Prova scritta Programmazione Procedurale con Lab. - 3 Giugno 2026

Nome e Cognome: _____ Matricola: _____

1. 6 punti Elencare le conversioni di tipo (... *da* ... a). Dato *USHRT_MAX = 65535*, e se possibile scrivere il valore finale esatto della variabile *b* sapendo che il carattere *a* ha valore 97 (in ASCII) e le altre lettere seguono in ordine alfabetico. Spiegare perché possibile scrivere il valore, oppure non possibile (causa precisione).

```
1  unsigned int h2(long int p) {
2    return p + 'g' - 'a';
3    }
4
5  int h1(unsigned int p) {
6    float c = 'n';
7    return h2((p + c) - 'b');
8  }
9
10 int main(void) {
11   unsigned short x = -7L;
12   double b = h1(x);
13   printf("%f\n", b);
14 }
```

Linea 11: -7L da long int a unsigned short
Linea 12: x da unsigned short a unsigned int (parametro di h1)
Linea 6: 'n' da int a float
Linea 7: p da unsigned int a float (per la somma p + c)
Linea 7: 'b' da int a float (per la sottrazione (p + c) - 'b')
Linea 7: (p + c) - 'b' da float a long int
Linea 2: 'g' e 'a' (costanti di tipo int) convertite a long int ; (p + 'g' - 'a') da long int a unsigned int (ritorno di h2)
Linea 7: valore di ritorno di h2 da unsigned int a int (ritorno di h1)
Linea 12: valore di ritorno di h1 da int a double

b vale 65547, che ha 5 cifre significative ed è quindi rappresentabile esattamente con un double che può rappresentare 15 cifre.

2. 6 punti Scrivere cosa stampa il seguente programma e motivare su foglio protocollo.

```
1  int a = 0x1E & 073;
2  printf("%d\n", a);
3  while (a % 2 == 0 ? (a >>= 1, a) : (a += 3, 1)) {
4    printf("%d\n", a);
5    if (a == 6 || a-- == 10) {
6        a *= 2;
7      break;
8    }
9    a -= 1;
10 }
11 printf("a: %d\n", a);
```

26
13
14
6
a: 12

3. 6 punti Data la seguente *struct Node* definire su foglio protocollo una funzione di nome *sposta_dispari_in_testa()* (senza parametri) che riordina la lista spostando in testa tutti i nodi con campo *info* dispari, preservando l'ordine relativo sia dei nodi dispari sia dei nodi pari (i pari restano in coda, nel loro ordine originale). La funzione non deve allocare né deallocare nodi: deve solo modificare i collegamenti. Es.: se la lista è 4-7-2-9-6-3 la nuova lista sarà 7-9-3-4-2-6. Supporre un puntatore ad inizio lista globale *pFirst*.

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

4. 5 punti

Su foglio protocollo, scrivere un programma (minimale) compilabile senza errori, avente un oggetto memorizzato in ciascuna zona di memoria vista a lezione. Descrivere in quale zona di memoria è memorizzato ciascun oggetto del programma in questione.

5. 7 punti Cerchiare le affermazioni vere dato *long long a[3] = {2048, -3, LLONG_MIN + 768};*
*short int *p = (short*) a; char *q = (char*) a; p[1]=4098, p[3]=4095-2, *(q+15)=73, p[9]=4096*4+1;*
sapendo che i tre tipi usati occupano 8, 2, e 1 byte, e $4096 = 2^{12}$ (valori rappresentati in *little endian* e complemento a due). Scrivere la mappa di memoria su foglio protocollo e giustificare brevemente le affermazioni (vere o false).

A. $(*(p+7) - p[8])\%2$ B. $(((int)(p+11) - (int)(a+2)) + q[18])\,\%7$ C. $((\&p[9] - \&p[2]) + p[8])\,\%2$

00000000 ← &p[0] (p[0] = 2048) ; qui anche a+0
00010000
01000000 ← &p[1] (p[1] = 4098)
00001000
00000000 ← &p[2] (p[2] = 0)
00000000
10111111 ← &p[3] (p[3] = 4093)
11110000
10111111 ← &p[4] (p[4] = -3) ; qui anche a+1
11111111
11111111 ← &p[5] (p[5] = -1)
11111111
11111111 ← &p[6] (p[6] = -1)
11111111
11111111 ← &p[7] (p[7] = 18943)
10010010 ← qui *(q+15) = 73
00000000 ← &p[8] (p[8] = 768) ; qui anche a+2
11000000
10000000 ← &p[9] (p[9] = 16385) ; qui anche q[18]
00000010
00000000 ← &p[10] (p[10] = 0)
00000000
00000000 ← &p[11] (p[11] = -32768) ; qui anche p+11
00000001

A) (*(p+7) - p[8]) % 2 = (18943 - 768) % 2 = 18175 % 2 → RISULTATO: 1 (VERA)
B) ((int)(p+11) - (int)(a+2)) + q[18] % 7 = 6 + 1 = 7 ; 7 % 7 → RISULTATO: 0 (FALSA)
C) (&p[9] - &p[2]) + p[8] % 2 = 7 + 768 = 775 ; 775 % 2 → RISULTATO: 1 (VERA)
Note: &p[9]-&p[2] = 7 (differenza in numero di elementi short). (int)(p+11) sta al byte 22, (int)(a+2) al byte 16
→ 22 - 16 = 6 (differenza in byte).

Esercizio 3

```
void sposta_dispari_in_testa(void) {
  struct Node *dispH = NULL, *dispT = NULL; /* sottolista dispari */
  struct Node *pariH = NULL, *pariT = NULL; /* sottolista pari */
  struct Node *curr = pFirst;
  while (curr != NULL) {
    struct Node *next = curr->pNext; /* salva il prossimo */
    curr->pNext = NULL; /* stacca il nodo */
    if (curr->info % 2 != 0) /* DISPARI: in coda ai dispari */
       {
       if (dispH == NULL) dispH = dispT = curr;
       else { dispT->pNext = curr; dispT = curr; }
       }
     else {
       /* PARI: in coda ai pari */
       if (pariH == NULL) pariH = pariT = curr;
       else { pariT->pNext = curr; pariT = curr; }
      }
     curr = next;
   }
   /* concatena: prima i dispari, poi i pari */
   if (dispH == NULL)
     pFirst = pariH; /* nessun dispari */
   else
   {
   pFirst = dispH;
   dispT->pNext = pariH; /* attacca i pari in coda */
   }
}
```

Esercizio 5

```
#include <stdlib.h>

int g = 5; /* memoria permanente */

int main(void) {
int a = 10; /* stack */
static int s = 7; /* memoria permanente */
int *p = malloc(sizeof(int)); /* p: stack ; *p: heap */
*p = 20;
free(p);
return a + g + s;
}
```

a (l6): variabile locale di main → stack (memoria delle funzioni). Creata all'ingresso di main e distrutta all'uscita.
s (l7): variabile locale ma dichiarata static → memoria permanente. Pur essendo locale a main, persiste per tutta la durata del programma (durata statica).
p (l8): il puntatore p è una variabile locale → stack.