Prova scritta Programmazione Procedurale con Lab. - 13 Febbraio 2026

Nome e Cognome: _____ Matricola: _____

Le soluzioni che verranno valutate sono solamente quelle riportate in questo foglio. Utilizzare (e consegnare) i fogli protocollo utilizzati per i calcoli.

1. [6 punti] Descrivere la regola di conversione applicata alla linea 4.

```
1 int x = 0;
2 unsigned int limit = 200U;
3 long n = 30L;
4 x = limit * n;
```

Regola 2 slide conversioni di tipo

Se x ha tipo con segno signed TipoT (quindi NON unsigned) il cui grado di conversione è più elevato di quello dell'altro operando (y), si applica questa regola. L'altro operando (y) è convertito a signed TipoT solo se questo tipo è in grado di rappresentare tutti i valori di y. Altrimenti, tutti e due gli operandi (x e y) sono convertiti a unsigned TipoT.

2. [6 punti] Data la seguente *struct Node*, si definisca una funzione che prende in input due puntatori a liste di elementi. Scrivere una funzione alternate che restituisca una nuova lista i cui elementi siano presi dalle due precedenti e disposti in maniera alternata, seguendo la seguente sequenza: *elem1lista1*, *elem1lista2*, *elem2lista1*, *elem2lista2*, ..., *elemNlista1*, *elemNlista2*. Gestire i possibili errori.

```
1     struct Node {
2       int info;
3       struct Node* pNext;
4     };
5
```

```
struct Node* alternate(struct Node* l1, struct Node* l2) {

  struct Node* head = NULL;
  struct Node* tail = NULL;

  while (l1 != NULL && l2 != NULL) {

    // nodo da l1
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) return NULL;

    newNode->info = l1->info;
    newNode->pNext = NULL;

    if (head == NULL)
      head = tail = newNode;
    else {
      tail->pNext = newNode;
      tail = newNode;
    }

    l1 = l1->pNext;

    // nodo da l2
    newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) return NULL;

    newNode->info = l2->info;
    newNode->pNext = NULL;

    tail->pNext = newNode;
    tail = newNode;

    l2 = l2->pNext;
  }

  return head;
}
```

3. [5 punti] Definire una funzione *rotate90()* che prenda in input una matrice di dimensione $m \times n$, e ne crei una $n \times m$ che corrisponde ad una rotazione in senso antiorario di 90 gradi, come in esempio. La funzione stampa anche la matrice risultato, ma non la ritorna al chiamante.


```
void rotate90(int m, int n, int A[m][n]) {

  int B[n][m];   // matrice ruotata

  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
      B[n - 1 - j][i] = A[i][j];
    }
  }

  // stampa matrice ruotata
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
      printf("%d ", B[i][j]);
    }
    printf("\n");
  }
}
```

4. [6 punti] Qual è l'output del seguente programma?

```
1  int a= 0xa;
2  while(a > 8 ? (a--, (a>7?a--:a)): a--,a--) {
3    if (a + 2 >= 07) {
4      printf("HERE %d\n", a); a != 1;
5      continue; break;
6      printf("NO MORE OK\n");
7    }
8    printf("EXIT\n");
9  }
10 a= a++ && a++; a+= 0xae;
11 printf("%d\n", a);
```

```
HERE 7
HERE 5
EXIT
EXIT
174
```

5. [7 punti] Cerchiare le affermazioni vere dato *int a[6]={1707,-761,37,INT_MAX,(INT_MAX+INT_MIN)+1 ,-1}; long long *p = (long long *) a; short *q= (short*) a; p[2]+= 255, q[7]+= 1, q[5]= ~q[5];* sapendo che i tre tipi usati occupano 8, 4, e 2 byte, con valori rappresentati in *little endian* e complemento a due. Scrivere la mappa di memoria e giustificare le affermazioni (vere o false).

(A.) $(q[8] > q[11])$ B. $(q[5] \mid (short^*)(\&a[4]))$ C. $((\sim q[11] + SHRT\_MIN) > q[12])$

```
11010101
01100000
00000000
00000000

11100000
10111111
11111111
11111111

10100100
00000000
11111111    q[5]
11111111

11111111
11111111
00000000
00000001

11111111
00000000    q[8]
00000000
00000000

11111111
11111111
11111111    q[11]
11111111
```

A. Vera
255 > -1

B. accettata in due versioni
1) L'operatore bitwise | è definito solo per tipi interi. Quindi errore di compilazione
2) dato che q[5] vale -1 (quindi tutti 1), il risultato sarà sempre diverso da 0 (quindi vero) qualsiasi sia il valore dell'altro operando

C. Falsa
La negazione di q[11] vale 0
e SHRT_MIN vale -2^16
Qualunque sia il valore di q[12], oltre l'array, il risultato sarà sempre falso