S2/L5 - PROGETTO

Ci viene presentato il seguente programma

```c
#include <stdio.h>

void menu ();
void moltiplica ();
void dividi ();
void ins_string();


int main ()

{
        char scelta = {'\0'};
        menu ();
        scanf ("%d", &scelta);

        switch (scelta)
        {
                case 'A':
                moltiplica();
                break;
                case 'B':
                dividi();
                break;
                case 'C':
                ins_string();
                break;
        }

return 0;

}


void menu ()
{
        printf ("Benvenuto, sono un assitente digitale, posso aiutarti a sbrigare alcuni compiti\n");
        printf ("Come posso aiutarti?\n");
        printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");

}


void moltiplica ()
{
        short int  a,b = 0;
```

```
46          printf ("Inserisci i due numeri da moltiplicare:");
47          scanf ("%f", &a);
48          scanf ("%d", &b);
49
50          short int prodotto = a * b;
51
52          printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
53  }
54
55
56  void dividi ()
57  {
58          int  a,b = 0;
59          printf ("Inserisci il numeratore:");
60          scanf ("%d", &a);
61          printf ("Inserisci il denumeratore:");
62          scanf ("%d", &b);
63
64          int divisione = a % b;
65
66          printf ("La divisione tra %d e %d e': %d", a,b,divisione);
67  }
68
69
70
71
72
73  void ins_string ()
74  {
75          char stringa[10];
76          printf ("Inserisci la stringa:");
77          scanf ("%s", &stringa);
78  }
```

1. Esso è una sorta di assistente digitale che può aiutare l'utente a fare tre cose: moltiplicare due numeri, dividere due numeri (nel punto 3 spiego come in realtà non è ciò che va a fare effettivamente) e inserire una stringa. Il programma inizia con la funzione *main()*, la quale visualizza un menu di opzioni per l'utente. Quest'ultimo può quindi scegliere un'opzione premendo il tasto corrispondente. Se l'utente sceglie l'opzione *A*, viene chiamata la funzione *moltiplica()*. Tale funzione richiede all'utente di inserire due numeri e poi calcola il prodotto di questi due numeri. Il risultato viene quindi visualizzato sullo schermo. Se l'utente sceglie l'opzione *B*, viene chiamata la funzione *dividi()*. Questa funzione richiede all'utente di inserire un numeratore e un

denominatore, e quindi calcola la divisione di questi numeri. Il risultato viene quindi visualizzato sullo schermo. Se l'utente sceglie l'opzione *C*, viene chiamata la funzione *ins_string()*. Questa funzione richiede all'utente di inserire una stringa e poi la visualizza a schermo.

2. Il programma presenta le seguenti caratteristiche non standard che non vengono gestite:
    - Input non validi: il programma non controlla se gli input dell'utente sono validi. Ad esempio, un utente potrebbe inserire un numero non valido per una delle operazioni aritmetiche, come ad esempio una stringa o un numero negativo.
    - Divisione per zero: il programma non gestisce la divisione per zero. Se l'utente inserisce un denominatore pari a zero, il programma genera un errore indefinito.
    - Stringhe troppo lunghe: il programma non gestisce stringhe più lunghe di 10 caratteri. Infatti, se l'utente inserisce una stringa del genere, il programma mostrerà a schermo solo i primi 10 caratteri.

3. Il programma non presenta errori di sintassi, ma presenta i seguenti errori logici:
    - la funzione *moltiplica()* utilizza il tipo di dato *short int*, che è un tipo di dato a 16 bit. Questo significa che il risultato della moltiplicazione potrebbe essere arrotondato in modo non accurato se uno dei due numeri è maggiore di un certo valore.
    - la funzione *dividi()* utilizza il simbolo *%* per eseguire la divisione tra due numeri. Tuttavia questo simbolo viene utilizzato per il modulo, il quale restituisce il resto della divisione.

4. Per risolvere l'errore logico della funzione *moltiplica()*, è possibile utilizzare il tipo di dato *int* per le variabili *a*, *b* e *prodotto*, e alla riga 47

sostituire *%f* con *%d* (così indichiamo che il numero da acquisire è un intero) come nell'immagine seguente

```
43 void moltiplica ()
44 {
45         int  a,b = 0;
46         printf ("Inserisci i due numeri da moltiplicare:");
47         scanf ("%d", &a);
48         scanf ("%d", &b);
49 
50         int prodotto = a * b;
51 
52         printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
53 }
```

Per risolvere l'errore logico della funzione *dividi()*, è possibile utilizzare il simbolo */* come mostrato di seguito

```
62 void dividi ()
63 {
64         int  a,b = 0;
65         printf ("Inserisci il numeratore:");
66         scanf ("%d", &a);
67         printf ("Inserisci il denominatore:");
68         scanf ("%d", &b);
69 
70         int divisione = a / b;
71 
72         printf ("La divisione tra %d e %d e': %d", a,b,divisione);
73 }
```

Per risolvere i problemi relativi agli input non validi e alla divisione per zero, è possibile aggiungere delle verifiche all'interno delle funzioni *moltiplica()* e *dividi()* come nell'immagine seguente

```c
void moltiplica ()
{
        int  a,b = 0;
        printf ("Inserisci i due numeri da moltiplicare:");
        scanf ("%d", &a);
        scanf ("%d", &b);

        if (a < 0 || a > INT_MAX || b < 0 || b > INT_MAX)
        {
                printf ("Uno dei due numeri non è valido.\n");
                return;
        }

        int prodotto = a * b;

        printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
}


void dividi ()
{
        int  a,b = 0;
        printf ("Inserisci il numeratore:");
        scanf ("%d", &a);
        printf ("Inserisci il denominatore:");
        scanf ("%d", &b);

        if (b == 0)
        {
                printf ("Il denominatore non può essere zero.\n");
                return;
        }

        int divisione = a / b;

        printf ("La divisione tra %d e %d e': %d", a,b,divisione);
}
```

Per risolvere il problema del limite di 10 caratteri per le stringhe, è possibile utilizzare la funzione *fgets()* per leggere la stringa dall'utente. Questa funzione consente di leggere un numero arbitrario di caratteri dalla tastiera, fino al raggiungimento di un carattere di nuova riga o di un limite di caratteri specificato. La funzione *ins_string()* utilizza la funzione *fgets()* per leggere una stringa di qualsiasi lunghezza. In questo modo, il programma sarà in grado di gestire qualsiasi tipo di stringa.

```
84
85 void ins_string ()
86 {
87         char stringa[10];
88         printf ("Inserisci la stringa:");
89         fgets(stringa, 10, stdin);
90 }
91
```