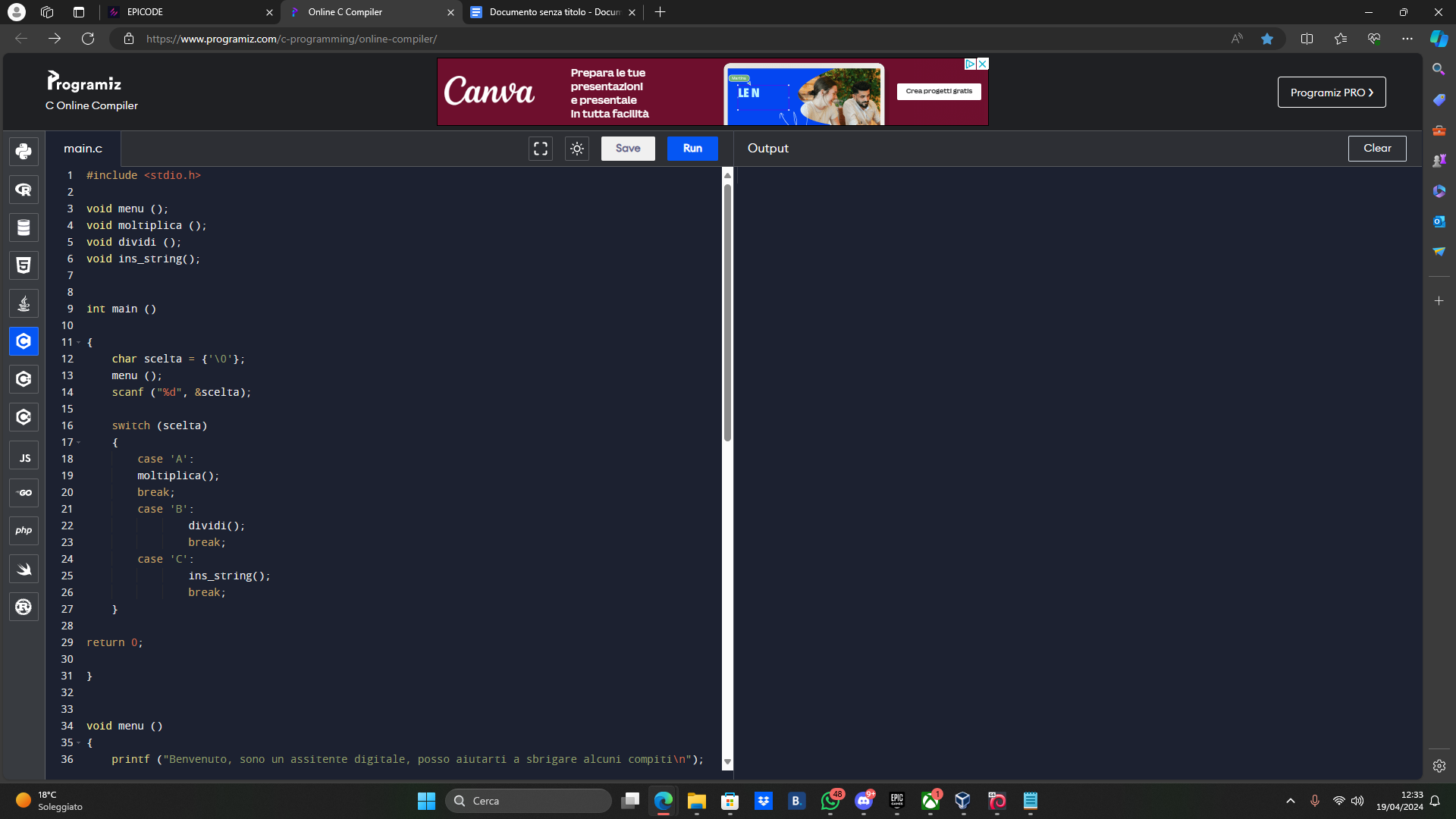
S2L5 BUG HUNTING

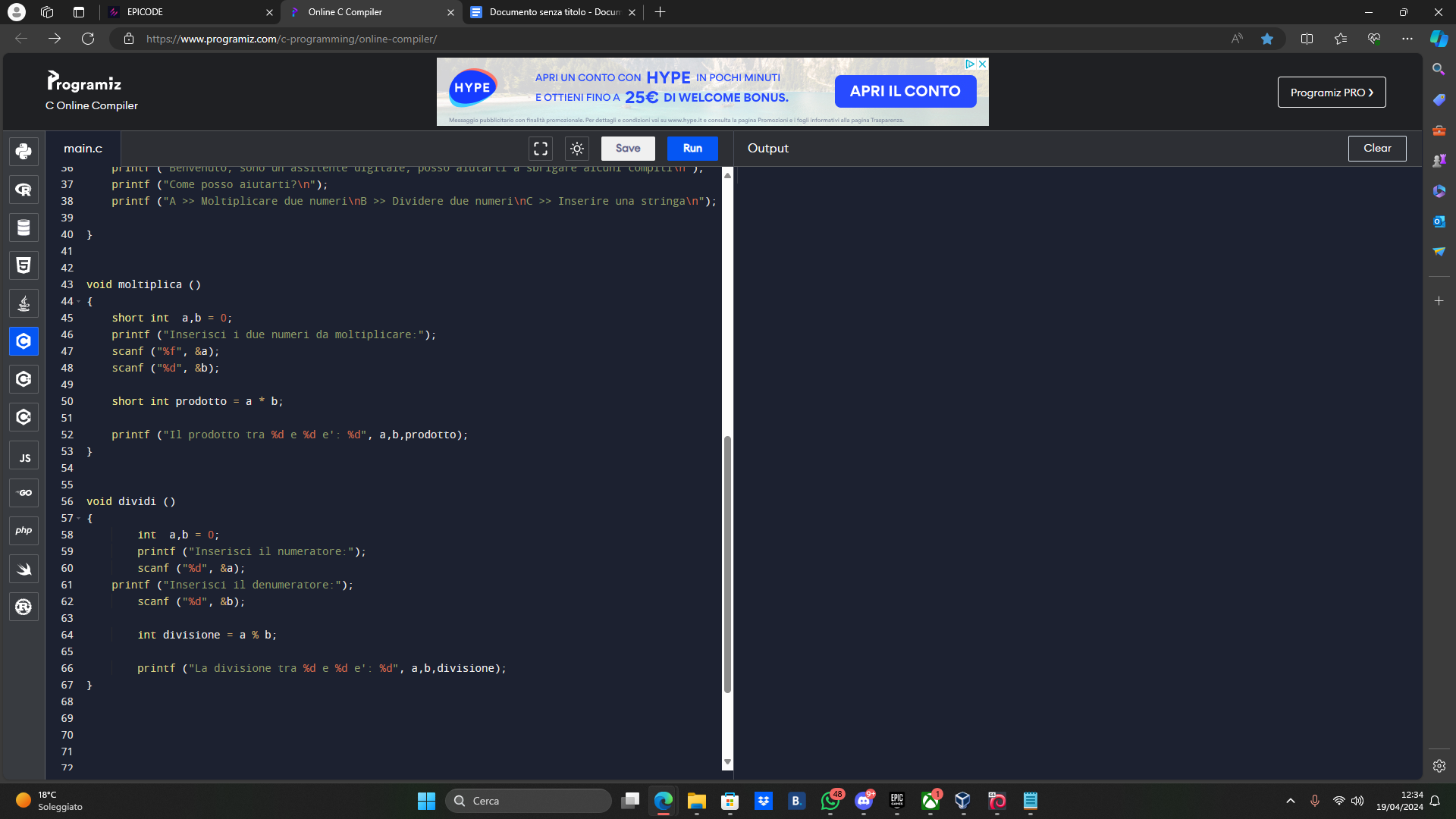
L'arte dell'hacking è spesso associata alla sua connotazione negativa, ma al suo cuore risiede un'abilità fondamentale: il pensiero creativo e fuori dagli schemi. Affrontare un problema di sicurezza informatica richiede molto più di una conoscenza tecnica approfondita; richiede una mente che possa vedere oltre ciò che è evidente, individuando le debolezze nascoste e immaginando scenari non convenzionali.

Nell'esercizio di oggi, ci troveremo di fronte a un compito intrigante: analizzare un codice sorgente con occhi critici e identificare non solo ciò che fa, ma anche ciò che potrebbe non fare o gestire in modo efficace. Questo esercizio non solo ci sfiderà a comprendere il funzionamento del codice, ma ci porterà anche a individuare potenziali lacune e a proporre soluzioni innovative.

Prepariamoci quindi a immergerci in un viaggio attraverso il codice, esplorando le sue sfaccettature e mettendo alla prova la nostra capacità di pensare al di là delle convenzioni. Siamo pronti ad affrontare questa sfida e a sviluppare le nostre abilità nel mondo sempre mutevole della sicurezza informatica.

vediamo il codice fornito:





A questo punto possiamo notare che nel codice(in questo caso programmato in linguaggio di programmazione C) sono presenti vari errori su quali ora faremo piccoli focus per capire gli errori e possibili soluzioni.

Come primo punto capiamo il codice a cosa servirebbe.

Questo codice costituisce la base di un assistente digitale progettato per facilitare diverse operazioni all'utente, dal calcolo matematico alla manipolazione delle stringhe. L'obiettivo principale è offrire un'interfaccia intuitiva e semplice per eseguire tali operazioni senza la necessità di complessi strumenti software.

L'assistente digitale si presenta con un menu chiaro e conciso, consentendo all'utente di selezionare facilmente l'azione desiderata tra le opzioni disponibili: moltiplicare due numeri, dividere due numeri o inserire una stringa. Questo approccio mira a semplificare il processo decisionale dell'utente e a fornire una rapida risposta alle sue esigenze.

Al di là delle funzionalità immediate, il codice illustra anche l'importanza della gestione corretta degli input utente e della manipolazione dei dati all'interno del programma. Questi aspetti sono cruciali per garantire un'esperienza utente senza problemi e per evitare errori di esecuzione o comportamenti imprevisti.

In breve, il codice fornisce una base solida per lo sviluppo di un assistente digitale versatile e pratico, che mira a semplificare le attività quotidiane dell'utente attraverso un'interazione intuitiva e diretta.

ERRORI DI CODICE:

1) Acquisizione della Scelta dell'Utente: La variabile scelta è dichiarata come un carattere (char), ma viene acquisita come un intero (%d). Questo provoca un comportamento imprevisto perché %d aspetta un intero mentre %c aspetta un carattere.

2) Input Errato per la Moltiplicazione: Nella funzione moltiplica(), le variabili a e b sono dichiarate come short int, ma vengono acquisite come float e int rispettivamente. Questo potrebbe causare errori di precisione o di tipo.

3) Operazione Errata nella Divisione: Nella funzione dividi(), l'operatore % viene utilizzato per la divisione anziché l'operatore /, causando il calcolo del resto anziché del risultato della divisione.

4) Acquisizione della Stringa: La funzione ins\_string() acquisisce una stringa dall'utente, ma utilizza l'operatore di indirizzamento (&) per la variabile stringa, che non è necessario per un array di caratteri.

5) Manca il Caso di Default nel switch: Non è presente un caso di default nel switch, il che significa che se l'utente inserisce un valore diverso da 'A', 'B', o 'C', il programma non gestirà correttamente questa situazione e potrebbe avere un comportamento imprevisto.

6) Nella riga 26, l'utilizzo dell'operatore di indirizzamento "&" con la variabile "scelta" all'interno della funzione "scanf" è errato. L'operatore "&" è utilizzato per ottenere l'indirizzo di memoria di una variabile, ma nel caso di "scanf" per acquisire un singolo carattere, non è necessario.

CORREZIONE/**SPIEGAZIONE/ RIGHE DI CODICE/TIPOLOLGIA**

**Errore 1: 12. SINTASSI**

char scelta = { ‘ \0 ‘ } ;

scanf (“%d” , &scelta);

**correzione 1**:

char scelta = {'\0'};

scanf (" %c", &scelta); // Corretto: %c acquisisce un singolo carattere correttamente

**Utilizzando %d nella funzione scanf, si attende l'acquisizione di un intero anziché un carattere. Correggendo con %c, si acquisisce un singolo carattere come desiderato.**

**errore 2: 28/29. SINTASSI**

scanf ("%f", &a); // Sbagliato: %f acquisisce un float anziché un short int

scanf ("%d", &b); // Sbagliato: %d acquisisce un intero anziché un short int

**correzione 2:**

scanf ("%hd", &a); // Corretto: %hd acquisisce un short int per il primo numero

scanf ("%hd", &b); // Corretto: %hd acquisisce un short int per il secondo numero

**Utilizzando %f e %d, si acquisiscono rispettivamente un float e un intero, mentre le variabili a e b sono dichiarate come short int. Correggendo con %hd, si acquisiscono short int come previsto.**

**errore 3: 45. LOGICA**

int divisione = a % b; // Sbagliato: l'operatore % calcola il resto anziché la divisione

**correzione 3:**

int divisione = a / b; // Corretto: l'operatore / calcola il risultato della divisione

**Utilizzando l'operatore %, si calcola il resto della divisione anziché il risultato. Correggendo con l'operatore /, si ottiene il risultato corretto della divisione.**

**errore 4: 62. SINTASSI**

scanf ("%s", &stringa); // Sbagliato: &stringa non è necessario per un array di caratteri

**correzione 4:**

scanf ("%s", stringa); // Corretto: acquisisce una stringa senza l'operatore di indirizzamento &

**Per acquisire una stringa in un array di caratteri, non è necessario utilizzare l'operatore di indirizzamento (&) poiché l'array stesso rappresenta già l'indirizzo di memoria del primo elemento.**

**errore 5: 18./19./20. LOGICA**

// Caso di default mancante nel switch

**correzione 5:**

default:

printf("Scelta non valida.\n");

break;

**aggiungere un caso di default nel switch permette di gestire correttamente le scelte non previste dall'utente, fornendo un feedback appropriato nel caso in cui venga inserita un'opzione non valida.**

**ERRORE 6: 26. SINTASSI**

**scanf ("%s", &scelta); // Sbagliato: &scelta non è necessario per acquisire un carattere**

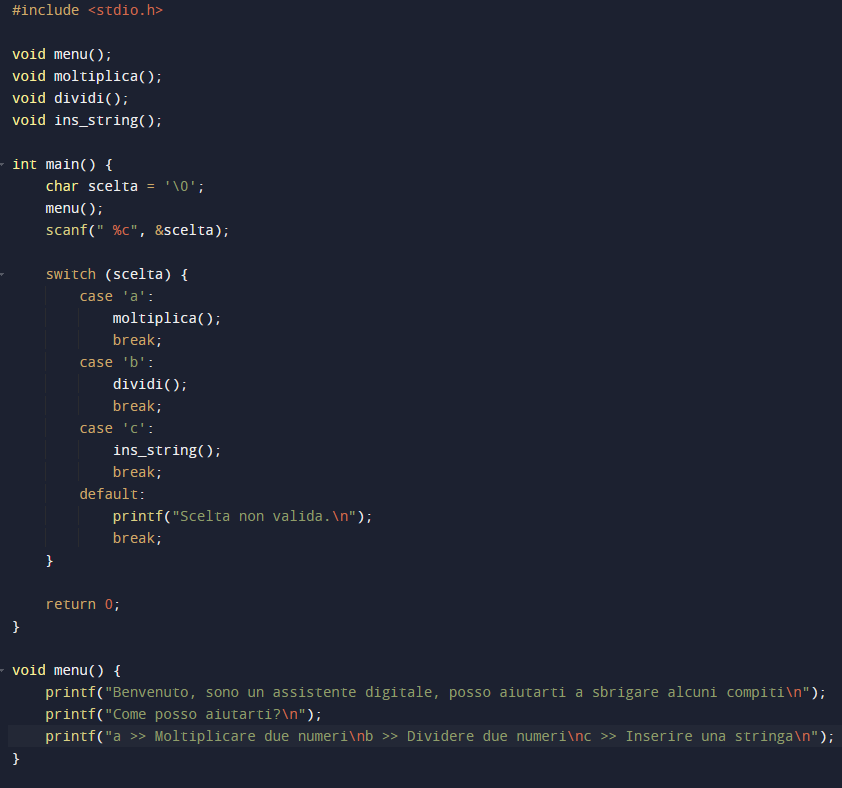
**CORREZIONE:**

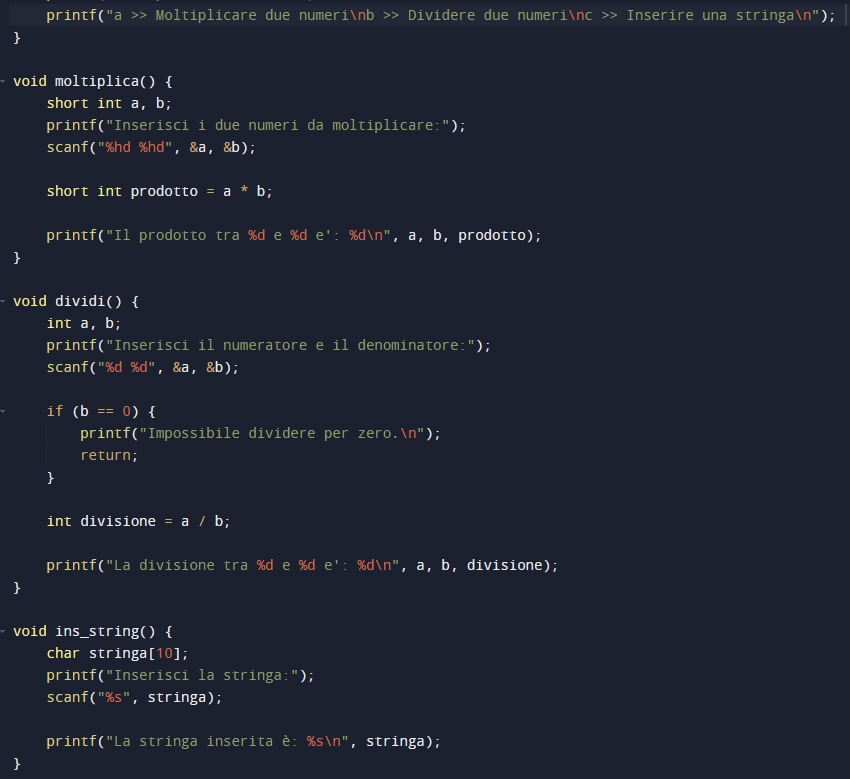
**26. scanf (" %c", &scelta); // Corretto: acquisisce un singolo carattere senza l'operatore di indirizzamento &**

**L'errore è di sintassi perché l'operatore di indirizzamento & non è necessario per acquisire un singolo carattere in una variabile di tipo char. L'operatore & è utilizzato per ottenere l'indirizzo di memoria di una variabile, ma nel caso di scanf per acquisire un carattere singolo, non è richiesto. Utilizzando scanf(" %c", &scelta);, si acquisisce correttamente un singolo carattere senza l'operatore di indirizzamento &.**

**\*CODICE CORRETTO PAGINE SEGUENTI**

**CODICE SENZA BUG**





MESSO SU KALI

