



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 (**D**) sulla scheda anagrafica, di fianco al proprio nome;
 3. compilare la scheda delle risposte, indicando sulla stessa con una croce × 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.

<p>1. [3.5]</p>	<p>Si consideri l'architettura di un Personal Computer.</p> <p>A La memoria centrale puo' contenere i dati del programma correntemente in esecuzione.</p> <p>B L'accesso alla cache di secondo livello e' piu' veloce rispetto all'accesso alla RAM.</p> <p>C La memoria di massa e' volatile.</p> <p>D La Control Unit (CU) esercita il controllo sui trasferimenti tra CPU e memoria.</p> <p>E Il bus può essere utilizzato per il trasferimento di dati e istruzioni tra memoria e CPU.</p>	<p>2. [3.5]</p>	<p>Linguaggi di programmazione.</p> <p>A Lo sviluppo di programmi in linguaggio assembler necessita dell'uso di un traduttore.</p> <p>B Un compilatore per il linguaggio C e' un programma che e' indipendente dall'architettura HW/SW del computer utilizzato.</p> <p>C L'esecuzione di un programma compilato è tipicamente più veloce dell'esecuzione dello stesso programma interpretato.</p> <p>D Un programma scritto in linguaggio assembler non necessita di traduzione.</p> <p>E Il debugger serve per trovare più facilmente errori sintattici nei programmi.</p>
<p>3. [4]</p>	<p>Si consideri il seguente programma C:</p> <pre>#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*/ }</pre> <p>A Al punto 2 la variabile A ha valore 2.</p> <p>B Al punto 2 le variabili A e B hanno valore diverso.</p> <p>C Al punto 3 la variabile num ha valore 1.5.</p> <p>D Al punto 4 la variabile num e' visibile.</p> <p>E L'istruzione 1 effettua 2 assegnamenti.</p>	<p>4. [4]</p>	<p>Si consideri il seguente programma C:</p> <pre>#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 */ }</pre> <p>A L'operatore di casting nell'<i>istruzione 1</i> e' influente.</p> <p>B Il programma puo' generare errori durante l'esecuzione.</p> <p>C Immediatamente dopo l'esecuzione dell'<i>istruzione 2</i>, la variabile B ha il valore -0.5.</p> <p>D Immediatamente dopo l'esecuzione dell'<i>istruzione 3</i>, la variabile ch ha valore '0'.</p> <p>E Immediatamente dopo l'esecuzione dell'<i>istruzione 1</i>, la variabile A ha il valore corrispondente al carattere ASCII 'A'.</p>

<p>5. [4]</p>	<p>Si consideri il seguente programma C:</p> <pre>#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*/ }</pre> <p>A Immediatamente dopo il ciclo for, la variabile C vale 0. B Il blocco di istruzioni corrispondente al case 1: viene eseguito 2 volte. C Immediatamente dopo l'istruzione 3, la variabile C ha il valore 3. D Al termine dell'esecuzione, il valore di due elementi di V risultano modificati rispetto ai loro valori iniziali. E Al termine dell'esecuzione, il valore di V[3] e' uguale a 7.</p>	<p>Si consideri il seguente programma C:</p> <pre>#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*/ }</pre> <p>A Il programma contiene un errore di sintassi. B Al punto 2: x[i-1] ha valore 0.0. C Al punto 1: il valore di x[4] e' 1.5. D Al punto 1: almeno un elemento di y e' stato modificato. E Al punto 2: la variabile i ha valore N-3.</p>
<p>7. [3.5]</p>	<p>Si considerino i vettori nel linguaggio C:</p> <p>A E' possibile stampare il contenuto di un vettore di 5 interi con una sola printf. B E' possibile ordinare un vettore di interi in ordine crescente. C Nella definizione di un vettore, la dimensione puo' essere espressa mediante una espressione matematica. D L'indice di un elemento di un vettore deve essere minore della sua dimensione fisica. E L'indice di un elemento puo' essere una variabile di tipo char.</p>	<p>Si considerino i tipi di dato scalari primitivi nel linguaggio C:</p> <p>A La sottrazione tra un float e un char produce un errore in compilazione. B Nel linguaggio C il vettore e' un tipo scalare primitivo. C L'applicazione della regola di conversione implicita produce una espressione in cui tutti gli operandi assumono uno stesso tipo. D La conversione esplicita puo' trasformare un dato scalare in uno strutturato. E 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.</p>
<p>9. [3.5]</p>	<p>Si considerino le istruzioni del linguaggio C:</p> <p>A Ogni istruzione switch e' sempre esprimibile mediante uno o piu' if (eventualmente annidati). B Nell'istruzione for l'espressione di modifica viene sempre eseguita alla fine di ogni iterazione. C Ogni istruzione for e' sempre esprimibile mediante while. D Ogni istruzione if e' esprimibile mediante un unico ciclo while. E Non e' possibile usare un if all'interno di uno switch.</p>	