B

Fondamenti di Informatica L-A Prof. Paolo Torroni

Prima prova parziale 5 Novembre 2005

Note allo svolgimento. Ti sono stati consegnati:
a — il *testo del compito*;
b — una *scheda anagrafica* con i tuoi dati: sul retro della scheda vi sono due adesivi che riportano un codice a barre;
c — una *scheda delle risposte*;
d — un *foglio di istruzioni* per la compilazione della scheda delle risposte.
Per un corretto svolgimento della prova, è necessario seguire in sequenza le seguenti istruzioni:

1. applicare *uno* dei due adesivi contenenti il codice a barre nell'apposito riquadro sulla scheda anagrafica;
2. scrivere il tipo di compito assegnato (**B**) sulla scheda anagrafica, di fianco al proprio nome;
3. compilare la scheda delle risposte, indicando sulla stessa con una croce $\times$ le risposte ritenute corrette, (vedi foglio di istruzioni). A questo proposito, è importante sapere che:
   - ogni errore determinerà un **punteggio negativo**;
   - se a un quesito non si fornisce alcuna risposta, si otterrà per quel quesito il **punteggio** 0;
4. al termine della prova, applicare il secondo codice a barre adesivo nell'apposito riquadro sulla scheda delle risposte. Consegnare i due fogli su cui è stato applicato il codice a barre (scheda anagrafica e scheda delle risposte).

È severamente proibito consultare libri, appunti, manuali, o strumenti elettronici (computer, cellulari, palmari ecc.).
La prova dura 1 ora.

1. [3.5]

Si consideri l'architettura di un Personal Computer.
A La memoria di massa e' volatile.
B La Control Unit (CU) esercita il controllo sui trasferimenti tra CPU e memoria.
C Il bus può essere utilizzato per il trasferimento di dati e istruzioni tra memoria e CPU.
D Nella fase di fetch viene eseguita l'istruzione indirizzata dal Program Counter.
E Il registro Flag (o PSW) contiene informazioni riguardanti l'ultima operazione eseguita dalla ALU.

2. [3.5]

Linguaggi di programmazione.
A L'esecuzione di un programma compilato è tipicamente più veloce dell'esecuzione dello stesso programma interpretato.
B Un programma scritto in linguaggio assembler non necessita di traduzione.
C Il debugger serve per trovare più facilmente errori sintattici nei programmi.
D Un programma che viene compilato correttamente può contenere errori.
E Un programma espresso in linguaggio macchina e' una sequenza di bit.

3. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>

main()
{ int A=2;
  float B=A--;
  char C=A;

  if (--B)
      if(--A)
          C='B';
      else
          C='A';
  else  C=A;
  /* punto 1 */
  B=(A+=3, A--); /*istruzione 1*/
  /* punto 2 */
  { float num;
    num=(C>'B'? B--: B/A);
    /* punto 3*/
  }
  /*punto 4*/
}
```

A Al punto 3 la variabile num ha valore 1.5.
B Al punto 4 la variabile num e' visibile.
C L'istruzione 1 effettua 2 assegnamenti.
D Al punto 1 la variabile A ha valore 0.
E Al punto 1 la variabile C assume il valore 'B'.

4. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>

main()
{int A; float B; char C, ch;
 A=4; B=-2; C='C'; ch='D';

 A=( A%(int)B ?++A: ch-C); /* istruzione 1 */
 B=A++/B; /* istruzione 2 */
 ch=C+=1; ch-=C;/* istruzione 3 */
}
```

A Immediatamente dopo l'esecuzione dell'*istruzione 2*, la variabile B ha il valore -0.5.
B Immediatamente dopo l'esecuzione dell'*istruzione 3*, la variabile ch ha valore '0'.
C Immediatamente dopo l'esecuzione dell'*istruzione 1*, la variabile A ha il valore corrispondente al carattere ASCII 'A'.
D Immediatamente dopo l'esecuzione dell'*istruzione 3*, le variabili C e ch hanno lo stesso valore.
E Immediatamente dopo l'esecuzione dell'*istruzione 2*, la variabile A ha il valore 2.

5. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>

main()
{int x[4]={0, 3, 4, 8};
 int j, A, B=0, C;
 for(j=3; j>=0; j--) /*istruzione 1*/
   switch(x[j]%3-1)  /*istruzione 2*/
     {case 0:     x[j]--;
      case 1:     A=x[j]; break;
      case 2:     C=0; break;
      default:    B=--A;
     }
 C=x[0]+x[2] - B; /* istruzione 3*/
}
```

A Immediatamente dopo l'istruzione 3, la variabile C ha il valore 3.
B Al termine dell'esecuzione, il valore di due elementi di V risultano modificati rispetto ai loro valori iniziali.
C Al termine dell'esecuzione, il valore di V[3] e' uguale a 7.
D Immediatamente dopo l'istruzione 3, la variabile A ha il valore 1.
E Immediatamente dopo il ciclo for, la variabile B ha il valore 1.

6. [4]

Si consideri il seguente programma C:

```
#include <stdio.h>
#define N 8

main()
{typedef float V1[N];
 typedef char V2[N];
 V1 x={0,0.5,1,1.5,2.0,2.5,3.0,3.5};
 V2 y={'C','i','a','o',0,1,2,3};
 int i;
 x[N-1]=0;
 for (i=1; i<N; i+=i)
   x[i]=y[N-i]/i;
 /* punto 1 */
 if (x[--i]?1:x[0])
   printf("Ciao!\n");
 else
   {i=x[0];
    for (;i<=4;) printf("%c", y[i++]);
   }
 /* punto 2*/
}
```

A Al punto 1: il valore di x[4] e' 1.5.
B Al punto 1: almeno un elemento di y e' stato modificato.
C Al punto 2: la variabile i ha valore N-3.
D Il programma stampa "Ciao!".
E Al punto 1: il valore di x[1] e' 3.

7. [3.5]

Si considerino i vettori nel linguaggio C:

A Nella definizione di un vettore, la dimensione puo' essere espressa mediante una espressione matematica.
B L'indice di un elemento di un vettore deve essere minore della sua dimensione fisica.
C L'indice di un elemento puo' essere una variabile di tipo char.
D L'indice di un elemento non puo' essere una variabile di tipo float.
E E' possibile copiare gli elementi di un vettore in un altro vettore.

8. [3.5]

Si considerino i tipi di dato scalari primitivi nel linguaggio C:

A L'applicazione della regola di conversione implicita produce una espressione in cui tutti gli operandi assumono uno stesso tipo.
B La conversione esplicita puo' trasformare un dato scalare in uno strutturato.
C Il criterio di equivalenza strutturale puo' consentire l'assegnamento del valore di una variabile di tipo non primitivo a una variabile di tipo scalare primitivo.
D Non e' possibile eseguire una somma tra una variabile `double` e una variabile `char`
E L'*overloading* degli operatori permette di denotare operatori diversi con lo stesso simbolo.

9. [3.5]

Si considerino le istruzioni del linguaggio C:

A Ogni istruzione `for` e' sempre esprimibile mediante `while`.
B Ogni istruzione `if` e' esprimibile mediante un unico ciclo while.
C Non e' possibile usare un `if` all'interno di uno `switch`.
D Nell'istruzione `do..while` il blocco di istruzioni puo' non essere eseguito neanche una volta.
E Nell'istruzione `do..while` se la condizione non e' verificata, si esce dal ciclo.