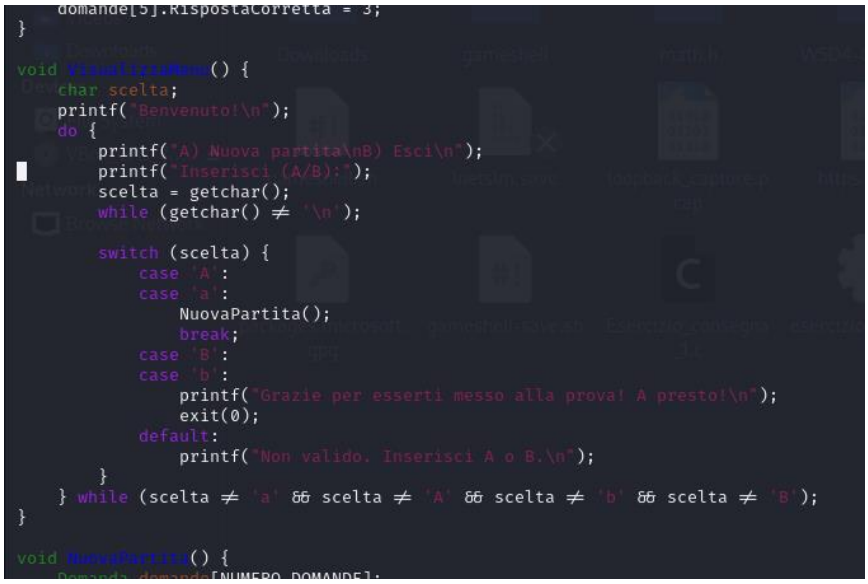
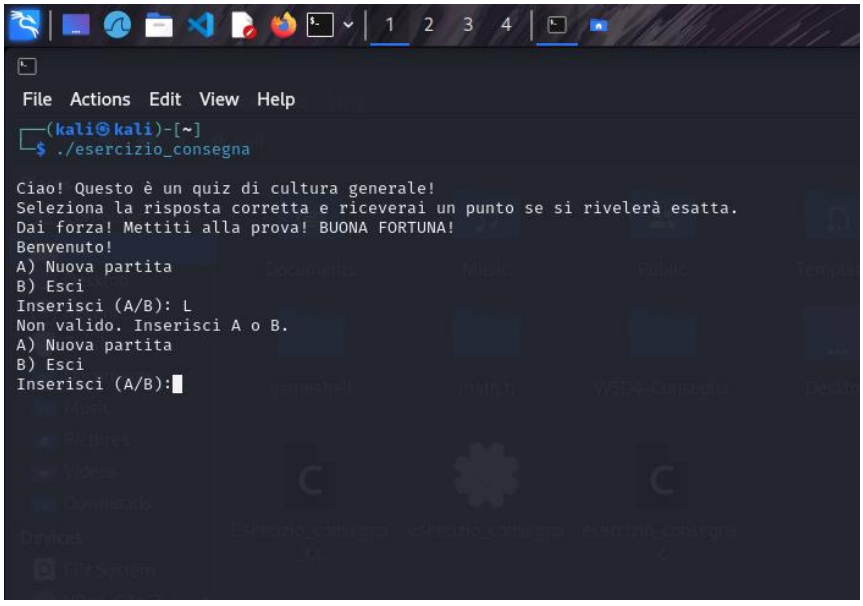
1 - Per quanto riguarda la problematica che si dovesse creare se l'utente inserisse una lettera diversa da 'A', 'a', 'B' o 'b' nel menu principale all'avvio del programma, io avevo già programmato il codice per far riconoscere la scelta come non valida. Per fare ciò avevo inserito il costrutto 'switch' nella funzione 'VisualizzaMenu'. Infatti, grazie a ciò, se l'input del giocatore non corrisponde a quanto scritto prima, il programma stamperà a schermo un messaggio con scritto "Non valido. Inserisci A o B.\n" e chiederà di scegliere di nuovo. Con il 'do' e successivamente il 'while', sempre nella funzione 'VisualizzaMenu', si instaurerà un ciclo che continuerà a richiedere all'utente di inserire un input fino a quando la scelta non sarà valida. Ciò assicura che il giocatore sia sempre in grado di iniziare una nuova partita o di uscire dal gioco, evitando che il programma proceda la sua esecuzione, almeno fino a quando l'input non sarà quello esatto.



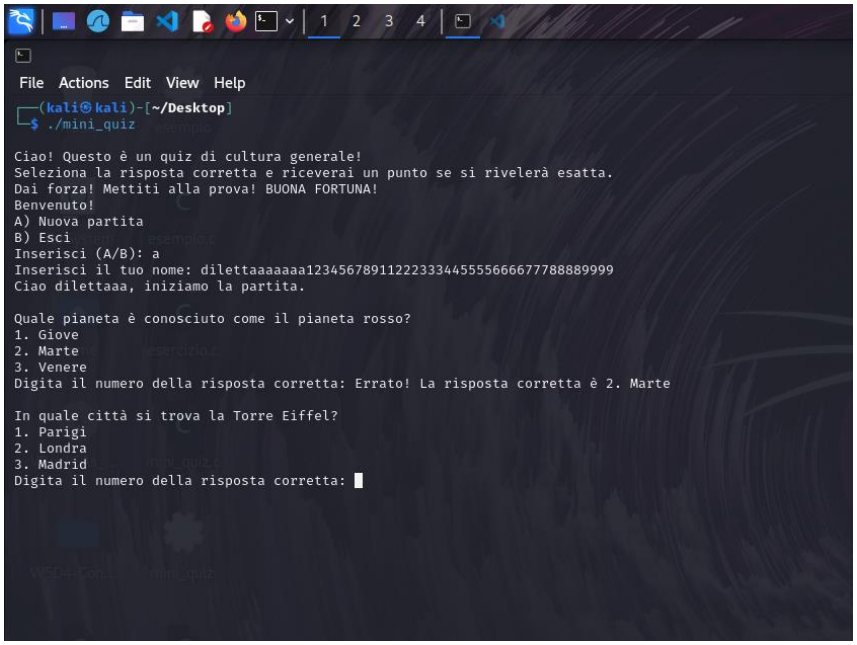
```
    domande[5].RispostaCorretta = 3;
}

void VisualizzaMenu() {
    char scelta;
    printf("Benvenuto!\n");
    do {
        printf("A) Nuova partita\nB) Esci\n");
        printf("Inserisci (A/B):");
        scelta = getchar();
        while (getchar() != '\n');

        switch (scelta) {
            case 'A':
            case 'a':
                NuovaPartita();
                break;
            case 'B':
            case 'b':
                printf("Grazie per esserti messo alla prova! A presto!\n");
                exit(0);
            default:
                printf("Non valido. Inserisci A o B.\n");
        }
    } while (scelta != 'a' && scelta != 'A' && scelta != 'b' && scelta != 'B');
}

void NuovaPartita() {
    Domanda domande[NUMERO_DOMANDE];
```



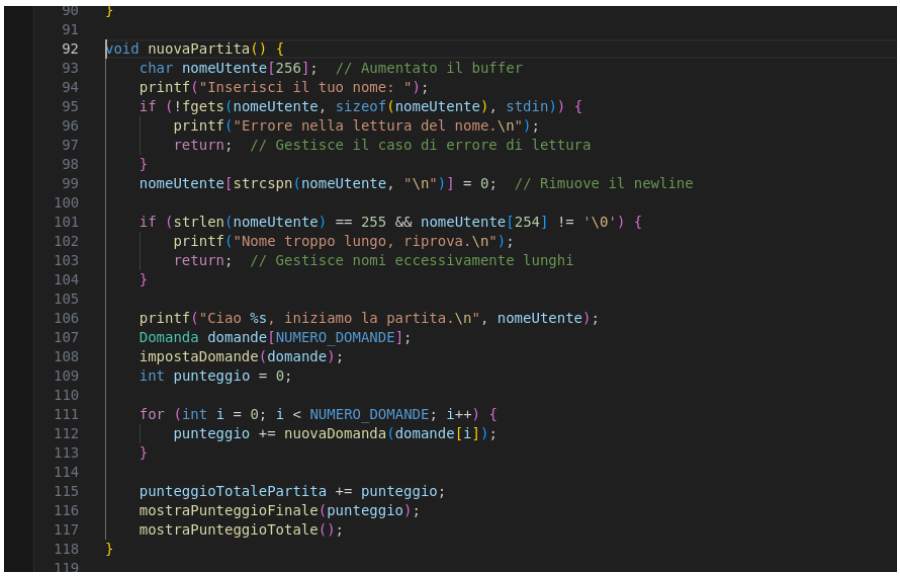
```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~]
└─$ ./esercizio_consegna

Ciao! Questo è un quiz di cultura generale!
Seleziona la risposta corretta e riceverai un punto se si rivelerà esatta.
Dai forza! Mettiti alla prova! BUONA FORTUNA!
Benvenuto!
A) Nuova partita
B) Esci
Inserisci (A/B): L
Non valido. Inserisci A o B.
A) Nuova partita
B) Esci
Inserisci (A/B):
```

2 - Se l'utente inserisce un nome utente contenente più di 10 caratteri, grazie alla funzione 'fgets' si utilizzeranno sono i primi 9 caratteri per il nome e il decimo carattere sarà '\0' e chiuderà la stringa. Questo perché 'fgets' legge, in questo specifico caso, massimo 9 caratteri, quindi uno in meno rispetto alla dimensione specificata e come ultimo inserisce la terminazione '\0'. Quindi, quando il nome è più lungo di 9 caratteri, quelli rimanenti resteranno nel buffer 'stdin' fino alla prossima funzione di input e ciò interferirà con future letture del buffer.

Nell'immagine, infatti, si vede che l'immissione del nome utente molto lungo ha interferito con la successiva domanda, selezionandomi una risposta non corretta e skippando direttamente alla domanda successiva.



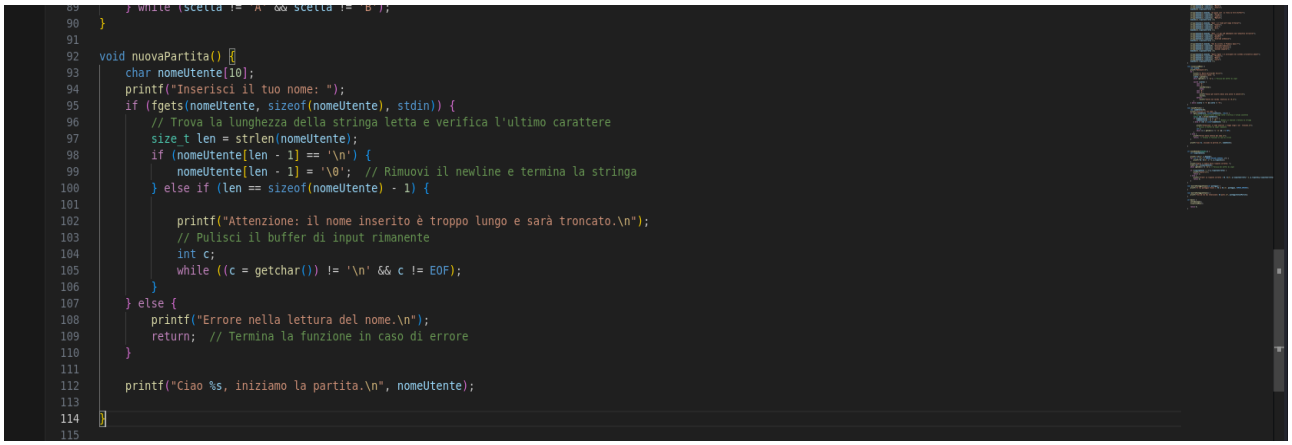
Per risolvere questo problema si può semplicemente inserire un array con una dimensione più grande, come nell'immagine sottostante:



```
}

void nuovaPartita() {
    char nomeUtente[256];  // Aumentato il buffer
    printf("Inserisci il tuo nome: ");
    if (!fgets(nomeUtente, sizeof(nomeUtente), stdin)) {
        printf("Errore nella lettura del nome.\n");
        return;  // Gestisce il caso di errore di lettura
    }
    nomeUtente[strcspn(nomeUtente, "\n")] = 0;  // Rimuove il newline

    if (strlen(nomeUtente) == 255 && nomeUtente[254] != '\0') {
        printf("Nome troppo lungo, riprova.\n");
        return;  // Gestisce nomi eccessivamente lunghi
    }

    printf("Ciao %s, iniziamo la partita.\n", nomeUtente);
    Domanda domande[NUMERO_DOMANDE];
    impostaDomande(domande);
    int punteggio = 0;

    for (int i = 0; i < NUMERO_DOMANDE; i++) {
        punteggio += nuovaDomanda(domande[i]);
    }

    punteggioTotalePartita += punteggio;
    mostraPunteggioFinale(punteggio);
    mostraPunteggioTotale();
}
```

Oppure, una ulteriore soluzione, potrebbe essere questa nell'immagine sottostante.

Come nel caso precedente, se l'input dell'utente supera i 10 caratteri, 'fgets' leggerà i primi 9 caratteri e inserirà '\0' alla fine. Per ripulire il buffer e, quindi per evitare eventuali caratteri rimasti che possono interferire con future chiamate, si esegue un ciclo che legge e scarta tutti i caratteri fino a raggiungere '\n' o 'EOF', quindi la fine del file.



```
    } while (scelta != 'A' && scelta != 'B');
}

void nuovaPartita() {
    char nomeUtente[10];
    printf("Inserisci il tuo nome: ");
    if (fgets(nomeUtente, sizeof(nomeUtente), stdin)) {
        // Trova la lunghezza della stringa letta e verifica l'ultimo carattere
        size_t len = strlen(nomeUtente);
        if (nomeUtente[len - 1] == '\n') {
            nomeUtente[len - 1] = '\0';  // Rimuovi il newline e termina la stringa
        } else if (len == sizeof(nomeUtente) - 1) {

            printf("Attenzione: il nome inserito è troppo lungo e sarà troncato.\n");
            // Pulisci il buffer di input rimanente
            int c;
            while ((c = getchar()) != '\n' && c != EOF);
        }
    } else {
        printf("Errore nella lettura del nome.\n");
        return;  // Termina la funzione in caso di errore
    }

    printf("Ciao %s, iniziamo la partita.\n", nomeUtente);

}
```

3 - Per quanto riguarda il comportamento del codice qualora l'utente inserisse un carattere alfanumerico casuale invece di quello associato ad una risposta, nel mio caso avevo attribuito ad ogni risposta un valore numerico e non una lettera, per procedere alla scelta.

La funzione 'scanf()' da me inserita, con lo specificatore '%d', non riesce a leggere un carattere che non sia numerico tra quelli assegnati alle opzioni di risposta e quindi verificarlo. In questo caso l'input che non è valido rimane nel buffer di input e potrebbe eventualmente causare problemi nei cicli successivi o nelle successive 'scanf' che sono presenti.

Nel codice iniziale avevo già inserito una verifica della risposta, in cui il programma confrontava l'indice della risposta dall'utente (-1 per adattarmi all'indice con base zero dell'array) con l'indice della risposta corretta scritto in 'q.rispostaCorretta'. Nel caso in cui la risposta fosse corretta viene stampato a schermo il messaggio di conferma e si dà 1, in caso contrario ci sarà un messaggio di errore con scritta la risposta corretta e si dà 0.

Si constata, quindi, che in qualsiasi caso il codice darà indietro il messaggio di errore, anche se dovessi scrivere una lettera o una serie di numeri o lettere che non siano quelle definite in precedenza.

```
In quale città si trova la Torre Eiffel?
1. Parigi
2. Londra
3. Madrid
Digita il numero della risposta corretta: 36
Errato! La risposta corretta è 1. Parigi

Qual è il fiume più lungo d'Italia?
1. Tevere
2. Arno
3. Po
Digita il numero della risposta corretta: ciao
Errato! La risposta corretta è 3. Po

Qual è il gas più abbondante nell'atmosfera terrestre?
1. Ossigeno
2. Azoto
3. Anidride carbonica
Digita il numero della risposta corretta: diletta123
Errato! La risposta corretta è 2. Azoto
```

Una eventuale soluzione potrebbe essere quella di utilizzare un loop per richiedere l'input all'utente ripetitivamente fino a quando non viene inserito un valore valido:

```c
111
112 int rispostaUtente;
113 int controlloInput;
114
115 do {
116     printf("Digita il numero della risposta corretta: ");
117     controlloInput = scanf("%d", &rispostaUtente);
118     while (getchar() != '\n'); // Pulizia del buffer di input
119
120     if (controlloInput != 1 || rispostaUtente < 1 || rispostaUtente > NUMERO_OPZIONI_DOMANDA) {
121         printf("Inserisci un numero valido tra 1 e %d.\n", NUMERO_OPZIONI_DOMANDA);
122         rispostaUtente = -1; // Resetta il valore di rispostaUtente per sicurezza
123     }
124 } while (rispostaUtente == -1);
```

Oppure si può eliminare ‘scanf’, che appunto causa problemi di overflow, e utilizzare ‘fgets’, che invece legge l’intera linea fino ad un limite ben specificato (fino a ‘\n’). In questo caso ‘sscanf’ (molto più sicuro rispetto a ‘scanf’ e che riduce praticamente a zero i problemi di overflow) estrae un numero dall’input, dato da una stringa con lunghezza certa, e il ciclo controlla subito se il numero si trova in un range valido, fornendo subito a schermo un messaggio immediato all’utente e dà successivamente la possibilità di correggere l’input.

```c
}

int nuovaDomanda(Domanda q) {
    char inputLine[100];
    int rispostaUtente, risultatoScan;

    printf("\n%s\n", q.domanda);
    for (int i = 0; i < NUMERO_OPZIONI_DOMANDA; i++) {
        printf("%d. %s\n", i + 1, q.risposte[i]);
    }

    do {
        printf("Digita il numero della risposta corretta: ");
        if (!fgets(inputLine, sizeof(inputLine), stdin)) {
            printf("Errore di lettura. Riprova.\n");
            continue;
        }
        risultatoScan = sscanf(inputLine, "%d", &rispostaUtente);

        if (risultatoScan != 1 || rispostaUtente < 1 || rispostaUtente > NUMERO_OPZIONI_DOMANDA) {
            printf("Per favore inserisci un numero valido tra 1 e %d.\n", NUMERO_OPZIONI_DOMANDA);
        }
    } while (risultatoScan != 1 || rispostaUtente < 1 || rispostaUtente > NUMERO_OPZIONI_DOMANDA);

    if (rispostaUtente - 1 == q.rispostaCorretta) {
        printf("Esatto!\n");
        return 1;
    } else {
        printf("Errato! La risposta corretta è %d. %s\n", q.rispostaCorretta + 1, q.risposte[q.rispostaCorretta]);
        return 0;
    }
}

void mostraPunteggioFinale(int punteggio) {
    printf("Il tuo punteggio finale è: %d su %d.\n", punteggio, NUMERO_DOMANDE);
}

void mostraPunteggioTotale() {
    printf("Fino ad ora hai totalizzato: %d punti.\n", punteggioTotalePartita);
```