

## Esercizio 1

Discutere le funzioni: il loro utilizzo, il passaggio di parametri, il valore di ritorno, la dichiarazione e la definizione.

### Correzione

Occorreva indicare e argomentare brevemente la differenza tra la dichiarazione, la definizione e l'utilizzo di una funzione, ad esempio:

```
int sum (int, int);

int sum (int a, int b);
{
return a+b;
}

...
int c=2,d=3;
printf("La somma di %d e %d e' %d.\n",c,d,sum(c,d));
...
```

Spiegando anche come avviene il passaggio parametri in input (per valore e per riferimento) e in output (valore di ritorno, oppure passando il riferimento in input).

## Esercizio 2

Discutere la moltiplicazione binaria e svolgere l'operazione  $8_{10} \times 3_{10}$  usando la rappresentazione binaria in complemento a 2 con 6 bit.

### Correzione

Occorreva discutere brevemente come viene effettuata la moltiplicazione tra numeri binari, notando che moltiplicando per  $2^n$  equivale ad aggiungere n zeri. Il numero  $3_{10}$  si scrive  $000011_2$ , mentre il numero  $8_{10}$  si scrive  $001000_2$ . Il loro prodotto binario e' semplicemente il numero  $011000_2$  che equivale infatti a  $24_{10}$  ( $16+8$ ).

## Esercizio 3

Leggere attentamente il seguente listato di codice. Il main presenta svariati errori; dopo averli individuati e motivati, correggerli riscrivendo la funzione main e illustrare cosa stampa il programma dopo aver letto la stringa "aveva".

```
1 #include <stdio.h>
2
3 int fun(char *a, char *b)
4 {
5     int i=0,j=0,m;
6     int n=0;
7     for(i=0;a[i]!='\0';i++)
8         if (a[i]!=' ')
9             {
10                m=0;
11                for(j=i+1;a[j]!='\0';j++)
12                    if (a[i]==a[j])
13                        {
14                            m++;
15                            a[j]=' ';
16                        }
```

```

17     b[n++] = m + '1';
18     b[n++] = a[i];
19 }
20 b[n] = '\0';
21 return n/2;
22 }
23
24 int main(void)
25 {
26     char str[20], out[20];
27     scanf("%s", str[20]);
28     fun(str, out);
29     printf("La stringa %c contiene %c e %d lettere.\n", str, out, fun(str, out));
30     exit(0);
31 }

```

## Correzione

La dimensione massima che puo' raggiungere `out`, dato che la funzione `fun` conta di quanti caratteri uguali e' composta `str` che conterra' massimo 19 caratteri (il 20imo serve per il terminatore di stringa) deve essere 39 (19x2+1 per il terminatore) infatti nel caso peggiore la stringa e' composta da 19 caratteri distinti ("abcd..." contiene "1a1b1c1d...").

A riga 27 la `scanf` deve leggere un stringa quindi bisogna passare il puntatore al primo elemento, ovvero la stringa stessa.

A riga 28 e 29 viene richiamata due volte la funzione `fun`. Affinche' il programma funzioni correttamente bisogna eliminare una i queste due occorrenze e mantenerne solo una.

A riga 29 il template va passato con i doppi apici e non quelli singoli, inoltre i `%c` vanno sostituiti con `%s`.

Una volta corretti questi errori dopo aver letto la stringa "aveva" il programma stamperebbe

La stringa ave contiene 2a2v1e e 3 lettere.

Infatti la stringa `str` viene modificata dalla funzione `fun`. Per correggere anche questo errore bisogna spezzare l'output della `printf` nel seguente modo per ottenere il seguente main corretto e la stringa:

La stringa aveva contiene 2a2v1e e 3 lettere.

```

int main(void)
{
    char str[20], out[39];
    scanf("%s", str);
    printf("La stringa %s ", str);
    printf("contiene %s e %d lettere.\n", out, fun(str, out));
    exit(0);
}

```

Altra versione equivalente proposta da alcuni studenti:

```

int main(void)
{
    char str[20], str2[39];
    int out;
    scanf("%s", str);
    printf("La stringa %s ", str);
    out = fun(str, str2);
    printf("contiene %s e %d lettere.\n", str2, out);
    return 0;
}

```