Prova scritta Programmazione I FILA A - 18 Aprile 2018.

Nome e Cognome: _____ Matricola: _____

1. [4 punti] Riportare tutte le conversioni di tipo implicite; le due costanti *MAX hanno specificatore *LL* e *U* risp.. Scrivere inoltre il valore finale di *b* e di *c*, sapendo che il valore intero del carattere *'1'* è 49.

C = -36.0 b = 1 (TRONCAMENTO: SLIDE 19/20 SU CONVERSIONI)

```
1 float f(double a) {
2   char k= '1';
3   k+= k;
4   return (a - k + 1);
5 }
6
7 int main(void) {
8   long long int a= (LLONG_MAX-
      UINT_MAX)+1LL;
9   int b= a;
10  double c= f((b < a)? b : a);
11 }
```

LINEA 8: UINT_MAX DA UNSIGNED INT A LONG LONG INT SECONDA REGOLA SLIDE 11 SU CONVERSIONI
LINEA 9: a DA LONG LONG A INT
LINEA 10: b DA INT A LONG LONG
LINEA 1: f CHIAMATA CON LONG LONG CONVERTITO A DOUBLE
LINEA 2: '1' DA INT A CHAR
LINEA 3: k DA CHAR A INT E k+k DA INT A CHAR
LINEA 4: k DA CHAR A DOUBLE E 1 DA INT A DOUBLE
VALORE RITORNO DA DOUBLE A FLOAT
LINEA 10: RISULTATO f DA FLOAT A DOUBLE

2. [3 punti] Scrivere cosa stampa il seguente programma.

```
1 int r= 6, c = 5, s, i, j;
2 for(i=0; i<r; i++) {
3   for(s=1; s <= r-i; s++)
4     printf("-");
5
6   for(j=0; j <= i; j++) {
7     if (j==0 || i==0)
8       c = 2;
9     else
10      c = c*(i-j+1)/j;
11    printf("-%d", c);
12  }
13 printf("\n");
14 }
```

```
-------2
------2-2
-----2-4-2
----2-6-6-2
---2-8-12-8-2
--2-10-20-20-10-2
```

3. [3 punti] Scrivere cosa stampa il seguente programma, sapendo che *a* si trova all'indirizzo 0x7ffec4399ffe.

```
1 int a= 0xaa, i= -03, *b= &a;
2
3 for (int* p= &i; (a-= 1) ? ((*p)++, (--a)-1) : ((*p)+=2, a); ++(*
    p)) {
4   a= (a - i);
5   printf("%d %d OK\n", a, *p);
6   if (*p > 5) { a= !a + 1; continue; } }
7 printf("%d %p %d\n", a, ((long long*) b) + 2, i);
```

170 -2 ok
168 0 ok
164 2 ok
158 4 ok
150 0 ok
0 0x7ffee4399a00e 9

4. [4 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. Per prima cosa controllare che le matrici siano moltiplicabili. *Suggerimento: crearsi la matrice risultato come int *res = (int *) malloc(rows * cols * sizeof(int)); indicizzandola quindi come un array.*

5. [3 punti] Su foglio protocollo, descrivere (anche con brevi esempi di codice) le differenti zone di memoria dove possono essere memorizzate le variabili in C, e le loro caratteristiche.

6. [2 punti] Su foglio protocollo, scrivere due file, *main.c* e *write.c*: in *main.c* viene stampato una variabile *a* (di tipo *int*) definita in *write.c*, mentre in *write.c* viene stampata da una funzione di nome *write* una variabile *b* (di tipo *int*) definita in *main.c*. Scrivere i due file in modo che possano essere compilati anche separatamente.

7. [4 punti] Scrivere cosa stampa il seguente programma, sapendo che le variabili *a*, *b*, *c* in *main* sono memorizzate rispettivamente agli indirizzi $0x7ffeef2c9704$, $0x7ffeef2c9804$ e $0x7ffeef2c9904$.

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

5 10 0x7ffeef2c9908 8
10 17 0x7ffeef2c9908 11
17 29 0x7ffeef2c9908 17
3 38 6

8. [3 punti] Data la seguente *struct Node*, definire una funzione ricorsiva di nome *recursive_print* che stampa il campo *info* a partire dall'ultimo elemento di una lista di tali nodi. La funzione ha come parametro solamente una variabile di tipo *struct Node**.

```
1  struct Node {
2    int info= 0;
3    struct Node* pNext= NULL;
4  }
5
```

9. [4 punti] Cerchiare le affermazioni vere dato *long long a[3]= {129 + 256, INT_MAX, LLONG_MIN};* *short int *p = (short*) a; char *q= (char*) a; p[1]= 8192, p[3]= 8190, *(q+15)= 72, p[9]= 8192*2+1;* sapendo che i tre tipi usati occupano 8, 2, e 1 byte, e $8192 = 2^{13}$ (valori rappresentati in *little endian* e complemento a due). Scrivere la mappa di memoria e giustificare le affermazioni (vere o false). Gli operatori | e & ritornano rispettivamente l'*or* e l'*and* bit-a-bit dei due operandi. Supporre che i tipi *char* siano tipi con segno: il bit più significativo uguale a 1 rappresenta un valore negativo.

A. $(q[11] \mid q[23]) + q[1]$ B. $(*(p+5) - p[3]) \;\%\; 2$ C. $(((int)(p+10) - (int)(a+1)) + q[18]) \;\%\; 13$
D. $(\&a[3] - (a+1)) + q[1] <= 4$ E. $q[23] - q[0] + q[18]$ F. $(*(p+4) \;\&\; *(p+5)) == p[5]$

10. [1 punti] Quale comando di compilazione permette di ottenere anche la traduzione in linguaggio assembly del file *file.c*?

○ gcc -A file.c; ⊗ gcc -S file.c; ○ gcc -assembler file.c; ⊗ gcc -S -c file.c; ○ gdb file.c;
○ gcc -E file.c

10000001 q[0]
10000000
00000000
00001000

00000000
00000000
01111111 } p[3]
11110000

11111111 ← a+1, *(p+4)
11111111 ← *(p+5)
11111111
11111110 q[11]

00000000
00000000
01000000
00010010

00000000
00000000
10000000 q[18]
00000010

p+10 ←
00000000
00000000
00000000
00000001 q[23]

xxxxxxxx ← &c[3]
xxxxxxxx
xxxxxxxx

Ⓐ q[11] | q[23] + q[1] = 0 FALSA
11111111 = -1 + 1

Ⓑ *(p+5) - p[3] %2 = 1 VERA
1

Ⓒ (INT)(p+10) - (INT)(a+1) + q[18] = 13 ≥ 13
12 1 0 FALSA

Ⓓ &c[3] - (a+1) + q[1] 3 < 4 VERA
2 1

Ⓔ q[23] - q[0] + q[18] = 0 FALSA
-128 - (-127) + 1

Ⓕ *(p+4) && *(p+5) == *(p+5) VERA
==
11111111
11111110

ESERCIZIO 8

```
VOID RECURSIVE_PRINT (STRUCT NODE* P) {
    IF (P == NULL)
        RETURN;
    ELSE {
        RECURSIVE_PRINT (P -> PNEXT);
        PRINTF ("%d", P -> INFO);
        RETURN;
    }
}
```

ESERCIZIO 6

MAIN.C

```
#include<stdio.h>
EXTERN INT a;
void write(void);
INT b=5;
INT MAIN (VOID) {
    PRINTF("%d", a);
    WRITE ();
    RETURN 0;
}
```

WRITE.C

```
#include<stdio.h>
INT a=3;
EXTERN INT b;
VOID WRITE (VOID) {
    PRINTF ("%d", b);
    RETURN;
}
```

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


// FUNZIONE DELL'ESERCIZIO
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 = sum + m1[i][k] * m2[k][j];
                }
                *(res + (i*r1+j))= sum;
            }
        }

        return res;
    }
}

// MAIN DI TEST
int main(void) {
    int m1[3][3]= {1,2,3,4,5,6,7,8,9};
    int m2[3][2]= {1,2,3,4,5,6};
    int* m3= matrix_multiplication(3,3,m1,3,2,m2);
    for (int i=0; i < 3; i++)
        for (int j=0; j < 2; j++)
            printf("Elem [%d][%d]= %d\n", i, j, m3[i*3+j]);
    return 0;
}
```