Prova scritta Programmazione Procedurale con Lab. - 10 Febbraio Gennaio 2025

Nome e Cognome: _____ Matricola: _____

1. 5 punti Su foglio protocollo, scrivere la definizione di una funzione *inverti_righe12* che prende come parametri una matrice di interi *m* e scambia gli elementi della prima riga con quelli dell'ultima riga (controllare prima che la matrice abbia per lo meno due righe).

2. 6 punti Scrivere cosa stampa la seguente porzione di codice. Che valore contiene la variabile b alla fine del programma?

```
int a= 0x3c, i= -05, *b= &a;
for (int* p= &i; (a++, (*p)++) ? (++(*p), (a--)-1) : ((*p
    )+=3, a-1); (*p)++) {
  --a;
  printf("%d %d \n", a, *p);
  if (*p > 3) {
    a= (!(-a) && a++) ? 3 : 2;
    break; }
  else continue; }
printf("%d\n", a);
```

```
59 -3
58 0
57 3
56 6
2
```

3. 6 punti Definire una funzione *ins_elem* che prende come parametri due valori di tipo *int*, *value* e *pos*, ed inserisce un nuovo elemento in posizione *pos* in una lista di elementi come specificati dalla seguente strutture; *value* è il campo *info* del nuovo elemento. Per esempio, se la lista è 1-4-67, *value* è 11 e *pos* è 2, la nuova lista sarà 1-11-4-67. Supporre *pFirst* come puntatore globale all'inizio della lista.

```
struct Node {
    int info;
    struct Node* pNext;
}
```

4. 6 punti Su foglio protocollo, scrivere (esercizio1) un tentativo di definizione che rimane dichiarazione, (esercizio2) un tentativo di definizione che diventa definizione, (esercizio3) una dichiarazione di variabile.

5. 7 punti Cerchiare le affermazioni vere dato *int a[4]= {7+4*64, INT_MIN + 39, [2]= 131002, 131072/2+111}; 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 $131072 = 2^{17}$ (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. $((q[12] >> 4) \mid q[4]) >= 35$

Esercizio 3

```
void ins_elem(int value, int pos) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Errore di allocazione di memoria\n");
        return;
    }
    newNode->info = value;
    newNode->pNext = NULL;

    // Caso in cui si inserisce all'inizio della lista
    if (pos == 0 || pFirst == NULL) {
        newNode->pNext = pFirst;
        pFirst = newNode;
        return;
    }

    struct Node* current = pFirst;
    int count = 0;

    // Trova il nodo precedente alla posizione desiderata
    while (current->pNext != NULL && count < pos - 1) {
        current = current->pNext;
        count++;
    }

    // Se la posizione è oltre la lunghezza della lista, restituisce errore
    if (count < pos - 1) {
        printf("Errore: la posizione %d è fuori dalla lunghezza della lista.\n", pos);
        free(newNode); // Dealloca la memoria per evitare perdite
        return;
    }

    // Inserisce il nuovo nodo nella posizione specificata
    newNode->pNext = current->pNext;
    current->pNext = newNode;
}
```

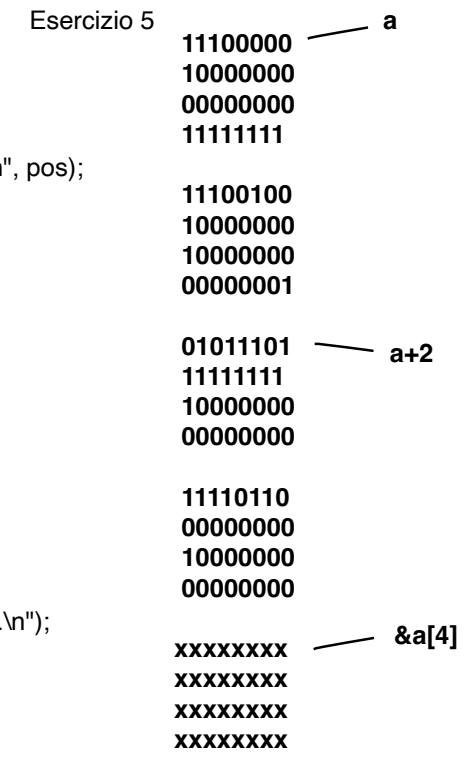
Esercizio 1

```
void inverti_righe(int righe, int colonne, int m[righe][colonne]) {
    // Controlla se la matrice ha almeno due righe
    if (righe < 2) {
        printf("Errore: la matrice deve avere almeno due righe per invertire.\n");
        return;
    }

    // Scambio della prima e dell'ultima riga
    for (int j = 0; j < colonne; j++) {
        int temp = matrice[0][j];
        matrice[0][j] = matrice[righe - 1][j];
        matrice[righe - 1][j] = temp;
    }
}
```

Esercizio 4

```
esercizio 1
int a; // tentativo che rimane dichiarazione
int a= 5;

int main() {
    printf(“”CIAO);
}

esercizio 2
int a; // tentativo che diventa definizione

int main() {
    printf(“”CIAO);
}

esercizio 3
extern int a; // dichiarazione variabile

int main() {
    printf(“”CIAO);
}
```

Esercizio 5



A: &a[4] - a == 4 (ci sono 4 interi tra i due indirizzi) + -1 (-q[5]). Il risultato è 3%2 che equivale a 1, quindi VERA

B: Tra a+2 e &q[2] la differenza è 6 (byte), q[q4] vale 1, la loro somma è 7 che modulo 2 fa 1, quindi VERA

C: q[12] nel processore 01101111
q[12] >> 4 == 00000110 in or bit a bit con
q[4] == 00100111

00000110 |
00100111 ==
00100111 che in big endian (siamo nei registri del processore) vale 39. Quindi VERA