Domino Lineare:

Introduzione:

Il progetto è stato affrontato in modo individuale con l'obiettivo di implementare il gioco del dominio lineare, comprendente funzionalità di gioco sia per l'utente umano che per l'intelligenza artificiale (AI). Nel seguente documento, saranno descritte la struttura del codice, l'organizzazione del lavoro e le difficoltà incontrate durante lo sviluppo.

Struttura:

Il domino lineare che ho realizzato è costituito da 3 funzioni principali, il main in cui vengono eseguite tutte le funzioni, il gameplay senza ai e quello con ai, oltre a queste principali ne ho affiancate altre per evitare la ridondanza del codice e per garantire maggiore pulizia all'interno. Un esempio di implementazione è fornito dalla funzione **popolamento()** riportata di seguito:

```
void popolamento(int (*a)[n], int num_tessere) {
    for (int i = 0; i < num_tessere; i++) {
        for (int j = 0; j < n; j++) {
            a[i][j] = (rand() % 6) + 1;
        }
    }
}
```

che riceve in input il puntatore all'array bidimensionale "a" e la grandezza di questo array, in modo da poterlo popolare nella posizione a[i][0] e a[i][1], fino a quando i non raggiunge la grandezza dell'array. "a" verrà popolato secondo la regola imposta dalla consegna, con numeri casuali da 1 a 6, attraverso la funzione rand()%6 che restituisce un numero casuale da 0 a 5, a cui ho aggiunto 1 per farlo diventare da 1 a 6, questa funzione viene sempre utilizzata all'inizio del programma per popolare casualmente l'array fornito in input.

Organizzazione:

Il lavoro è stato svolto in modo autonomo, non ho dovuto gestire più persone per lavorare a questo progetto e quindi mi sono organizzato svolgendolo per punti:

- bozza del main
    - In cui ho creato un esoscheletro della funzione main in modo da poter capire quali fossero le prime funzioni da implementare, per poi andarlo a completare dopo la realizzazione delle funzioni principali.
- realizzazione delle prime funzioni quali:
    - void popolamento(int (*a)[n], int num_tessere):
        - popola l'array della lunghezza data.

    - void stampaArray(int (*a)[n], int num_tessere) e void stamp(int (*a)[n], int pos):
        - due funzioni simili che però svolgono lavori differenti:
            - la prima stampa tutto l'array dato in base alla lunghezza sempre fornita in input.
            - la seconda stampa l'array fino alla posizione data.

    - void ruota_tessera(int (*a)[n], int pos):
        - utilizzata nel caso in cui la tessera che vogliamo posizionare debba essere ruotata.
    - void inizializzaArrayGame(int (*game)[n], int num_tessere):
        - che inizializza l'array game in modo che contenga tessere con soli 0, utilizzata nella funzione che sviluppa il gameplay con l'ai.
- realizzazione delle funzioni principali:
    - void gioco(int (*a)[n], int num_tessere):
        - crea un nuovo array bidimensionale che inizialmente sarà vuoto, verrà poi popolato manualmente dal giocatore, prima selezionando la tessera da cui iniziare, attraverso l'inserimento da tastiera della posizione che ha la tessera selezionata nell'array. Successivamente verrà popolato tutto l'array finché si possono posizionare una affianco all'altra tessere con numeri combacianti uguali, attraverso delle determinate sequenze di lettere e numeri, come per esempio: p 1 r, che significa che la tessera nella posizione 1 verrà posizionata al primo posto nel nuovo array creato nella funzione e tutti gli altri elementi verranno spostati di uno a destra in modo da farle spazio, inoltre la tessera verrà ruotata. Questa funzione stamperà poi a video il punteggio ottenuto sommando i numeri all'interno delle tessere, assieme all'array game contenente le tessere che hanno portato a quel risultato.
    - void giocaAI(int (*a)[n], int num_tessere):
        - intelligenza artificiale che risolve il domino secondo la strategia del first match, ovvero la prima combinazione che trova viene utilizzata, questa funzione stampa tutte le mosse che vengono fatte per poi stampare alla fine tutto l'array game(creato sempre dentro la funzione) con all'interno il maggior numero di tessere, stamperà poi a video il punteggio ottenuto sommando i numeri all'interno delle tessere.

in questo modo sono riuscito a svolgere un lavoro organizzato e in modo efficiente, ottimizzando anche il tempo che ho impiegato per svolgerlo

Difficoltà riscontrate:

Le principali difficoltà riscontrate sono principalmente nella parte della creazione dell'algoritmo per l'ai, per la quale ho impiegato maggior tempo in quando ho cambiato la mia strategia dopo averne già iniziata una visto che era molto pesante e non funzionava

correttamente. Dopo aver cambiato strategia il tutto mi è risultato più semplice, e sono riuscito ad ottimizzare il mio programma.