome, cognome, m);
      free(V);
      continue;
    }
    for(i=0; i<num; i++) {
      crediti+=trova_corso(V[i], &abb);
      if(abb && !controlla_abb(V, num, abb))
        printf("%s %s %d (abbinamento non rispettato)\n", nome, cognome, m);
    }
    if(crediti<150)
      printf("%s %s %d (crediti insufficienti)", nome, cognome, m);
    if(num<min) {
      best=m;
      min=num;
    }
    free(V);
  }
  fclose(fp); /* fine domanda 1. */
  scanf("%d %d %d", &c.codice, &c.crediti, &c.abb);
  fp=fopen("CORSI.DAT", "ab");
  fwrite(&c, sizeof(corso), 1, fp);
  fclose(fp); /* fine domanda 2. */
  printf("%d", best); /* fine domanda 3. */
}

int trova_corso(int codice, int *abb) {
  FILE *fp=fopen("CORSI.DAT", "rb");
  corso c;
  while(1) {
    fread(&c, sizeof(corso), 1, fp);
    if(c.codice==codice) {
      *abb=c.abb;
      fclose(fp);
      return c.crediti;
    }
  }
} /* fine domanda 4. */
int controlla_abb(int *corsi, int n, int codice) {
  int i;
  for(i=0; i<n; i++) if(corsi[i]==codice) return 1;
  return 0;
} /* fine domanda 5. */
```