Prova scritta Programmazione Procedurale con Lab. - 8 Luglio 2026 — CORREZIONE

Nome e Cognome: _____ Matricola: _____

1. 6 punti (soluzione nel riquadro) Cosa stampa il programma? Gli operatori unari hanno precedenza massima, *a* si trova all'indirizzo di memoria *0x7fff54824ff8*, uno *short* occupa 2 byte, un *int* 4 byte, un *long* 8 byte. Supporre anche che, se l'operando sinistro di un operatore *&&* è falso (oppure quello di un operatore || è vero), l'operando destro NON viene valutato.

```
1 int a = 5, *b = &a;
2 int c = !(a-=2, ((a-=3) && ++a));
3 int d = c || (a-=1, ((a-=4) || a++));
4 int e = (d -= 1, (a || d) ||
5           (c = a - 2, ++a));
6 printf("%d %d %d %d\n", a, c, d, e);
7 printf("%p %p %lu\n", b, (long*)(
8     short*)b + 2, sizeof(*b));
```

1 -2 0 1
0x7fff54824ff8 0x7fff54825008 4

Riga 2: a-=2 → a=3; (a-=3) → a=0, vale 0 (falso) → && corto-circuita e ++a NON viene valutato; l'operatore virgola ritorna 0 → c = !0 = 1.
Riga 3: c=1 (vero) → || corto-circuita: tutta l'espressione a destra NON viene valutata (a resta 0); d = 1.
Riga 4-5: d-=1 → d=0; (a || d) = (0 || 0) = 0 (falso) → viene valutato l'operando destro: c = a-2 = -2, poi ++a → a=1 (vero) → e = 1.
Stampa: 1 -2 0 1.
Riga 7-8: b = 0x7fff54824ff8; il cast (short*) non sposta nulla, (long*)b + 2 avanza di 2 long = 16 byte → 0x7fff54825008; sizeof(*b) = sizeof(int) = 4.

2. 6 punti (su foglio protocollo) Scrivere una funzione che crea un array di n elementi (con n intero senza segno passato come parametro) e lo inizializza con la successione dei numeri di Fibonacci. Per esempio, se $n = 6$ l'array deve contenere 1-1-2-3-5-8. L'array creato deve essere poi ritornato dalla funzione come risultato. **(Soluzione: Esercizio 2, ultima pagina)**

3. 6 punti (su foglio protocollo) Data la seguente *struct Node* definire una funzione di nome *inserisci_ordinato* che prende come parametro un valore di tipo *int*, *x*, e inserisce un nuovo nodo contenente x mantenendo la lista ordinata in modo crescente. Per esempio, se la lista è 2-5-9 e x è 7, la nuova lista sarà 2-5-7-9. Considerare tutti i casi possibili (la lista può anche essere vuota). Supporre di avere un puntatore ad inizio lista globale di nome *pFirst*. **(Soluzione: Esercizio 3, ultima pagina)**

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

4. 5 punti (su foglio protocollo) Dire quali compilazioni provocano errore a causa del linker (e perché): 1) *gcc -c out.c*, 2) *gcc -o main main.c*, 3) *gcc -o out out.c*, 4) *gcc -c main.c*, 5) *gcc main.c out.c -o prog*. In caso il punto 5) ritorni un errore, descrivere come può essere corretto. Infine, che tipo di *linkage* hanno *val* (in entrambi i file), *j*, e *stampa*, ed in quale file sono definite? Cosa stampa il programma? **(Soluzione: Esercizio 4, ultima pagina)**

```c
/* main.c */
int val;
int val = 3;
void stampa(int val);

int main(void) {
  extern int val;
  while (val >= 0) {
    stampa(val);
  }
  return 0;
}
```

```c
/* out.c */
#include <stdio.h>
int j = 2;
static int val = 6;

void stampa(int a) {
  a++;
  printf("%d\n", val = val - j);
}
```

5. 7 punti (soluzione nel riquadro) Cerchiare le affermazioni vere dato *int a[4] = {3+2*64, INT_MIN + 9, [2]=131076, 524288/4+33}; short int *p = (short*) a; char *q = (char*) a; *(q+2)=-1; *((short int*)&q[9])=513;* sapendo che i tre tipi usati occupano 4, 2, e 1 byte, e $524288 = 2^{19}$ (valori rappresentati in *little endian* e complemento a due). Scrivere la mappa di memoria e giustificare le affermazioni (vere o false).

A. $((\&a[3] - a) + p[5])\%2$ **[VERA]** B. $(((int)(a+3) - (int)\&q[6]) + q[10])\,\%4$ **[FALSA]** C. $((q[12] >> 2) \,|\, q[4]) >= 9$ **[VERA]**

Mappa di memoria (un byte per riga, *little endian*):

`10000011` ← &q[0], &p[0] (p[0]=131) ; a[0] = 3+2*64 = 131 = 0x83
`00000000`
`11111111` ← &p[1] ; *(q+2) = -1 sovrascrive il byte 2 → p[1] = 0x00FF = 255
`00000000`
`00001001` ← &p[2] (p[2]=9) ; a[1] = INT_MIN + 9 = 0x80000009
`00000000`
`00000000` ← &p[3] (p[3]=-32768)
`10000000`
`00000100` ← &p[4] (p[4]=260) ; a[2] = 131076 = 2^{17}+4 = 0x00020004
`00000001` ← *((short*)&q[9]) = 513 = 0x0201 sovrascrive i byte 9-10 (q[9]=1, q[10]=2)
`00000010` ← &p[5] (p[5]=2)
`00000000`
`00100001` ← &p[6] (p[6]=33) ; a[3] = 524288/4+33 = 131072+33 = 131105 = 0x00020021
`00000000`
`00000010` ← &p[7] (p[7]=2)
`00000000`

A) ((&a[3] - a) + p[5]) % 2 = (3 + 2) % 2 = 5 % 2 → RISULTATO: 1 (**VERA**). &a[3]-a = 3 (differenza in numero di elementi int); p[5] = 2.

B) (((int)(a+3) - (int)&q[6]) + q[10]) % 4 = (6 + 2) % 4 = 8 % 4 → RISULTATO: 0 (**FALSA**). (int)(a+3) sta al byte 12, (int)&q[6] al byte 6 → differenza 6 byte; q[10] = 2.

C) ((q[12] >> 2) | q[4]) >= 9 : q[12] = 33 = 00100001b, 33>>2 = 8 ; q[4] = 9 ; 8 | 9 = 9 ; 9 >= 9 → RISULTATO: **VERA**.

Esercizio 2

```c
#include <stdlib.h>

int *crea_fib(unsigned int n) {
    int *v = malloc(n * sizeof(int));    /* crea l'array di n elementi */
    if (v == NULL)
        return NULL;
    for (unsigned int i = 0; i < n; i++) {
        if (i < 2)
            v[i] = 1;                    /* primi due numeri: 1, 1 */
        else
            v[i] = v[i-1] + v[i-2];      /* somma dei due precedenti */
    }
    return v;                            /* l'array viene ritornato */
}
```

Esercizio 3

```c
void inserisci_ordinato(int x) {
    struct Node *nuovo = malloc(sizeof(struct Node));
    if (nuovo == NULL)                    /* malloc fallita */
        return;
    nuovo->info = x;
    if (pFirst == NULL || x <= pFirst->info) {
        nuovo->pNext = pFirst;            /* lista vuota o inserimento in testa */
        pFirst = nuovo;
        return;
    }
    struct Node *prev = pFirst;
    while (prev->pNext != NULL && prev->pNext->info < x)
        prev = prev->pNext;               /* prev: ultimo nodo con info < x */
    nuovo->pNext = prev->pNext;           /* inserimento in mezzo o in coda */
    prev->pNext = nuovo;
}
```

Esercizio 4

1) *gcc -c out.c* → nessun errore di linker: -c compila soltanto (produce out.o), il linker non viene invocato.
2) *gcc -o main main.c* → ERRORE di linker: *stampa* non è definita (*undefined reference to stampa*).
3) *gcc -o out out.c* → ERRORE di linker: *main* non è definito.
4) *gcc -c main.c* → nessun errore di linker: -c compila soltanto.
5) *gcc main.c out.c -o prog* → nessun errore: la *val* di out.c è *static* (linkage interno), quindi non collide con la *val* di main.c; tutti i simboli si risolvono. (Non serve correzione.)

Linkage: *val* in main.c ha linkage ESTERNO (*int val;* è definizione tentativa, *int val = 3* la definizione), definita in main.c. *val* in out.c ha linkage INTERNO (*static*), definita in out.c: sono due oggetti DISTINTI. *j* ha linkage ESTERNO, definita in out.c. *stampa* ha linkage ESTERNO (dichiarata in main.c, definita in out.c). Il parametro *a* non ha linkage.

Output: in main la *val* globale di main.c vale 3 e non viene mai modificata (stampa lavora sulla propria *val* static e sulla copia *a*), quindi *while (val >= 0)* resta sempre vera: CICLO INFINITO. Ad ogni chiamata stampa esegue *val = val - j* sulla sua *val* static: 6-2=4, poi 2, 0, -2, ...

4
2
0
-2
-4
... (CICLO INFINITO)