Test seconda settimana Epicode

```
#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;
printf ("Inserisci i due numeri da moltiplicare:");
scanf ("%f", &a);
scanf ("%d", &b);

short 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 denumeratore:");
scanf ("%d", &b);
int divisione = a % b;
printf ("La divisione tra %d e %d e': %d", a,b,divisione);
}
void ins_string ()
{
char stringa[10];
```

printf ("Inserisci la stringa:");

scanf ("%s", &stringa);

}

Questo è il codice che dobbiamo esaminare, senza eseguirlo. A primo impatto, sembrerebbe un programma che, una volta eseguito, da l'opportunità all'utente di scegliere da un menù diverse operazioni da poter eseguire (scusate per la ripetizione).

Possiamo individuare infatti 3 operazioni: moltiplicazione, divisione e inserimenti di una stringa.

Il codice, però, presenta diversi errori sia di sintassi che logici. Proveremo quindi ad individuarli tutti e correggerli, affinché il programma sia correttamente eseguibile. Se possibile, inoltre, proveremo a capire se il programma è ottimizzabile per risparmiare memoria una volta che andremo ad eseguirlo.

Ho usato un elenco numerato, in modo da poter scrivere direttamente a che riga è presente un errore e spiegarlo, rendendo più facile l'individuazione della riga per chi leggerà successivamente questo documento.

Partiamo dagli errori di sintassi:

Alla riga 8 possiamo trovare il primo errore. Sono state inserite le parentesi graffe, che vengono usate per dichiarare blocchi di codice. In questo caso, non sono necessarie. La riga corretta sarà quindi char scelta = '\0';

andando avanti, possiamo notare che alla riga 10 possiamo notare che è stato utilizzato %d (che viene utilizzato per dichiarare numeri interi) mentre noi vogliamo che siano presenti caratteri, rendendo quindi errata questa stringa. Il codice giusto sarà scanf(" %c, &scelta) utilizzeremo anche lo spazio prima di "%" per evitare problemi di buffer.

Successivamente, alla riga 35 possiamo notate che per il primo numero è stato utilizzato %f che si usa solitamente per i numeri reali, mentre per il secondo è stato inserito %d, che si usa per i numeri interi. IN questo caso, a meno che l'utente non voglia esplicitamente un'operazione con possibili risultati di numeri decimali, punteremo a eseguire la moltiplicazione tra soli numeri interi, andando quindi a sostituire il primo %f con %d e facendo si che l'operazione avvenga sempre tra due numeri interi e che dia come risultato solo un numero intero (considerando inoltre che successivamente viene riportato come %d).

Sia per l'operazione di moltiplicazione di divisione che per quella di divisione, la riga presentata è int a,b = 0; la seconda parte, però, non è necessaria. Sarà sufficiente scrivere int a, b;

Alla riga 45, è stato scritto "denumeratore" ma il termine corretto è "denominatore";

Ancora, alla riga 47 viene riportata l'operazione di divisione che dovrà essere eseguita, ma l'operatore risulta essere errato, in quanto è riportato & al posto di /.

Come ultimo errore di sintassi, alla riga 54 è presente il comando scanf ("%s", &stringa); in questo caso, però, la & non è necessaria in quanto andremo a leggere un array di caratteri. Ricordiamo che la & solitamente

viene usata per ottenere l'indirizzo di memoria di una variabile. Essendo però presente il comando "scanf" non è necessario utilizzare la &.

Passiamo adesso agli errori logici:

Nell'operazione di divisione, il secondo operatore che l'utente può inserire potrebbe essere 0, che porterebbe ad un errore in quanto non è possibile dividere per 0. Andremo quindi ad usare un "if" "else if" per evitare che questo accada. Il codice, una volta corretto, diventerà:

```
if (b != 0) {
   int divisione = a / b;
   printf("La divisione tra %d e %d e': %d\n", a, b, divisione);
} else {
   printf("Errore: Divisione per zero non consentita.\n");
}
```

Si possono inserire lettere come operatori delle moltiplicazioni e divisioni. Per risolvere il problema e far comparire un messaggio che ci dica che l'input è errato, possiamo utilizzare le funzioni WHILE e IF in questo modo (riporto esempio solo di moltiplicazione):

```
while (1) {
      printf("Inserisci i due numeri da moltiplicare: ");
      if (scanf("%d %d", &a, &b) == 2) {
         int prodotto = a * b;
         printf("Il prodotto tra %d e %d e': %d\n", a, b, prodotto);
         break;
      } else {
         printf("Input non valido. Inserisci due numeri validi.\n");
         scanf("%s", input);
```

Proseguendo, possiamo notare che nella sezione per l'inserimento di una stringa, non è riportato un numero massimo di caratteri da poter inserire all'interno della stessa, il che potrebbe portare ad eventuali errori. Potremo risolvere così il problema:

```
void ins_string(char *stringa, int dimensione) {
```

```
printf("Inserisci la stringa: ");
scanf("%*c");

if (fgets(stringa, dimensione, stdin) != NULL) {
    size_t len = strlen(stringa);
    if (len > 0 && stringa[len - 1] == '\n') {
        stringa[len - 1] = '\0';
    } else {
        int c;
        while ((c = getchar()) != '\n' && c != EOF); questa parte di codice, serve a eliminare eventuali caratteri residui o "spazzatura" nel buffer di input in modo che il prossimo input possa essere letto correttamente.
    }

    if (len > dimensione) { questa riga verifica che la dimensione della stringa non superi il massimo consentito
        stringa[dimensione - 1] = '\0'; nel caso lo sia, questa parte di codice tronca la parte di stringa eccessiva, riportandola al massimo consentito
        printf("La stringa inserita è troppo lunga e verrà troncata
```

Questo blocco di codice utilizza diversi comandi e variabili. Il tutto ci permette di inserire al massimo 100 caratteri senza che il programma crashi o si verifichino bug.

Nello specifico, ci sono diversi comandi che saltano sicuramente all'occhio:

`if (fgets(stringa, dimensione, stdin) != NULL)` utilizza fgets per leggere una linea di testo. Se fgets non risucirà a leggere la linea con successo, apparirà un messaggio di errore.

`size_t len = strlen(stringa);` calcola quanto è lunga la stringa utilizzando strlen

`if (len > 0 && stringa[len - 1] == '\n') {`

`stringa[len - 1] = '\0';`

questa seconda parte, invece, sostituisce il carattere di nuova riga con uno che invece termina la riga, poiché l'utente potrebbe non volere che lo stesso venga rappresentato nel messaggio finale. Assicura inoltre che la dimensione della stringa sia maggiore di 0

Facendo le operazioni di moltiplicazione e divisione, se inseriamo un carattere non numerico il programma continua in modo errato. È possibile correggere questa cosa e mostrarci un messaggio di errore nel caso in cui si inserisca un carattere non valido.

Possiamo correggere altre righe, più per ottimizzare che per andare a correggere effettivamente errori.

Ai risultati di moltiplicazione e divisione possiamo aggiungere \n dopo il risultato.

Per la moltiplicazione, il messaggio che porta al risultato potrebbe essere “il risultato della moltiplicazione è: prodotto” invece di “il risultato del prodotto è: prodotto”. Questo renderà più chiaro il passaggio.

Viene usato int short, che prevede un database minore di numeri per le operazioni. Lo lasceremo così, ma se fosse necessario potremmo inserire int, che prevede una scelta più ampia di operatori.

Possiamo inoltre notare che, una volta terminate una delle operazioni, il programma termina in modo automatico. Possiamo aggiungere un comando che ci faccia scegliere se fare altro o uscire dal programma.

Possiamo anche notare che le lettere minuscole non sono contemplate nel codice, possiamo ottimizzarlo in modo da accettare anche quelle e non solo le maiuscole usando un IF in questo modo if (scelta == 'A' || scelta == 'a') {

    moltiplica();

Non so se sono riuscito a trovare tutti gli errori e spero di non essermi dilungato troppo, ma volevo essere più chiaro possibile nelle spiegazioni per rendere tutto più comprensibile. Il programma, alla fine delle modifiche, sembra funzionare e non crasha in alcun modo. Ovviamente, non essendo esperto di C, ho passato la giornata online per capire le soluzioni ai vari problemi che ho trovato. Meglio essere chiari. Lascio il codice finale:

```
#include <stdio.h>

#include <string.h>


void menu();

void moltiplica();

void dividi();

void ins_string();


int main() {

char scelta;


do {

menu();

scanf(" %c", &scelta);


if (scelta == 'A' || scelta == 'a') {

moltiplica();

} else if (scelta == 'B' || scelta == 'b') {

dividi();
```

```
} else if (scelta == 'C' || scelta == 'c') {
ins_string();
} else if (scelta == 'E' || scelta == 'e') {
printf("Arrivederci!\n");
return 0;
} else {
printf("Scelta non valida.\n");
}

printf("Vuoi fare un'altra operazione? (S per Sì, qualsiasi altro tasto per uscire): ");
scanf(" %c", &scelta);
} while (scelta == 'S' || scelta == 's');

return 0;
}

void menu() {
printf("Benvenuto, sono un assistente 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");
printf("E >> Esci\n");
}

void moltiplica() {
int a, b;
char input[10];

while (1) {
printf("Inserisci i due numeri da moltiplicare: ");
if (scanf("%d %d", &a, &b) == 2) {
int prodotto = a * b;
printf("Il prodotto tra %d e %d e': %d\n", a, b, prodotto);
break;
} else {
```

```
printf("Input non valido. Inserisci due numeri validi.\n");
scanf("%s", input);
}
}
}

void dividi() {
int a, b;
char input[10];

while (1) {
printf("Inserisci il numeratore: ");
if (scanf("%d", &a) == 1) {
printf("Inserisci il denominatore diverso da zero: ");
if (scanf("%d", &b) == 1 && b != 0) {
int divisione = a / b;
printf("La divisione tra %d e %d e': %d\n", a, b, divisione);
break;
} else {
printf("Input non valido. Inserisci un denominatore valido diverso da zero.\n");
scanf("%s", input);
}
} else {
printf("Input non valido. Inserisci un numeratore valido.\n");
scanf("%s", input);
}
}
}

void ins_string() {
char stringa[100];

printf("Inserisci la stringa: ");
scanf("%*c");
```

```
if (fgets(stringa, sizeof(stringa), stdin) != NULL) {
size_t len = strlen(stringa);
if (len > 0 && stringa[len - 1] == '\n') {
stringa[len - 1] = '\0';
} else {
int c;
while ((c = getchar()) != '\n' && c != EOF);
}

printf("Hai inserito: %s\n", stringa);
} else {
printf("Errore durante la lettura dell'input.\n");
}
}
```