Prova scritta Programmazione Procedurale - 13 Aprile 2022

Nome e Cognome: _____ Matricola: _____

1. [5 punti] Descrivere cosa accade quando si converte un valore ad un tipo intero UNSIGNED, con esempi (esporre le relative regole di conversione di tipo).

2. [6 punti] Su foglio protocollo, definire una funzione *crea_array* che prende come parametro un valore *unsigned short n* e restituisce come risultato un array di n elementi che corrispondono ai primi elementi della serie di Fibonacci (1, 1, 2, 3, 5, 8, 13, etc).

3. [6 punti] Data la seguente *struct Node* definire su foglio protocollo una funzione di nome *inverti_lista* che prende come parametro una lista fatta di tali strutture e restituisce una nuova lista in ordine inverso: se la lista originale è 7-4-11, la lista ritornata sarà 11-4-7.

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

4. [6 punti] Per ogni identificatore di variabile e funzione scrivere se è definito o dichiarato, ed il suo linkage. Quali identificatori sono visibili in altri file?

```
1   #define B 3
2   int a= B;
3   int a;
4   extern int b;
5   extern int cmp(float, float);
6   static double area(double);
7
8   int* func(int c, int v) {
9   static double e= 4.0;
10  double* f= &e;
11  auto int q= 4;
12  extern int a;
13  register int rr;
14  // Altri comandi
15  }
16
```

linea 3: a definito / ext link.
linea 4: a tentativo di definizione che rimane dichiarazione dato che a definito alla linea 2 / ext link.
b dichiarato / ext. link.
cmp dichiarato / ext. link.
are dichiarato / int. link.
func definito / ext. link.
c e v definito / no link.
e definito / no link.
f definito / no link.
q definito / no link.
a dichiarato / external link. (a alla linea 2)
rr definito / no link.

In altri file sono visibili tutti gli identificatori con external linkage

5. [7 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). L'operatore & ritorna l'*and* bit-a-bit dei due operandi a cui è applicato.

A. $p[3] > SHRT\_MIN$ B. $((\&a[4]-a)+-p[5])\ \%2$ C. $(((int)\ (a+2)-(int)\ \&q[2])+\ ^*(q+14))\ \%2)$
D. $(^*(q+3)\ \&\ q[1])+\ \ q[2]$

TUTTE VERE

Esercizio 1
1)Slide “conversions to unsigned integer types”
Quando si assegna un valore ad un tipo unsigned, il valore è preservato se tra 0 e UType_MAX (UType è il tipo a cui si assegna il valore, per esempio UINT_MAX se unsigned int). Per valori al di fuori di questo intervallo, il valore che risulta assegnato si trova addizionando/sottraendo (UType_MAX + 1), tante volte quanto.è necessario affinché il valore risultante sia nell’intervallo di UType. Per esempio
unsigned short n = 1000;
n = - 1
il valore in n è –1 + (USHRT_MAX + 1) = USHRT_MAX
2) se il valore assegnato ad un UType è in virgola mobile, allora la parte frazionaria è scartata

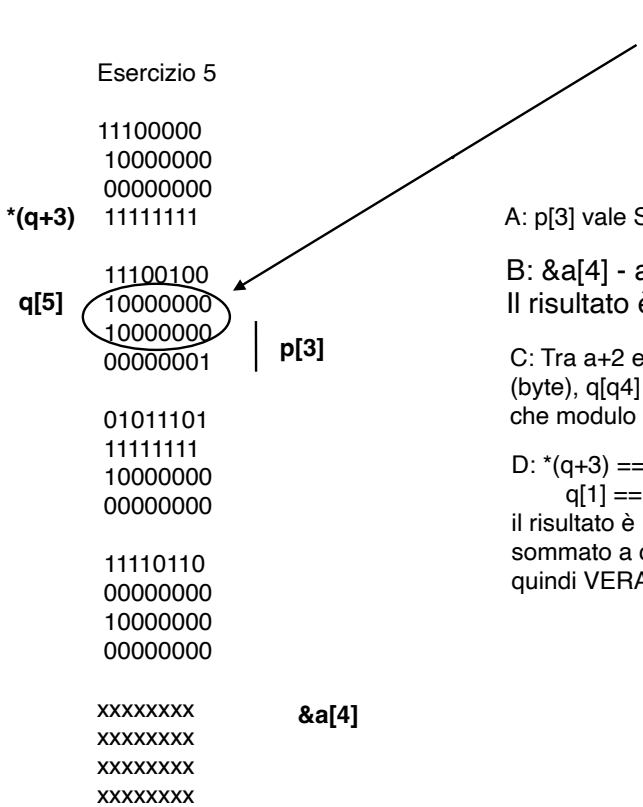
Esercizio 2

```
unsigned int* crea_array(unsigned short n) {
  if (n == 0)
    return NULL;
  unsigned int* array = NULL;
  array= (unsigned int*) calloc(n, sizeof(unsigned int));
  unsigned int fib1 = 0;
  unsigned int fib2 = 0
  unsigned int fib = 1;
  array[0] = fib;
  for (int i=1; i < n; i++) {
    fib1 = fib2;
    fib2 = fib;
    fib = fib1 + fib2;
    array[i] = fib;
  }
  return array;
}
```

Esercizio 3

```
struct Node* inverti_lista(struct Node* lista_da_invertire) {
  pFirst = NULL;
  struct Node* pScan = lista_da_invertire;
  while (pScan != NULL) {
    Node *pNew = (struct Node*) malloc(sizeof(struct Node));
    pNew -> info = pScan -> info;
    pNew -> pNext= NULL;
    if(pFirst == NULL)
      pFirst = pNew;
     else {
      pNew-> pNext= pFirst;
      pFirst = pNew;
     }
    pScan = pScan -> pNext;
  }
  return pFirst;
}
```

Esercizio 5



con *((short int*) &q[5])= 257, prima si controlla l’indirizzo di &q[5], poi se si converte a puntatore a short int, si inserisce 257 nei successivi due byte

A: p[3] vale SHRT_MIN + 1, quindi VERA

B: &a[4] - a == 4 (ci sono 4 interi tra i due indirizzi) + -1 (-q[5]). Il risultato è 3%2 che equivale a 1, quindi VERA

C: Tra a+2 e &q[2] la differenza è 6 (byte), q[q4] vale 1, la loro somma è 7 che modulo 2 fa 1, quindi VERA

D: *(q+3) == 11111111 in and a bit a bit con q[1] == 10000000 il risultato è 10000000 (che equivale a 1), sommato a q[1] (che vale 0), il risultato è 1, quindi VERA