Prova scritta Programmazione Procedurale con Lab. - 22 Giugno 2026

Nome e Cognome: _____ Matricola: _____

1. 5 punti (soluzione nel riquadro) Elencare le conversioni di tipo (... *da* ... a). Sia *long int* che *int* sono rappresentati su 32 bit. Cosa stampa alla fine il programma?

```
1 int x = 0L, i = -2.5L;
2 char a = (char)70, b = (char)70,
3 c = (char)50;
4 a = (a*b) / c;
5 unsigned int limit = 8U;
6 long n = 30L;
7 if (i < limit)
8       x = limit * n;
9 printf("%d %d\n", a, i);
```

Tutte le conversioni sono implicite, eccetto dove indicato.

Linea 1: 0L da long int a int ; -2.5L da long double a int (i diventa -2)
Linea 2-3: 70 e 50 da int a char (conversioni ESPLICITE, cast)
Linea 4: a, b, c da char a int (integer promotion) per il calcolo;
(a*b)/c e' int; il risultato (98) da int a char (assegnato ad a)
Linea 7: i da int a unsigned int (per il confronto con limit)
Linea 8: limit da unsigned int a unsigned long ; n da long a
unsigned long ; il risultato limit*n da unsigned long a int (assegnato a x)
Regole 1 e 2 delle slide

Stampa: 98 -2
x = limit*n non viene eseguito (ma x non viene stampato

2. 6 punti (soluzione nel riquadro) Scrivere cosa stampa la seguente porzione di codice. Che valore contiene la variabile *b* alla fine del programma?

```
1 int a = 0x14, i = !00, *b = &a;
2 for (int *p = &i; (a++, (*p)++) ? (++(*p), (a--)-1) :
3         ((*p)+=3, a-1); (*p)++) {
4           --a;
5     printf("%d %d\n", a, *p);
6     if (*p > 3) {
7           a = (!(!a) && a++) ? 3 : 2;
8       break; }
9     else continue; }
10 printf("%d\n", a);
```

19 3
18 6
3

Inizializzazione: a = 0x14 = 20 ; i = !00 = 1 (negazione logica di 0) ; b punta ad a ; p punta a i.
1a iterazione: condizione del for: (a++,(*p)++) vale 1 (vero) → ramo vero: ++(*p) porta *p a 3, (a--)-1 = 20
(vero); nel corpo --a porta a a 19; STAMPA '19 3'; *p>3? 3>3 no → else continue; poi (*p)++ porta *p a 4.
2a iterazione: (a++,(*p)++): a passa per 20, *p a 5, valore 4 (vero) → ++(*p) porta *p a 6, (a--)-1 = 19 (vero);
nel corpo --a porta a a 18; STAMPA '18 6'; *p>3? 6>3 si → a = (!(!a) && a++) ? 3 : 2 ; a=18 != 0 quindi
!(!a)=1 e a++ (=18, vero) → a diventa 3; break.
Stampa finale: 3. La variabile ~~b punta ad a, quindi alla fine *b = 3 (b contiene l'indirizzo di a~~).

3. 6 punti (su foglio protocollo) Definire una funzione *canc_elem* che prende come parametro un valore di tipo *int*, *pos*, e cancella l'elemento che si trova in posizione *pos* in una lista di elementi come specificati dalla seguente struttura (le posizioni partono da 1). Per esempio, se la lista è 1-4-67-9 e *pos* è 3, la nuova lista sarà 1-4-9 (viene cancellato l'elemento 67). Supporre *pFirst* come puntatore globale all'inizio della lista.

```
1 struct Node {
2   int info;
3   struct Node* pNext;
4 };
```

4. 6 punti (su foglio protocollo) Dire quali compilazioni provocano errore a causa del linker (e perché): 1) *gcc -c stampa.c*, 2) *gcc -o prog calc.c*, 3) *gcc -o stampa stampa.c*, 4) *gcc -c calc.c*, 5) *gcc calc.c stampa.c -o prog*. In caso il punto 5) ritorni un errore, descrivere come può essere corretto. Dopo aver corretto l'errore, che tipo di *linkage* hanno *totale*, *k*, e *logga*, ed in quale file sono definite? Cosa stampa il programma?

```c
/* calc.c */
int totale = 5;
void logga(int totale);

int main(void) {
  int totale = 5;
  do {
    logga(totale);
  } while (totale >= 0);
return 0;
}
```

```c
/* stampa.c */
#include <stdio.h>
extern int totale;

void logga(int a) {
  static int k = 0;
  printf("%d\n", totale - (a + k));
  k += 5;
}
```

5. 7 punti (soluzione nel riquadro) Cerchiare le affermazioni vere dato *int a[4] = {5+2*32, INT_MIN + 21, [2]=65540, 262144/2+99}; short int *p = (short*) a; char *q = (char*) a; *(q+3)=-1; *((short int*)&q[5])=257;* sapendo che i tre tipi usati occupano 4, 2, e 1 byte, e $262144 = 2^{18}$ (valori rappresentati in *little endian* e complemento a due). Scrivere la mappa di memoria e giustificare le affermazioni (vere o false).

A. $((\&a[4] - a) + p[5])\%2$ B. $(((int)(a+2) - (int)\&q[2]) + q[14])\,\%2$ C. $((q[12] >> 4) \,|\, q[4]) >= 35$

10100010 ← &p[0] (p[0]=69) ;
00000000
00000000 ← &p[1] (p[1]=-256)
11111111 ← *(q+3) = -1 (byte 3)

10101000 ← &p[2] (p[2]=277)
10000000
10000000 ← &p[3] (p[3]=-32767) ; *((short*)&q[5])=257 (byte 5-6)
00000001

00100000
00000000. ← &p[4] (p[4]=4)
10000000 ← &p[5] (p[5]=1)
00000000

11000110 ← &p[6] (p[6]=99)
00000000
01000000 ← &p[7] (p[7]=2)
00000000

A) ((&a[4] - a) + p[5]) % 2 = (4 + 1) % 2 = 5 % 2 → RISULTATO: 1 (VERA). &a[4]-a = 4 (differenza in numero di elementi int); p[5] = 1.

B) (((int)(a+2) - (int)&q[2]) + q[14]) % 2 = (6 + 2) % 2 = 8 % 2 → RISULTATO: 0 (FALSA). (int)(a+2) sta al byte 8, (int)&q[2] al byte 2 → differenza 6 byte; q[14] = 2.

C) ((q[12] >> 4) | q[4]) >= 35 : q[12]=99=01100011b, 99>>4 = 6 ; q[4]=21 ; 6 | 21 = 23 ; 23 >= 35 → RISULTATO: FALSA.

Esercizio 3

```
void canc_elem(int pos) {
  if (pos < 1 || pFirst == NULL)
     return;
  if (pos == 1) {
    struct Node *tmp = pFirst;
    pFirst = pFirst->pNext;
    free(tmp);
    return;
  }
  struct Node *prev = pFirst;

  /* prev avanza fino al nodo in posizione pos-1 */
  for (int k = 1; k < pos-1 && prev != NULL; k++)
      prev = prev->pNext;
  if (prev == NULL || prev->pNext == NULL)
      return; /* pos non valida */
  struct Node *target = prev->pNext;
  prev->pNext = target->pNext; /* ricollega saltando target */
  free(target);
}
```

Esercizio 4

1) gcc -c stampa.c → nessun errore di linker: -c compila soltanto (produce stampa.o), non viene invocato il linker.
2) gcc -o prog calc.c → ERRORE di linker: logga non è definita (undefined reference to logga).
3) gcc -o stampa stampa.c → ERRORE di linker: main e totale non sono definiti (manca main; totale è solo dichiarata extern).
4) gcc -c calc.c → nessun errore di linker: -c compila soltanto.
5) gcc calc.c stampa.c -o prog → nessun errore: entrambi i file sono presenti, i simboli si risolvono. (Non serve correzione.)

Linkage: totale ha linkage ESTERNO (in entrambi i file: definita in calc.c, dichiarata extern in stampa.c). logga ha linkage ESTERNO (funzione non static: dichiarata in calc.c, definita in stampa.c). k ha NO linkage (static locale dentro logga: durata statica ma nessun linkage).
Definizioni: totale definita in calc.c ; logga definita in stampa.c ; k definita in stampa.c.
Output: totale globale = 5, a = 5 (il totale locale passato), k = 0,5,10,15,... stampa: totale-(a+k) = 5-(5+0)=0, 5-(5+5)=-5, 5-(5+10)=-10, ...
0
-5
-10
-15
-20
... (CICLO INFINITO)
Il ciclo è INFINITO: in main la variabile totale (locale) non viene mai modificata, quindi la condizione while (totale >= 0) con totale = 5 resta sempre vera.