Prova scritta Programmazione I - 9 Luglio 2019

Nome e Cognome: _____ Matricola: _____

1. [3 punti] Riportare le conversioni di tipo implicite e scrivere quanto valgono alla fine le variabili x, k, z, sapendo che UINT_MAX vale 4294967295 (le conversioni da *_MAX non comportano conversioni).

```
1  unsigned int x = UINT_MAX, k= 10U, z =
     -1,;
2  int a= -2.5l;
3  long long y = LLONG_MAX;
4  long j= 2LL;
5  x = y - x;
6  k = a < k;
7
```

z == (UINT_MAX +1) - 1 == 4294967295 x == 0 (identico ad esempio su slide 21 conversioni)
linea 1: -1 da int a unsigned int
linea 2: 2.5 da long double a int
linea 4: 2LL da long long int a long int
linea 5: d da unsigned int a long long, y-x da long long a unsigned int (regola 2 slide 12 su conversioni)
linea 6: a da in a unsigned int, il risultato a < k ha tipo int che viene quindi convertito a unsigned int
alla linea 6 a viene convertito da -2 a (UINT_MAX +1) -2 che non è minore di 10, quindi la comparazione è falsa e k == 0

2. [3 punti] Scrive una funzione *main()* che disegni su terminale la seguente composizione di numeri. Il numero di righe è preso come input da tastiera.

```
0
0 1
0 2 4
0 3 6 9
0 4 8 12 16
```

http://www.techcrashcourse.com/2016/02/c-program-to-print-multiplication-table-pattern.html

3. [3 punti] Scrivere cosa stampa la seguente porzione di codice. Che valore contiene la variabile b alla fine del programma?

```
1  int a= 0x3c, i= -05, *b= &a;
2  for (int* p= &i; (a++, (*p)++) ? (++(*p), (a--)-1) : ((*p)
     +=3, a-1); (*p)++) {
3  --a;
4  printf("%d %d \n", a, *p);
5  if (*p > 3) {
6  a= (!(-a) && a++) ? 3 : 2;
7  break; }
8  else continue; }
9  printf("%d\n", a);
10
```

59 -3
58 0
57 3
56 6
2

4. [3 punti] Definire una funzione che prende in input una matrice quadrata di dimensione n, e scambia i valori della prima e dell'ultima colonna: la prima colonna diventa uguale all'ultima e viceversa.

5. [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 9 se viene passata la stringa"Mad World".

6. [3 punti] Cerchiare se vero dato *int a= 2, *p= &a;* A. $*\&a$ è un *rvalue*; B. $**\&p$ è un *lvalue*; C. $a = 2, a{+}{+};$ non genera warning; D. $a = 2, a{+}{+};$ contiene 2 effetti collaterali; E. $a{+}{+} \parallel a{+}{+};$ genera warning; F. $a{+}{+}\,?\,a{+}{+} : a{+}{+};$ genera warning; G. p ha tipo *int*; H. $!{*}p == -2;$ I. $\&a == p.$

Vere B, C, D, I

7. [2 punti] Dato *flaot 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. $++a == 2.$

Vere A, C

8. [3 punti] Su foglio protocollo, scrivere il contenuto di due file, *main.c* e *secondary.c*. I due file devono essere compilati insieme producendo con successo l'eseguibile di nome *myexe*: scrivere il comando per fare ciò. I due file devono poter essere compilati anche indipendentemente per produrre un file oggetto; scrivere il comando per fare ciò. In *main.c* ci devono essere esattamente un tentativo di definizione che rimane dichiarazione, una definizione di variabile e una di funzione entrambi con *external linkage*, una dichiarazione di funzione con *external linkage*. In *secondary.c* ci devono essere esattamente una definizione di funzione con *external linkage*, una variabile con *internal linkage* e due variabili con *no linkage*.

9. [4 punti] Data la seguente struttura, definire una funzione di nome *ins_head* che prende come parametri un puntatore a *struct Node* (il puntatore al primo elemento della lista) e un valore *int*, e inserisce un nuovo elemento in testa alla lista (cioè come primo elemento), con il campo *info* assegnato al valore passato. La funzione non crea il nuovo elemento e stampa un errore, se il campo *info* del nuovo elemento è minore del campo *info* dell'elemento successivo in lista.

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

10. [4 punti] Cerchiare le affermazioni vere dato *int a[4]= {7+4*64, INT_MIN + 39, [2]= 131002, 131072/2+111}; short int *p = (short*) a; char *q= (char*) a; *(q+3)= -1; *((short int*) &q[5])= 257;* 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, inserendo 0 nelle posizioni eliminate (operazione fatta nel processore).

A. $p[3] > SHRT\_MIN$ B. $q[5] - q[6] + q[10] - q[14]$ C. $((\&a[4] - a) + -p[5])\ \%2$ D. $(((int)\ (a+2) - (int)\ \&q[2]) + {}^*(q+14))\ \%2)$ E. $({}^*(q+3)\ \&\ q[1]) + \ q[2]$ F. $((q[12] >> 4) \mid q[4]) >= 35$

Vere A,C,D,E,F (vedere correzione 28 Gennaio 2019)

Esercizio 8

main.c

```
int a;
int a= 10;

void fun(int);
int main(void) {
 fun(a);
}
```

secondary.c

```
static int c= 4;
void fun(int);
void fun(int b) {
int d= 4;
c+= b + d;
}
```

```
gcc -c main.c
gcc -c secondary.c
gcc -o myexe main.o secondary.o (o .c)
```

Esercizio 9

```
void ins_head(struct Node** pFirst, int infoField) {

if (*pFirst != NULL)
  if (((*pFirst) -> pInfo) < infoField) {
     printf("Errore parametro infoField minore del successivo
elemento")
    return;
}
else
 struct Node *pNew = (struct Node*) malloc(sizeof(struct Node));
 pNew-> pNext= NULL;
 pNew-> pInfo= infoField;
 if(*pFirst == NULL)
  *pFirst = pNew;
 else {
  pNew-> pNext= *pFirst;
  *pFirst= pNew;
}
return;

}
```

Esercizio 4

```
void interchangeFirstLast(int n, int m[n][n])
{
   // swapping of element between first
   // and last columns
   for (int i = 0; i < n; i++) {
      int t = m[i][0];
      m[i][0] = m[i][n - 1];
      m[i][n - 1] = t;
   }
}
```

Esercizio 5

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