Prova scritta Programmazione I - 11 Febbraio 2019 - FILA A

Nome e Cognome: _____ Matricola: _____

1. [3 punti] Riportare le conversioni di tipo implicite e scrivere quanto valgono alla fine le variabili a, x, i ($INT\_MAX == 2147483647$).

a == 4, x == 0, i == -2147483646 (INT_MAX + 1 fa INT_MIN, +2 fa INT_MIN + 2 == -2147483646 dato che INT_MIN == -2147483648)

```
1 int x = 0U, i = INT_MAX + 3;
2 char a= (char) 10, b= (char) 40, c= (
      char) 100;
3 a= (a*b) / c;
4 unsigned int limit = 10U;
5 long n = 2L;
6 if ( i < limit )
7   x = 3.0;
```

linea 1: 0U da unsigned int a, INT_MAX + 3 da long a int

linea 3: a, b, c da char a int per integer promotion, il risultato da int a char

linea 6: i convertito da int a unsigned int

linea 7: 3.0 da double a int

La condizione alla linea 6 è falsa: il valore di i diventa -2147483646 + (UINT_MAX + 1) che è sicuramente maggiore del valore di limit

2. [4 punti] Scrivere cosa stampa il seguente programma.

```
1 int i, j, N= 4;
2 for(i=1; i<=N; i++) {
3   for(j=1; j<i; j++)
4     printf("-");
5   printf("%d", i);
6   for(j=1; j <= ((N - i) * 2
      - 1); j++)
7     printf("-");
8   if(i != N)
9     printf("%d", i);
10    printf("\n"); }
11 for(i=N-1; i>=1; i--) {
12   for(j=1; j<i; j++)
13     printf("-");
14   printf("%d", i == j / 2? j
       : 0);
15   for(j=1; j <= ((N - i) * 2
       - 1); j++)
16     printf("-");
17   printf("%d\n", i); }
```

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

3. [3 punti] Scrivere cosa stampa la seguente porzione di codice sapendo che c si trova all'indirizzo 0x7ffee7419f00.

```
1 int a= 0x2e, b = 06, *c= &a;
2 do {
3   for (int i= 2; i ? --i : !!0 ; i--) {
4     a/=2; b = a;
5     printf("%d %d\n", a, i);
6     if (a == b) break; }
7 } while (a--, a && b);
8 printf("%d %p\n", a, ((short*) c) + a);
```

23 1
11 1
5 1
2 1
0 1
-1 0x7ffee7419efe

4. [3 punti] Su foglio protocollo, scrivere la definizione di una funzione *arrayCreation* che ha come parametro solamente *unsigned int lunghezza*. Questa funzione crea un nuovo array, lo inizializza con valori da 0 a *lunghezza* − 1, e ritorna l'array alla funzione che lo ha chiamato in modo che lo possa utilizzare. Non utilizzare variabili globali o *static*; tutto il codice necessario deve essere contenuto all'interno della funzione.

5. [4 punti] Data la seguente struttura, definire una funzione di nome *del_head_if_in* che prende come parametro *int infoToRemove*, e rimuove l'elemento in testa alla lista solamente se il campo *info* di questo elemento ha valore uguale a *infoToRemove*. In caso invece tale valore sia differente, aggiunge in testa un nuovo elemento della lista con campo *info* uguale a *infoToRemove*.

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

Esercizio 4

```
unsigned int* arrayCreation(unsigned int length) {

   unsigned int *p = (unsigned int*) calloc(length, sizeof(unsigned int));
   for (unsigned int i= 0; i < length; i++)
      *(p+ i) =  i;

   return p;
}
```

Se l'array non viene creato in memoria dinamica, viene deallocato all'uscita della funzione e quindi non è più disponibile

Esercizio 5

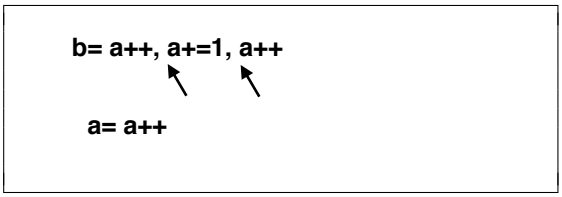
```
// supponiamo che il puntatore alla lista sia globale e si chiami pFirst

void del head_if_in(int infoToRemove) {
   if (pFirst == NULL)
      printf("Nessun elemento da cancellare!");
   }
   else {
      if  (pFirst -> info == infoToRemove) {
         Node* temp= pFirst-> pNext;
         free(pFirst);
         pFirst= temp;
      else {
         Node *pNew= (Node*) malloc(sizeof(Node));
         pNew->info= infoToRemove;
         pNew-> pNext= pFirst;
         pFirst= pNew;
      }
   }
   return;
}
```

Esercizio 6

```
int stringLength(char* stringa) {
   if (stringa == NULL)
      return 0;
   if (*stringa == '\0')
      return 0;
   return 1 + stringLength(stringa + 1);
}
```

6. [3 punti] Su foglio protocollo, scrivere una funzione ricorsiva di nome *stringLength* che calcola la lunghezza di una stringa (di lunghezza sconosciuta alla funzione) passata come parametro. Per esempio, la funzione ritorna 12 se viene passata la stringa"Hello, World".

7. [2 punti] Scrivere una espressione con 3 effetti collaterali su una variabile a e 1 effetto collaterale su b, che *NON* generi un warning *multiple unsequenced modifications*. Scrivere una seconda espressione con 2 effetti collaterali su a che invece generi tale warning. Evidenziare in entrambi i casi i *sequence points*.



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

```
1  int f1(int* x, int *y, int z) {
2    static int a= 0;
3    a++;
4    int r= 1;
5    if (a < 4) {
6      r+= (*x)++ + ++(*y), (*y+= z+1);
7      r+= f1(x, y, z+1);
8      printf("%d %d %d %d\n", *x, *y, z, r);
9      return (r);
10   }
11   else
12    return r;
13 }
14 int main(void) {
15   int a= 2, b= 4, c=-3;
16   b= f1(&b, &c, a);
17   printf("%d %d %d\n", a, b, c);
18 }
```

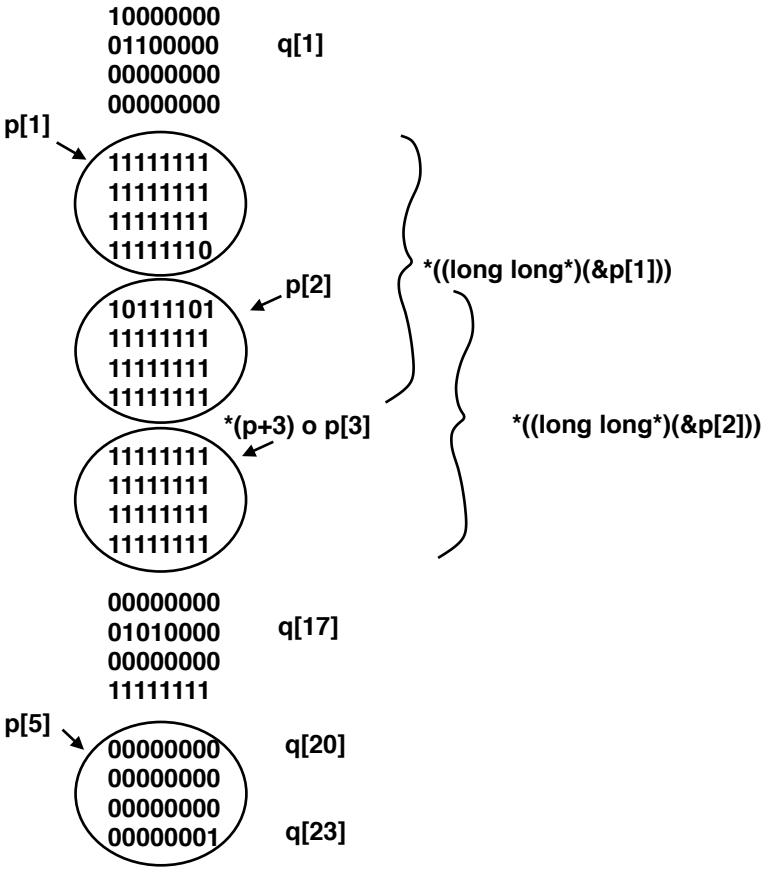
7 12 4 15
7 12 3 23
7 12 2 26
2 26 12

9. [4 punti] Cerchiare le affermazioni vere dato *long long a[3]= {1537, -67, (LLONG_MAX + 1) + 512}; int *p = (int*) a; char *q= (char*) a; p[1]= INT_MAX, p[4]+= 2048, q[19]= ∼q[19];* sapendo che i tre tipi usati occupano 8, 4, e 1 byte, con 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, ∼ è la negazione bit a bit.

A. $((q[17] \mid q[20]) + q[1]) \ \% \ 4$ B. $(*(p+3) - p[2]) \ \% \ 11$ (C.) $(((int)(p+12) - (int)(a+4)) - q[1]) < \sim q[23]$ (D.) $(\sim (p[3] \ \& \ p[1])) == p[5]$ (E.) $*((long\ long^*)(\&p[1])) < *((long\ long^*)(\&p[2]))$ (F.) $((long\ long^*)(\&p[1])) < ((short\ int^*)(\&p[2]))$

10. [1 punti] Quali affermazione/i sono vere? Il linguaggio C è ...

⊗ staticamente tipato; ○ fortemente e staticamente tipato; ○ non tipato; ○ sotto tipato.
○ retro tipato. ⊗ debolmente tipato.



A: q[17] | q[20] == q[17], q[17] + q[1] == 16
16% 4 == 0 quindi FALSA

B: *(p+3) == -1, p[2] == -65 (equivale a -1 a cui si sottrae 2 e 64, i bit che da 1 diventano 0)
-66 % 11 = 0, quindi FALSA

C: (int) (p+12) – (int) (a+4) == 4 e q[1] == -1
~q[23] == 11111110 == 127
4 - (-1) == 5
5 < 127 quindi VERA

D: p[3] & p[1] == p[1] dato che p[3] sono tutti 1
la negazione bit a bit del risultato equivale a
che è uguale a p[5], quindi VERA

00000000
00000000
00000000
00000001

E: dal bit più significativo si capisce che entrambi i valori *((long long*)(&p[1])) e *((long long*)(&p[2])) sono negativi.
((long long)(&p[2])) equivale a -1 da cui poi sottraggo 2 + 64 - 67== mentre
((long long)(&p[1])) equivale a -1 da cui sottraggo potenze di due più elevate (2 alla 30, 2 alla 32, 2 alla 37).
Quindi *((long long*)(&p[1])) < *((long long*)(&p[2]))
VERA

F: &p[1] è un indirizzo che precede &p[2] in memoria, che sia convertito a indirizzo a short o a long long non fa mutare il suo valore, quindi VERA