

Prova scritta Programmazione Procedurale con Lab. - 17 Gennaio 2023 - FILA A

Nome e Cognome: \_\_\_\_\_

Matricola: \_\_\_\_\_

1. **5 punti** Elencare le conversioni di tipo implicite (... da ... a). Riportare il valore finale di *c* sapendo che *USHRT\_MAX* = 65535. Che precisione ha *b*?

```
1 long fun(long long z) {
2     char a = ('h' + 'h') / 2;
3     return (a - z);
4 }
5
6 int main(void) {
7     unsigned int a = 'a' + 1L;
8     float b = fun(a);
9     unsigned short c = -(int) (b-1);
10 }
```

linea 7: 'a' convertito da int a long int

linea 7: il risultato della somma convertito da long int ad unsigned int

linea 8: a parametro di fun convertito da unsigned int a long long int

linea 2: l'espressione a destra dell'uguale è convertita da int a char

linea 3: a convertito da char a long long int

linea 3: il risultato della differenza convertito da long long a long

linea 8: il risultato della funzione convertito da long a float

linea 9: 1 convertito da int a float

linea 9: il risultato a destra dell'uguale convertito da int (dopo conversione esplicita)

a unsigned short

La precisione di b è singola

Il valore finale di c è 65531 calcolato come -5 + (USHRT\_MAX + 1)

c vale infatti -5, 'h' - ('a' + 1) vale 6 considerando la tabella ASCII

2. **6 punti** Scrivere cosa stampa la seguente porzione di codice.

```
1 int main() {
2     int a= 0777 - 0x1f1;
3     printf("%d\n", a);
4     while ((a-- || a--) ? a--1 : 0) {
5         if (!(--a && --a))
6             break;
7         else
8             printf("%d\n", a);
9             continue; }
10    a + 1, a++, a!=a;
11    printf("a: %d\n", a); }
```

14  
10  
6  
2  
a: 1

3. **6 punti** Data la seguente *struct* scrivere la definizione una funzione di nome *copia\_lista\_array* che prende come parametro una lista e ritorna un array (creato nella funzione) che contiene gli stessi elementi (solo campo *info*) della lista.

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

4. **6 punti** Dire quali compilazioni provocano errore a causa del linker (e perché): 1) *gcc -o main main.c*, 2) *gcc -o write write.c*, 3) *gcc write.c main.c -o main*. In caso il punto 3) ritorni un errore, descrivere come può essere corretto. Dopo aver corretto l'errore, che tipo di *linkage* hanno *count*, *i*, e *mywrite*, ed in quale file sono definite? Cosa stampa il programma?

main.c

```
void mywrite(int *count);
static int count;

int main(void) {
    do {
        mywrite(&count);
    } while(count <= 3);
}
```

write.c

```
#include <stdio.h>
static int i = 1;
int count;
int count= 4;
void mywrite(int *a) {
    static int count= -3;
    (*a)++;
    printf("%d\n", count= count - i);
}
```

- 1) errore manca la definizione di mywrite()  
2) errore manca la definizione di main  
3) questo comando non causa nessun errore

Output:

-4

-5

-6

-7

main.c

mywrite dichiarata - external link., count definita - internal link.

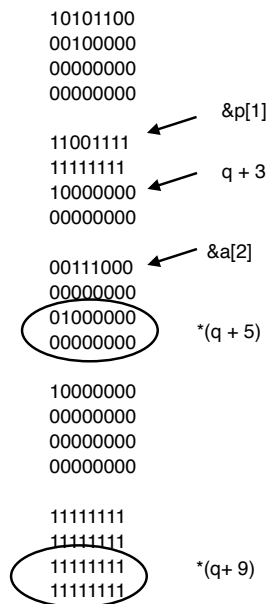
write.c

static int i definita - internal link., count globale è tentativo di definizione che rimane dichiarazione, count globale definita sotto con external linkage, static int count locale a mywrite definita - no link.

mywrite definita - linkage esterno

5. 7 punti Cerchiare le affermazioni vere dato  $\text{int } a[5] = \{1077, 131059, 131100, -9, \text{UINT\_MAX}\};$   
 $\text{int } *p = (\text{int} *) a; \text{short int } *q = (\text{short} *) a; p[3] = 1;$  sapendo che i due tipi usati occupano 4 e 2 byte, e  
 $131072 = 2^{17}$  (valori rappresentati in *complemento a due* e *little endian*). Rappresentare la zona di memoria  
in cui è memorizzato l'array. A.  $(*(q + 5) - *(q + 9)) \% 2;$  B.  $*(p + 12) < 0;$  C.  $q + 3 > \&p[1];$   
D.  $((\text{int})(\&a[2]) - (\text{int})(q + 3)) \leq 2;$  E.  $(p + 7) - (p + 2) < 0x000000f.$

Mapa di memoria:



Esercizio 10

A:  $2 - (-1) = 3 \% 2 == 1$

B:  $p + 12$  è al di fuori dell'intervallo in memoria, il valore di  $*(p - 12)$  è quindi indefinito

C:  $q + 3$  è un indirizzo più grande del secondo elemento dell'array  $p[1]$

D: Tra i due indirizzi ci sono due byte, la differenza fa quindi 2, che è minore uguale di 2

E: la differenza da 5, che è minore di 15

Esercizio 3

```
int* copia_lista_array(struct Node* lista) {
    if (lista == NULL){
        printf("Lista vuota!\n");
        return NULL;
    }
    int contatore = 0;
    struct Node* pScan = lista;
    while (pScan != NULL) {
        contatore++;
        pScan = pScan -> pNext;
    }
    int* array = (int*) calloc(contatore, sizeof(int));
    pScan = lista;
    for (int i = 0; i < contatore; i++) {
        *(array + i) = pScan -> info;
        pScan = pScan -> pNext;
    }
    return array;
}
```