Prova scritta Programmazione Procedurale con Lab. - 26 Marzo 2024

Nome e Cognome: _____ Matricola: _____

1. [4 punti] Giustificare su foglio protocollo se le seguenti affermazioni sono vere o false (e cerchiare qui solo quelle vere): *float a=1.2, *const q= &a; const double b= 2.0, *p= &b;*
(A.) L'inizializzazione di a contiene una conversione di tipo; B. $^*p = 4.1$ è permesso; (C.) $^*q = 4.2$ è permesso; D. $q = (float^*)p$ è permesso; E. La precisione di a è maggiore di quella di b; F. $++a == 2$.

2. [7 punti] Scrivere cosa stampa il seguente programma, sapendo che le variabili a, b, c in *main* sono memorizzate rispettivamente agli indirizzi $0x7fff7be44946c$, $0x7fff7be449468$ e $0x7fff7be449464$.

```
1  static int a= 5;
2  int f(int* x, int y, int* z) {
3    static int res= 2;
4    if ((a++, a-=2) > 1) {
5      a ? res= (**&z+=3) + ++y, (y+= *x) : 2;
6      printf("%d %d %p %d\n", *x, y,  z+2, res);
7      res+= 3;
8      return (-res);
9    }
10   else
11     return -res;
12 }
13
14 int main(void) {
15   int a= 3, b= 010, c= 0X4;
16   b= f(&b, a, &c);
17   b= f(&a, c, &a);
18   printf("%d %d %d\n", a, b, c);
19 }
```

```
8 12 0x7ff7b692546c 11
6 14 0x7ff7b6925474 14
6 -17 7
```

3. [6 punti] Data la seguente *struct Node*, definire su foglio protocollo una funzione di nome *scambia()* che scambia il secondo elemento con il terzo elemento della lista passata. Per esempio, se la lista originale è 7-4-11, la nuova lista sarà 7-11-4. Controllare quindi se la lista ha per lo meno tre elementi.

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

4. [6 punti] Su foglio protocollo, scrivere la definizione di una funzione *matrix_multiplication* che prende in input due matrici e restituisce una terza matrice che contiene il risultato della moltiplicazione tra le prime due matrici. Controllare che le matrici siano moltiplicabili. tra loro *Suggerimento: creare la matrice risultato come int *res = (int *) malloc(rows * cols * sizeof(int)); indicizzandola quindi come un array.*

5. [7 punti] Cerchiare le affermazioni vere dato *int a[5]= {25, 81, [2]= INT_MAX, 131046,131328}; short int *p = (short*) a; char *q= (char*) a; q[16]= -1;* 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). Su foglio protocollo Scrivere la mappa di memoria e giustificare perché le seguenti affermazioni sono vere o false . A. $(^*p - q[17] - ^*(q+18)) - 0x16$ (B.) $((\&a[5] - a) + q[5]) \,\%\, 2$ C. $((int)(a+5) - (int)(p+5) + q[5]) \,\%\, 2$ D. $(((q[17] \mid q[0]) \,\&\, q[4]) \,\%\, 5) \,\%\, 2$

Esercizio 1
A: Vero, 1.2 viene convertito da double a float
B: Falso, p è un puntatore a costante e non si può modificare il valore della variabile indirizzata da p attraverso p
C: Vero, q è un puntatore costante, non si può modificare l'indirizzo contenuto in q, ma si può modificare il valore della variabile indirizzata da q
D: Falso, q è un puntatore costante, non posso modificare il suo valore
E: Falso. b ha precisione doppia, a ha precisione singola
F: Falso, 1.2 + 1 fa 2.2

Esercizio 5

10011000 q[0]
00000000 *p
00000000
00000000

10001010 q[4]
00000000
00000000
00000000

11111111
11111111
11111111 ← p + 5]
11111110

01100111
11111111
10000000
00000000

11111111
10000000
01000000 q[17]
00000000 *(q+18)

xxxxxxxx ← &a[5] e anche a + 5
xxxxxxxx
xxxxxxxx
xxxxxxxx

A: FALSO
(25 - 1 - 2) - 22 == 0

B: VERO
&a[5] - a == 5
5 - 0 (q[5] == 0)
5 % 2 == 1

C: FALSO
(int) (a + 5) - (int) (p+5) == 10
10 + 0 (q[5] == 0)
10 % 2 == 0

D:
10011000 |
01000000 =
11011000 & == 1 + 2 + 8 + 16 == 27
10001010 =
10001000 == 17
17 % 5 == 2
2 % 2 == 0

Esercizio 6
supponendo pFirst puntatore alla lista come globale

```
void scambia(void) {
   if (pFirst == NULL || pFirst -> pNext == NULL || pFirst -> pNext -> pNext == NULL) {
      printf("Non posso scambiare i due elementi\n");
      return;
   }
   struct Node* pFourth = pFirst -> pNext -> pNext -> pNext;
   struct Node* pTemp = pFirst -> pNext;
   pFirst -> pNext; = pFirst -> pNext -> pNext;
   pFirst -> pNext -> pNext = pTemp;
   pFirst -> pNext -> pNext -> pNext = pFourth;
}
```

Esercizio 5

```
int* matrix_multiplication(int r1, int c1, int m1[r1][c1], int r2, int c2, int m2[r2][c2]) {
   if (c1 != r2) {
      printf("ERRORE USCITA");
   }  return (NULL);
   else {
      int* res= (int *) malloc (r1*c2);
      for (int i=0; i < r1; i++) {
         for (int j=0; j < c2; j++) {
            int sum= 0;
            for (int k= 0; k < r2; k++)
               sum += m1[i][k] * m2[k][j];
            *(res + (i * r1+ j))= sum;
         }
      }
   return res;
}
```