Prova scritta Programmazione Procedurale con Lab. - 14 Aprile 2025

Nome e Cognome: _____ Matricola: _____

Le prime quattro domande sono necessarie per la valutazione della seconda parte dell'esame

1. 0 punti Marcare le affermazioni vere.
   - ◯ Le funzioni *malloc()* e *calloc()* restituiscono la quantità di byte allocati.
   - ◯ L'indirizzo di memoria restituito da *realloc()* è sempre diverso da quello passato come parametro.
   - ◯ *malloc()* alloca la memoria ed inizializza tutti i byte a 0.
   - ☒ *calloc(5,3)* e *malloc(15)* allocano la stessa quantità di byte in memoria.

2. 0 punti Elencare le parti che compongono *gcc* (solo i loro nomi).

   preprocessor, compiler, assembler, linker

3. 0 punti Quali parametri prende la funzione *realloc()*?

   La funzione realloc prende due parametri:
   void *realloc(void *ptr, size_t new_size);
   ptr: è un puntatore a un blocco di memoria precedentemente allocato (con malloc, calloc, o realloc)
   new_size: è la nuova dimensione (in byte) che si vuole per il blocco di memoria.

4. 0 punti i) Definire una variabile p come puntatore costante ad una variabile di tipo *int* a, e ii) definire una variabile q come puntatore *a* costante *int* b.

   i) int a = 10;
   int * const p = &a; // p è un puntatore costante a int
   ii) const int b = 30;
   const int *q = &b; // q è un puntatore a costante int

Prova scritta Programmazione Procedurale con Lab. - 14 Aprile 2025

Nome e Cognome: _____ Matricola: _____

1. 6 punti Su foglio protocollo, scrivere una funzione di nome *sum_col()* che prende una matrice m di r righe e c colonne (valori di tipo *int*) come input e resistuisce un array a di lunghezza pari a c, in cui ogni elemento (sempre tipo *int*) contiene la somma dei valori di ciascuna colonna di m ($a[0] =$ somma prima colonna, etc).

2. 8 punti Scrivere cosa stampa il seguente programma. Riportare obbligatoriamente la risposta nel riquadro.

```
int i, j, s, r=6, st=0;
for(i = 1; i <= r; i++) {
    if(i <= 4){
      for(s = 1; s <= 5-i; s++)
        printf("-");
      while(st != (2*i - 1)) {
        printf("*");
        st++;;
      }
      st=0;
      printf("\n");
    } else {
      for(j = 0; j < 9; j++) {
        if((int)(j/3) == 1)
          printf("o");
        else
          printf("*");
      }
      printf("\n");
    }
  }
```

```
----*
---***
--*****
-*******
***ooo***
***ooo***
```

3. 8 punti Data la seguente *struct Node*, su foglio protocollo definire una funzione *ins()* che ha come parametri una lista di elementi di tale struttura, un intero *val* e un intero *pos*. La funzione inserisce il valore *val* in posizione di lista *pos*. Il primo elemento della lista è in posizione 1.

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

4. 8 punti Dato *int a[5]= {25, 81, [2]= INT_MAX, 131046,131328};*
*short int *p = (short*) a; char *q= (char*) a; q[16]= -1;* rappresentare su foglio protocollo l'array in memoria 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). Cerchiare le affermazioni vere e giustificarle. A. $(((q[17] \mid q[0]) \,\&\, q[4]) \,\%\, 5) \,\%\, 2$
B. $((\&a[5] - a) + q[5]) \,\%\, 2$ C. $((int)(a+5) - (int)(p+5) + q[5]) \,\%\, 2$ D. $(p[4] \mid {}^*(p+5)) \,\&\, p[8] \,\%\, 256$

Esercizio 3

```
struct Node* ins(struct Node* head, int val, int pos) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->info = val;
    newNode->pNext = NULL;

    // Inserimento in testa (posizione 1)
    if (pos == 1) {
        newNode->pNext = head;
        return newNode;
    }

    struct Node* curr = head;
    int i = 1;

    // Scorri fino al nodo precedente alla posizione desiderata
    while (curr != NULL && i < pos - 1) {
        curr = curr->pNext;
        i++;
    }

    // Se posizione non valida (troppo grande), non inserisce nulla
    if (curr == NULL) {
        free(newNode);
        return head;
    }

    // Inserisce il nuovo nodo nella posizione corretta
    newNode->pNext = curr->pNext;
    curr->pNext = newNode;

    return head;
}
```

Esercizio 1

```
int* sum_col(int r, int c, int[m][r]) {
    int* a = (int*)malloc(c * sizeof(int));
    for (int j = 0; j < c; j++) {
        a[j] = 0;
    }

    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            a[j] += m[i][j];
        }
    }

    return a;
}
```

```c
#include<stdio.h>
#include<limits.h>

int main() {
    int  a[5]= {25, 81, [2]= INT_MAX, 131046,131328};
    short int *p = (short*) a;
    char *q= (char*) a;
    q[16]= -1;

    printf("((*p - q[17] - *(q+18)) - 0x16 %d\n", (*p - q[17] - *(q+18)) -
     0x16);
    printf("(*(p+4) + *(q+14) + q[10]) == 128 %d \n", (*(p+4) + *(q+14) +
     q[11]) == 128);
    printf("((&a[5] - a) + q[5]) mod 2 %ld\n", ((&a[5] - a) + q[5]) % 2);
    printf("((int) (a + 5) - (int) (p+5) + q[5]) mod 2 %d \n", ((int) (a +
     5) - (int) (p+5) + q[5]) % 2);
    printf("(((q[17] | q[0]) & q[4]) % 5) mod 2 %d\n", (((q[17] | q[0]) &
     q[4]) % 5) % 2);
    printf("(p[4] | *(p+5)) & p[8] mod 256 %d\n", (p[4] | *(p+5)) & p[8] %
     256);

}


/* Mappa di memoria
 10011000
 00000000
 00000000
 00000000

 10001010
 00000000
 00000000
 00000000

 11111111
 11111111
 11111111
 11111110

 01011000
 00000000
 01000000
 00000000

 11111111
 10000000
 01000000
 00000000
 */
```