Prova scritta Programmazione I - 14 Gennaio 2019 - FILA A.

Nome e Cognome: _____ Matricola: _____

1. [3 punti] Elencare le conversioni di tipo (... *da* ... a). Dato *USHRT_MAX = 65535*, scrivere il valore finale della variabile *b* sapendo che il carattere *a* ha valore 97 e le altre lettere seguono in ordine alfabetico.

```
1 long int fun2(long int p) {
2   return (p + 'c'); }
3
4 int fun1(int p) {
5   char a= 'a';
6   return fun2(p - a); }
7
8 int main(void) {
9   unsigned short a =  -3;
10  float b= fun1(a);
11 }
```

per la regola (slide 18 conversioni)
n + (USHRT_MAX + 1)
ad a viene assegnato
-3 + (65535 + 1)= 65533
b= 65533 - 97 + 99= 65535.0

linea 9: -3 da int a unsigned short
linea 10: a da unsigned short a int
linea 5: 'a' da int a char
linea 6: a da char a int in p-a
linea 6: p-a da int a long int in chiamata fun2
linea 2: 'c' da int a long int in p+c
linea 6: risultato fun2 da long int a int (ritorno fun1)
linea 10: risultato fun1 da int a float

2. [3 punti] Scrivere cosa stampa la seguente porzione di codice.

```
1 int main() {
2   int i, s, c= 0, c1= 0, a= 4, k=0;
3   for(i=1; i<=a; ++i, k=0) {
4     for(s=1; s<=a-i; ++s) {
5       printf("-");
6       ++c; }
7     while(k != 2*i-1) {
8       if (c < a-1) {
9         printf("*-"); ++c; continue;}
10      else {
11        ++c1; printf("%d-", (i+k-2*c1));}
12      ++k; }
13    c1 = c = k = 0;
14    printf("\n"); }
15  return 0; }
```

----1-
--*-0--1--2-
-*-*-1-0--1--2--3-
***-*-*-2-1-0--1--2--3--4-**

3. [3 punti] Scrivere cosa stampa la seguente porzione di codice.

```
1 int a= 0xfb;
2 while(a > 021 ? !!1: !!(a--, --a)) {
3   printf("%d \n", a);
4   if (a + 2 >= 0x10) {
5     a= -0x2;
6     continue; }
7   a+= 3; }
8 !(a+1) && a++;
9 printf("a: %d\n", a);
```

251
-4
-3
-2
-1
a: 0

4. [4 punti] Su foglio protocollo, scrivere la definizione di una funzione *sum_diagonals* che somma gli elementi sulla prima diagonale (*somma_d*1) e quelli sulla seconda diagonale (*somma_d*2) di una matrice $n \times n$ (matrice di *int*, ricevuta come parametro dalla funzione) e ritorna 1 se *somma_d1* == *somma_d2*, o 0 altrimenti.

5. [3 punti] Data la seguente struttura ed un puntatore ad inizio lista *struct Node *pFirst*, definire una funzione di nome *print_l* che stampa su video il valore del campo *info* per tutti i primi *l* elementi della lista. Se $l == 3$ e la lista è lunga 5, il quarto e quinto elemento non vengono stampati.

Esercizio 4

```
int sum_diagonals (int n, matrix[n][n]) {
    int somma_d1, somma_d2;
    for (int i= 0; i < n; i++) {
        somma_d1+= m[i][i];
        somma_d2+= m[i][n-i-1];
    }
   return (somma_d1== somma_d2);
}
```

Esercizio 5

```
void print_l(Node* pFirst, int l) {
    if(pFirst == NULL)
        printf(”Lista vuota!”);
    else {
        if (l > 0) {
            Node* pScan = pFirst;
            l-= 1;
            do{
                printf(”Info: %d\n", pScan->info);
                pScan = pScan->pNext;
            }while(pScan!= NULL && l > 0);
        }
    }
return;
}
```

Esercizio 6

```
int sumDigits(int num) {
    static int sum=0;
    if(num>0) {
        sum+= (sum%10);
        sumDigits(num/10);
    }
    else
        return sum;
}
```

```c
struct Node {
  int info= 0;
  struct Node* pNext= NULL;
}
```

6. [3 punti] Su foglio protocollo, scrivere una funzione ricorsiva di nome *digit_sum* che calcola la somma di tutte le cifre di un numero intero passato come input. Per esempio, se l'input è 1232 il risultato deve essere 8. Tutto il codice deve esser contenuto all'interno della funzione. Quanti *frame* vengono aperti se l'input è 1232?

7. [4 punti] Dire quali compilazioni provocano errore a causa del linker (e perché): 1) *gcc –o write write.c*, 2) *gcc –c main.c*, 3) *gcc –o main main.c*, 4) *gcc write.c main.c -o main*. In caso il punto 4) ritorni un errore, descrivere come può essere corretto. Infine, che tipo di *linkage* hanno *count*, *i*, e *write*? Cosa stampa il programma *output*? Elencare tutte le definizioni e dichiarazioni in ogni file.

main.c
```c
void mywrite(int *count);
extern int count;

int main(void) {
  do {
    mywrite(&count);
  }while(count >=0);
}
```

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

extern int i= 1;
int count= 4;
void mywrite(int *a) {
  static int count= -5;
  (*a)--;
  printf("%d\n", count= count - i);
}
```

1) errore: manca la definizione di main

3) errore: manca la definizione di mywrite e count

count (globale), i, mywrite hanno linkage esterno in tutti e due i file
count (locale a mywrite) ha no linkage

	Stampa
in main.c definiti: main	**-6**
in main.c dichiarati: mywrite, count	**-7**
in write.c definiti: i, count (globale), mywrite,	**-8**
count (locale in mywrite), a	**-9**
in write.c dichiarati: mywrite, count	**-10**

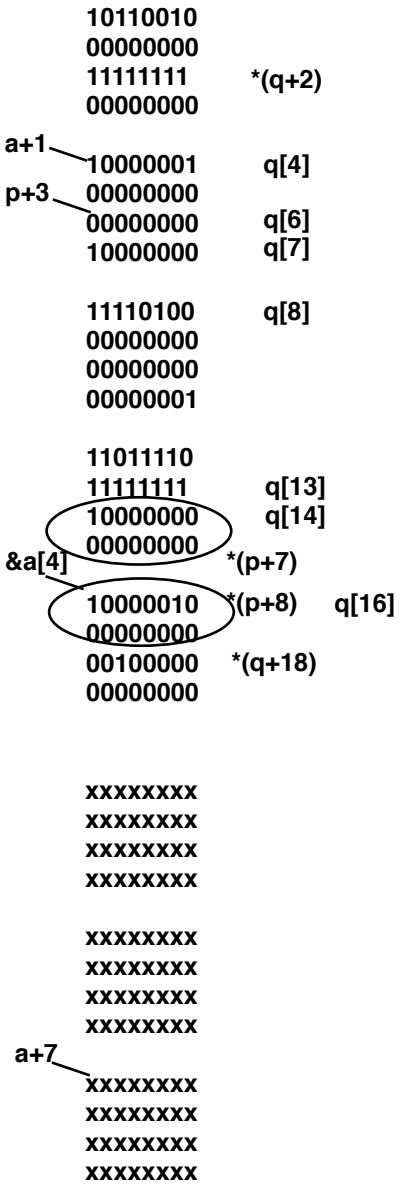
8. [4 punti] Scrivere cosa stampa il seguente programma.

```c
int a= 3;
int f1(int* b, int c) {
  static int a= 5;
  if (a <= 6) {
    int d= a++ + ++(*b) % c--;
    printf("%d %d %d %d\n", a, *b, c, d);
    d= f1(&c, *b);
    return d; }
  else return a; }
int f2(int* b, int c, int* d) {
  int e= a++ * ((*b)+=1) + ++(*d);
  *d= f1(&c, a+1);
  printf("%d\n", a);
  return *d*2; }
int main(void) {
  int a= 2, b= -2, c= 1;
  b= f2(&a, b, &c);
  printf("%d %d %d\n", a, b, c); }
```

6 -1 4 4
7 5 -2 6
4
3 14 7

9. [5 punti] Cerchiare le affermazioni vere dato *int a[5]= {13+2*32, 129, [2]= INT_MIN + 47, 130939, 131072*2 +65}; short int *p = (short*) a; char *q= (char*) a; q[2]= -1; p[3]= 128*2;* 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). Gli operatori | e & ritornano rispettivamente l'*or* e l'*and* bit-a-bit dei due operandi, $\sim$ è la negazione bit a bit, mentre >> (<<) rappresenta l'operatore di *shift* di n posizioni a destra (sinistra), inserendo 0 nelle posizioni eliminate (operazione fatta nel processore).

(A.) $(q[13] - q[14] + {}^*(p+7))\%2$ (B.) $({}^*(p+8) - {}^*(q+2) - q[6]) > 65)$ C. $(\&a[4] - (a+1)) + {}^*(q+18) - 7$
D. $(((int)(a+7) - (int)(p+3)) + q[16])\%87$ E. $({}^*(q+2) \,|\, q[8]) + (q[7] \,\&\, q[13])$ (F.) $((q[8] >> 3) + \sim q[4])\%130$



A: (-1 - (1) + 1) % 2 == -1 Quindi VERA

B: (65 - (-1) - 0) > 65 VERA

C: 3 + 4 - 7 == 0 FALSA
con &a[4] - (a+1) == 3 ci sono tre interi tra quei due puntatori

D: (22 + 65) % 87 == 0 FALSA
con (int) (a+7) - (int) (p+3) == 22 ci sono 22 byte tra quei due indirizzi

E: -1 + 1 == 0 FALSA
con
***(q+2) == 11111111 l**
q[8] == 11110100
dato che *(q+2) sono tutti 1, in or con qualsiasi altro operando il risultato è sempre 11111111 (che vale -1 in complemento a 2)
e
q[7] == 10000000 &
q[13] == 11111111
dato che q[13] sono tutti 1, in and con qualsiasi altro operando il risultato sempre l'altro operando, in questo caso 10000000 (che vale 1)

F: (5 + 126) % 130 == 1 VERA
con
q[8] vale 11110100, portato nei registri del processore viene rappresentato in big endian, cioè 00101111
00101111 >> 3 == 00000101
che letto in big endian vale 5
e
q[4] vale 10000001
~q[4] vale 01111110
che vale 126