UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

Gioco dell'oca

Versione documento 1.0

del

12.07.2023

GRUPPO 15
AUTORI

Andrisano Sofia
Cannone Giuseppe
Falcone Luca
Germinario Alessandro
Lacedra Nicolò

Esame
Laboratorio di Informatica corso A-L

INDICE

Preambolo

La presente documentazione del progetto del gioco dell'oca è stata redatta in seguito a una revisione del progetto originale, al fine di migliorare la gestione complessiva e fornire una chiara comprensione del gioco e delle scelte effettuate durante lo sviluppo.

Dopo una dettagliata valutazione del progetto iniziale, abbiamo riconosciuto l'importanza di fornire una documentazione esaustiva per agevolare la comprensione del funzionamento del gioco e delle decisioni di progettazione adottate.
Pertanto, la documentazione è stata suddivisa in sezioni specifiche, ciascuna delle quali affronta un aspetto cruciale del progetto. Le sezioni principali includono:

***Requisiti funzionali**: Questa sezione descrive in dettaglio i requisiti funzionali del gioco dell'oca, cioè le principali funzionalità che il sistema deve fornire. Vengono identificate le azioni che gli utenti possono compiere all'interno del gioco.*

***Progettazione del tipo di dato e delle strutture dati**: Questa sezione presenta i tipi di dato e le strutture dati utilizzate nel contesto del progetto. Vengono fornite spiegazioni dettagliate sulle scelte di progettazione adottate per rappresentare le informazioni all'interno del gioco, nonché per gestire le relazioni tra gli elementi di gioco.*

***Testing**: La sezione di testing si focalizza sull'attività di verifica e validazione del software del gioco dell'oca. Vengono presentati i casi di test sviluppati per testare le diverse funzionalità del sistema, inclusi i moduli responsabili della gestione del percorso, delle partite, del menu, dei file e della classifica. Ogni caso di test include l'input fornito, l'output atteso e una descrizione delle condizioni previste.*

***Presentazione della soluzione:** Questa sezione fornisce una panoramica completa della soluzione implementata per il gioco dell'oca. Vengono descritte le scelte di progettazione attraverso la visione dell'esecuzione del gioco con degli screenshot.*

Per avviare il gioco...

Il progetto per essere avviato va buildato e aperto l'exe dalla cartella del debug, oppure se avvirarlo da system editor cambiare l'intero percorso dei file.

Analisi

Descrizione del sistema

All'avvio del gioco, viene visualizzata un'interfaccia iniziale che consente all'utente di aprire un file di testo principale. Questo file contiene le informazioni di base del gioco, come le regole e le descrizioni dei vari menu.
Dopo aver aperto il file di testo, l'utente viene invitato a premere un tasto per continuare, il che lo porta al menu principale.
Il menu principale offre diverse opzioni:

-"Nuova partita" (opzione 1): questa scelta apre un sottomenu che permette all'utente di selezionare tra due tipi di partite: "partita classica" e "partita personalizzata".
-"Carica partita" (opzione 2): questa opzione consente all'utente di caricare una partita precedentemente salvata.
-"Aiuto" (opzione 3): questa opzione fornisce all'utente informazioni e istruzioni sul gioco.
-"Classifica" (opzione 4) : questa opzione mostra la classifica dei giocatori in base alle partite precedenti.
-"Esci" (opzione 0): questa opzione permette all'utente di uscire dal gioco.

Se l'utente seleziona l'opzione "Nuova partita" nel menu principale, viene visualizzato un sottomenu che permette di scegliere tra due tipi di partite:

-"Iniziare partita classica" (opzione 1): questa scelta avvia una partita classica con una dimensione del tabellone predefinita di 90 caselle e 4 giocatori. L'utente viene quindi invitato a inserire il nome di ciascun

giocatore, dopodiché viene stampato a schermo il tabellone di gioco. - "Iniziare partita personalizzata" (opzione 2): questa scelta consente all'utente di specificare la dimensione del tabellone e il numero di giocatori per una partita personalizzata. Successivamente, l'utente viene chiesto di inserire i nomi dei giocatori e viene stampato a schermo il tabellone di gioco corrispondente alle specifiche.
Una volta avviata una partita, il gioco procede secondo le regole del gioco dell'oca, che includono il lancio dei dadi, il movimento dei giocatori sulla tabella, l'attivazione di caselle speciali.
Durante il gioco, l'utente può eseguire varie azioni come lanciare i dadi, muovere il proprio giocatore, visualizzare le informazioni sulla casella corrente, salvare la partita corrente e uscire dal gioco, infatti verrà visualizzato ad ogni turno il menu seguente :

-“Salva”(opzione 1): questa scelta permette di salvare la partita corrente in uno dei 5 slot disponibili e di continuare a giocare la partita

-“Salva ed Esci”(opzione 2): questa scelta permette di salvare la partita corrente in uno dei 5 slot disponibili e di uscire dalla partita tornando al menù principale.

-“Abbandona”(opzione 3) : questa scelta permette di abbandonare la partita senza effettuare alcun tipo di salvataggio.

-“Gioca Ancora”(opzione 4) : questa scelta permette di andare avanti nel gioco e di avanzare di turno.

Al termine di una partita, i punteggi dei giocatori vengono registrati e la classifica viene aggiornata per riflettere il risultato della partita. L'utente ha anche la possibilità di salvare la partita corrente in qualsiasi momento e di caricare partite salvate in precedenza per riprenderle.

Se l'utente selezione l'opzione "Carica partita" nel menu principale, viene visualizzato un sottomenu a schermo che permette di scegliere uno dei 5 slot in cui si è effettuato il salvataggio della partita in precedenza. Selezionando la partita richiesta, si riuscirà a riprendere a giocare la partita senza alcun problema.

Se l'utente seleziona l'opzione "Aiuto" nel menu principale, si aprirà un sottomenu che permette di scegliere tra tre opzioni :

-"Regolamento"(Opzione 1) : questa scelta permette all'utente di visualizzare l'intero regolamento del gioco dell'oca applicato alla realizzazione del progetto

-"Manuale"(Opzione 2) : questa scelta permette all'utente di visualizzare il manuale del gioco dell'oca del nostro progetto, al fine di far comprendere il funzionamento delle interfacce all'utente e dei comandi.

-"Esci" (Opzione 0) : Permette di uscire dal sottomenu.

Se l'utente seleziona l'opzione "Classifica" nel menu principale, si aprirà la classifica dei top 10 vincitori ordinati per il minor numero di lanci effettuati e che sono stati necessari al fine della vittoria della partita. La classfica conterrà :

- posizione del giocatore nella classifica
- nome del giocatore
- lanci effettuati

La classifica è ordinata attraverso i lanci effettuati in ordine crescente.

1.2 Requisiti Funzionali

In questa parte verranno descritte brevemente i requisiti funzionali principali che permettono l'eventuale funzionamento del progetto.

In questa sezione si definiscono le specifiche relative alle funzionalità e alle interazioni che devono essere implementate nel gioco. Questi requisiti stabiliscono le azioni e le regole che il gioco dovrà seguire al fine di fornire un'esperienza di gioco coerente e soddisfacente per gli utenti.

Verranno descritte le funzionalità delle funzioni principali in una tabella :

Esempio

Codice	Nome	Descrizione
R01	ricercare_il_menu	ricerca nella cartella del file quale prelevare per accedere al menu selezionato

R02	leggere_tastiera_int_verificato	Effettua la verifica sulla scelta del menu
R03	stampare_file_di_testo	Effettua la stampa del file di testo

R04	gestire_menu_aiuto	Permette la gestione del menu dell'aiuto con le eventuali scelte.
R05	gestire_menu_partita	Permette di gestire il menu della partita.
R06	configurare_partita_p	Permette di configurare la partita personalizzata.

R07	configurare_partita	Permette la configurazione della partita classica scelta dall'utente.

R08	configurare_giocatori	Permette la configurazione dei giocatori del gioco.
R09	visualizzare_partita	Permette di viusalizzare la partita con le informazioni sui giocatori e sul tabellone.
R10	gestire_turno	Permette di gestire il turno di una partita iniziata.

R11	gestire_turno_generale	Permette di visualizzare il turno successivo della partita iniziata.

R12	aggiornare_turno	Permette di aggiornare il turno di un giocatore alla fine del turno corrente.
R13	stampare_posizioni	Permette di visualizzare le posizioni dei giocatori correnti in ogni turno corrente.
R14	gestire_tabellone_lancio_vittoria	Permette di non superare la lunghezza del tabellone attraverso il lancio dei dadi per vincere, portandoti indietro.

R15	stampare_vincitore	Permette di effettuare la stampa del vincitore finale della partita.
R16	gestire_menu_caricare_partita	Permette di gestire il menu per caricare la partita dallo slot.
R17	scegliere_opzione_di_menu	Permette di selezionare le scelte presenti nell'attuale menu selezionato.
R18	gestire_partita_in_corso	Permette di gestire la partita in corso tramite il sottomenu apposito

2. Progettazione

2.1 Progettazione dei tipi di dato, delle strutture dati

La sezione di Progettazione del gioco dell'oca si concentra sulla definizione delle strutture dati e dei tipi di dato utilizzati nel contesto del progetto, nonché sull'individuazione dei file utilizzati nel programma e del loro scopo specifico.

Tipi di dato e Strutture dati:
Nel progetto del gioco dell'oca, vengono utilizzati diversi tipi di dato e strutture dati per rappresentare le informazioni necessarie al funzionamento del gioco. Le strutture dati utilizzate sono rappresentate nella tabella :

Esempio

Nome	Tipologia	Descrizione	Tipi / Campi / Valori
giocatore	struct	Tipo di dato definito per descrivere il giocatore nel gioco dell'oca	*posizione: intero blocco : intero lanci: intero.* nome_giocatore : char[LUNGH_NOME]

Nome	Tipologia	Descrizione	Tipi / Campi / Valori

tabellone	struct	Tipo di dato definito per descrivere il tabellone del gioco dell'oca	*percorso: intero[MAX_PERCORSO]* *dimensione : intero*

Nome	Tipologia	Descrizione	Tipi / Campi / Valori
competizione_oca	struct	Tipo di dato definito per descrivere l'intera partita del gioco dell'oca	*num_giocatori : intero.* *turno : intero.* *tabellone :* *di tipo tabellone.* *giocatori :* *giocatore[MAX_GIOCATORI]*

Nome	Tipologia	Descrizione	Tipi / Campi / Valori
LUNG_NOME	costante	Costante utilizzata per indicare la lunghezza del nome del giocatore	intero
PARTITA_INTERROTTA	Costante	Costante utilizzata per indicare la partita interrotta	intero
MIN_GIOCATORI	Costante	Costante utilizzata per indicare i giocatori minimi che possono competere nel gioco dell'oca	intero
MAX_GIOCATORI	Costante	Costante utilizzata per indicare i giocatori massimo che possono competere nel gioco dell'oca	intero

NUOVA_RIGA	Costante	Costante che indica di andare a capo (“\n”)	char

SEPARATORE	Costante	Costante che indica il carattere separatore(“|”)	char
GIALLO	Costante	Costante che indica il colore giallo	char
ROSSO	Costante	Costante che indica il colore rosso	char
MARRONE	Costante	Costante che indica il colore marrone	char
VERDE	Costante	Costante che indica il colore verde	char
BLU	Costante	Costante che indica il colore blu	char

ARANCIONE	Costante	Costante che indica il colore arancione	char
VIOLA	Costante	Costante che indica il colore viola	char

CYAN	Costante	Costante che indica il colore cyan	char
MIN_PERCORSO	Costante	Costante che rappresenta il valore minimo del percorso del gioco dell'oca (50)	intero
MAX_PERCORSO	Costante	Costante che rappresenta il valore massimo del percorso del gioco dell'oca (90)	intero
RESET_COLORE	Costante	Costante che rappresenta il reset del colore (reset effettuato al colore bianco)	intero
COLORE_ARANCIONE	Costante	Costante che rappresenta il colore arancione	char

CASELLA_VUOTA	Costante	Costante che rappresenta il valore di una casella non speciale	Intero
POSIZIONE_OCA	Costante	Costante che rappresenta il valore di una casella oca	Intero

POSIZIONE_PONTE	Costante	Costante che rappresenta il valore della posizione del ponte nel tabellone	Intero
POSIZIONE_LOCANDA	Costante	Costante che rappresenta il valore della posizione della locanda nel tabellone	intero
POSIZIONE_POZZO	Costante	Costante che rappresenta il valore della posizione del pozzo nel tabellone	intero
POSIZIONE_LABIRINTO	Costante	Costante che rappresenta il valore della posizione del labirinto nel tabellone	Intero

POSIZIONE_SCHELETRO	Costante	Costante che rappresenta il valore della posizione dello scheletro nel tabellone	Intero
INIZIO	Costante	Costante che rappresenta l'inizio della partita	intero

LANCIO_MINIMO	Costante	Costante che rappresenta il lancio minimo di un dado	Intero
LANCIO_MASSIMO	Costante	Costante che rappresenta il lancio massimo di un dado	Intero
FINE	Costante	Costante che rappresenta la fine della partita	intero
LANCIO_OCA	Costante	Costante che rappresenta il lancio necessario e preciso per raggiungere la casella oca al primo lancio	intero

POSIZIONE_PRIGIONE	Costante	Costante che rappresenta il valore della posizione della prigione nel tabellone	Intero
PROMPT_ERRORE	Costante	Costante che rappresenta un messaggio da vedere a schermo in caso di errore nel caricamento della partita	char

SCELTA_ESCI	Costante	Costante che rappresenta la scelta dell'uscita dai menu	intero
SCELTA_PARTITA_C	Costante	Costante che rappresenta la scelta della partita classica	intero
SCELTA_PARTITA_P	Costante	Costante che rappresenta la scelta della partita personalizzata	intero
SCELTA_SALVA	Costante	Costante che rappresenta la scelta salva	intero

SCELTA_S_E	Costante	Costante che rappresenta la scelta salva ed esci	intero
SCELTA_ABBANDONA	Costante	Costante che rappresenta la scelta abbandona	intero
SCELTA_CONTINUA	Costante	Costante che rappresenta la scelta continua	intero
LETTURA_TESTO	Costante	Costante che rappresenta la lettura nel file di testo	char

SCRITTURA_TESTO	Costante	Costante che rappresenta la scrittura nel file di testo	char
LETTURA_BINARIA	Costante	Costante che rappresenta la lettura nel file binario	intero
SCRITTURA_BINARIA	Costante	Costante che rappresenta la lettura nel file binario	intero

INTERFACCIA_INIZIALE	Costante	Costante che rappresenta l'interfaccia iniziale	intero
PRINCIPALE	Costante	Costante che rappresenta l'interfaccia principale	intero
PARTITA	Costante	Costante che rappresenta l'interfaccia della partita	intero
CARICA	Costante	Costante che rappresenta l'interfaccia del caricamento della partita	intero
AIUTO	Costante	Costante che rappresenta l'interfaccia del aiuto	intero

CLASSIFICA	Costante	Costante che rappresenta l'interfaccia della classifica	intero
CLASSIFICA_BINARIA	Costante	Costante che rappresenta l'interfaccia della classifica binaria	intero

REGOLE	Costante	Costante che rappresenta l'interfaccia delle regole	intero
MANUALE	Costante	Costante che rappresenta l'interfaccia del manuale	intero
PARTITA_IN_CORSO	Costante	Costante che rappresenta l'interfaccia della partita in corso nel gioco dell'oca	intero
FINE_STRINGA	Costante	Costante che rappresenta la fine della stringa nell'array di caratteri ("\0")	intero
MAX_SLOT	Costante	Costante che rappresenta il numero massimo degli slot	intero

SOTTOMENU	Costante	Costante che rappresenta un prompt che si vede a schermo	intero
ERRORE_FILE	Costante	Costante che rappresenta l'errore	intero

		nell'apertuta nel file di testo	
MAX_PERCORSO_FILE	Costante	Costante che rappresenta la grandezza massima del percorso del file	intero
MAX_CLASSIFICATI	Costante	Costante che rappresenta il massimo dei classificati (10)	intero
NO_RICERCA	Costante	Costante che rappresenta la non ricerca del file nel momento in cui quest'utlimo non viene trovato	intero
FINE_LISTA	Costante	Costante che rappresenta se si è arrivati a fine lista	intero
ERRORE_SLOT	Costante	Costante che rappresenta l'errore nel salvataggiodella partita in uno slot	intero

PROMPT_ERRORE_SLOT	Costante	Costante che rappresenta l'errore nel salvataggio nell'errore dello slot	char

3. Testing

La sezione di Testing del gioco dell'oca si propone di verificare la correttezza e l'affidabilità dei moduli principali del software, al fine di garantire un'esperienza di gioco senza errori e malfunzionamenti. Saranno sottoposti a test i seguenti moduli:

Modulo "gestire_percorso": Questo modulo si occupa di gestire il percorso di gioco, calcolando le nuove posizioni dei giocatori in base ai lanci dei dadi e agli effetti delle caselle speciali. Saranno eseguiti test per verificare che le posizioni vengano calcolate correttamente e che gli effetti delle caselle vengano applicati nel modo previsto.

Modulo "gestire_partite": Questo modulo si occupa di gestire il flusso di gioco all'interno di una partita, includendo la configurazione del tabellone, la creazione dei giocatori e l'alternanza dei loro turni. Saranno eseguiti test per verificare che la configurazione e la gestione delle partite avvengano correttamente, garantendo un'esperienza di gioco fluida e coerente.

Modulo "gestire_menu": Questo modulo si occupa di gestire il menu principale del gioco, offrendo all'utente le opzioni per avviare una nuova partita, configurare le impostazioni o visualizzare la classifica. Saranno eseguiti test per verificare che il menu risponda correttamente agli input dell'utente e che le funzionalità del gioco siano accessibili tramite il menu.

Modulo "gestire_file": Questo modulo si occupa di gestire la lettura e la scrittura dei dati su file, ad esempio per salvare lo stato di una partita o per caricare la classifica dei giocatori. Saranno eseguiti test per verificare che la lettura e la scrittura dei dati avvengano correttamente e che i file vengano gestiti in modo sicuro e affidabile.

Modulo "gestire_classifica": Questo modulo si occupa di gestire la classifica dei giocatori, tenendo traccia dei loro punteggi e delle loro prestazioni nel corso delle partite. Saranno eseguiti test per verificare che la classifica venga aggiornata correttamente e che i punteggi vengano calcolati in base alle regole del gioco.

Durante i test, si farà uso di una serie di casi di test che copriranno diversi scenari di gioco, inclusi casi limite e situazioni particolari. L'obiettivo sarà quello di individuare eventuali errori o bug nel software e di correggerli prima della distribuzione finale del gioco.

3.1 Esiti del Piano di Test ed eventuali commenti

CASI DI TEST MODULO – GESTIRE_PERCORSO

Funzione calcolare_proporzione Caso di test 1:
Descrizione: Valori positivi non nulli per gli elementi della proporzione.
Input: 5, 3, 2
Output Atteso: 3 Caso di test 2:
Descrizione: Valori negativi non nulli per gli elementi della proporzione.
Input: -2, -3, -4
Output Atteso: 2 Caso di test 3:
Descrizione: Tutti gli elementi della proporzione sono nulli. Input: 0, 0, 0
Output Atteso: 0 Caso di test 4:
Descrizione: Il primo elemento è nullo, il secondo e il quarto elemento sono non nulli. Input: 0, 1, 5

Output Atteso: 0

Funzione inserire_caselle_oca:

Caso di test 1:

Descrizione: La board di gioco è composta da 55 caselle. Tutte le caselle sono inizializzate a 0. Vogliamo verificare che le caselle "goose" vengano posizionate correttamente ogni 9 caselle.
Input: Una board di gioco con 55 caselle, tutte impostate a 0. Output atteso: Le caselle "goose" vengono posizionate ogni 9 caselle, ottenendo un board con 5 caselle "goose", mentre le rimanenti caselle rimangono a 0.

Caso di test 2:

Descrizione: La board di gioco è composta da 85 caselle. Tutte le caselle sono inizializzate a 0. Vogliamo verificare che le caselle "goose" vengano posizionate correttamente ogni 9 caselle.
Input: Una board di gioco con 85 caselle, tutte impostate a 0. Output atteso: Le caselle "goose" vengono posizionate ogni 9 caselle, ottenendo una board con 10 caselle "goose", mentre le rimanenti caselle rimangono a 0.

Caso di test 3:

Descrizione: La board di gioco è composta da 95 caselle. Tutte le caselle sono inizializzate a 0. Vogliamo verificare che le caselle "goose" vengano posizionate correttamente ogni 9 caselle.
Input: Una board di gioco con 95 caselle, tutte impostate a 0. Output atteso: Le caselle "goose" vengono posizionate ogni 9 caselle, ottenendo una board con 11 caselle "goose", mentre le rimanenti caselle rimangono a 0.

Caso di test 4:

Descrizione: La board di gioco è composta da 70 caselle. Tutte le caselle sono inizializzate a 0. Vogliamo verificare che le caselle "goose" vengano posizionate correttamente ogni 9 caselle.

Input: Una board di gioco con 70 caselle, tutte impostate a 0. Output atteso: Le caselle "goose" vengono posizionate ogni 9 caselle, ottenendo una board con 7 caselle "goose", mentre le rimanenti caselle rimangono a 0.

Funzione stabilire_percorso:

Caso di test 1:

Descrizione: Si desidera creare una mappa di gioco con il numero minimo consentito di caselle (50). La mappa deve essere completamente vuota.
Input: Una game board con il numero minimo di caselle (50).
Output atteso: La game board viene restituita con il campo "tabellone_percorso" inizializzato con 50 caselle impostate a 0 (casella normale). Caso di test 2:
Descrizione: Si desidera creare una mappa di gioco con il numero massimo consentito di caselle (90). La mappa deve essere completamente vuota.
Input: Una game board con il numero massimo di caselle (90).
Output atteso: La game board viene restituita con il campo "tabellone_percorso" inizializzato con 90 caselle impostate a 0 (casella normale). Caso di test 3:
Descrizione: Si desidera creare una mappa di gioco con un numero di caselle compreso tra il minimo e il massimo consentito (70). La mappa deve essere completamente vuota.
Input: Una game board con un numero di caselle pari a 70.
Output atteso: La game board viene restituita con il campo "tabellone_percoso" inizializzato con 70 caselle impostate a 0 (casella normale). Caso di test 4:
Descrizione: Si desidera creare una mappa di gioco con un numero negativo di caselle (-10). Nonostante il valore negativo, la funzione dovrebbe inizializzare correttamente la mappa.
Input: Una game board con un numero di caselle pari a -10. Output atteso: La game board viene restituita con il campo "tabellone_percorso" inizializzato con 0 caselle.

Funzione stampare_riga:

Caso di test 1:

Descrizione: Verifica se la funzione incrementa correttamente la posizione quando la posizione corrente è minore o uguale al numero totale di caselle del tabellone e il numero di elementi stampati nella riga è inferiore al numero di caselle desiderate per riga.
Input: numero_caselle = 10, elementi_riga_corrente = 0, caselle_per_riga = 5, posizione_corrente = 1, posizione_turno = 3
Output atteso: La posizione successiva dopo aver stampato l'elemento corrente dovrebbe essere posizione_corrente = 5.

Caso di test 2:

Descrizione: Verifica se la funzione incrementa correttamente la posizione quando la posizione corrente è minore o uguale al numero totale di caselle del tabellone e il numero di elementi stampati nella riga è inferiore al numero di caselle desiderate per riga. Input: numero_caselle = 20, elementi_riga_corrente = 3, caselle_per_riga = 4, posizione_corrente = 7, posizione_turno = 10
Output atteso: La posizione successiva dopo aver stampato l'elemento corrente dovrebbe essere posizione_corrente = 8.

Caso di test 3:

Descrizione: Verifica se la funzione non incrementa la posizione quando la posizione corrente supera il numero totale di caselle del tabellone. Input: numero_caselle = 5, elementi_riga_corrente = 4, caselle_per_riga = 5, posizione_corrente = 6, posizione_turno = 2 Output atteso: La posizione dovrebbe rimanere uguale a posizione_corrente = 6 poiché supera il numero totale di caselle disponibili.

Funzione stampare_ riga_invertita

Caso di test 1:

Descrizione: Verifica se la funzione stampa correttamente gli elementi quando il numero di elementi stampati nella riga è diverso dal numero di caselle mancanti per completare la riga.
Input: numero_caselle = 10, elementi_riga_corrente = 2, caselle_per_riga = 4, posizione_corrente = 5, posizione_turno = 3
Output atteso: La posizione successiva dopo aver stampato l'elemento corrente dovrebbe essere posizione_corrente = 8, e il numero di elementi stampati nella riga dovrebbe essere uguale al numero di caselle mancanti.

Caso di test 2:

Descrizione: Verifica se la funzione stampa correttamente gli elementi quando il numero di elementi stampati nella riga è diverso dal numero di caselle mancanti per completare la riga.
Input: numero_caselle = 15, elementi_riga_corrente = 5, caselle_per_riga = 3, posizione_corrente = 12, posizione_turno = 8
Output atteso: La posizione successiva dopo aver stampato l'elemento corrente dovrebbe essere posizione_corrente = 16, e il numero di elementi stampati nella riga dovrebbe essere uguale al numero di caselle mancanti.

Caso di test 3:

Descrizione: Verifica se la funzione stampa correttamente gli elementi quando il numero di elementi stampati nella riga è uguale al numero di caselle mancanti per completare la riga.
Input: numero_caselle = 8, elementi_riga_corrente = 2, caselle_per_riga = 6, posizione_corrente = 4, posizione_turno = 6
Output atteso: La posizione successiva dopo aver stampato l'elemento corrente dovrebbe essere posizione_corrente = 8.

Funzione stampare_riga_minima:

Caso di test 1:

: Verifica se la funzione stampa correttamente gli elementi quando il numero di elementi stampati nella riga è diverso dal numero di caselle da stampare per riga.
Input: numero_elementi_riga = 3, caselle_per_riga = 5, posizione_corrente = 9, posizione_turno = 6
Output atteso: La posizione successiva dopo aver stampato l'elemento corrente dovrebbe essere posizione_corrente = 14, e il numero di elementi stampati nella riga dovrebbe essere uguale al numero di caselle da stampare per riga.

Caso di test 2:

Descrizione: Verifica se la funzione stampa correttamente gli elementi quando il numero di elementi stampati nella riga è diverso dal numero di caselle da stampare per riga.
Input: numero_elementi_riga = 2, caselle_per_riga = 4, posizione_corrente = 6, posizione_turno = 2

Output atteso: La posizione successiva dopo aver stampato l'elemento corrente dovrebbe essere posizione_corrente = 11, e il numero di elementi stampati nella riga dovrebbe essere uguale al numero di caselle da stampare per riga.

CASI DI TEST MODULO – GESTIRE_PARTITE

Funzione aggiornare_turno:

Caso di test 1:
Descrizione: Aggiornamento del turno quando il numero di giocatori è superiore a 1.
Input:

turno: valore intero rappresentante il turno corrente. num_giocatori: valore intero indicante il numero totale di giocatori.
Output atteso: Il valore del turno viene incrementato di 1.
Caso di test 2:
Descrizione: Aggiornamento del turno quando il numero di giocatori è 1.
Input:
turno: valore intero rappresentante il turno corrente.
num_giocatori: valore intero indicante il numero totale di giocatori.
Output atteso: Il valore del turno viene impostato a 0.
Caso di test 3:
Descrizione: Aggiornamento del turno quando il turno corrente è 0 e il numero di giocatori è 1.
Input:
turno: valore intero rappresentante il turno corrente.
num_giocatori: valore intero indicante il numero totale di giocatori.
Output atteso: Il valore del turno viene incrementato di 1.

Funzione fissare_turno:

Caso di test 1:
Descrizione: Fissare il turno iniziale con un valore casuale. Input:

turno: valore intero rappresentante il turno corrente. num_giocatori: valore intero indicante il numero totale di giocatori. Output atteso: Il valore del turno viene generato casualmente tra 1 e il numero totale di giocatori.
Caso di test 2:
Descrizione: Fissare il turno iniziale quando il turno corrente non è quello di inizio.
Input:
turno: valore intero rappresentante il turno corrente.
num_giocatori: valore intero indicante il numero totale di giocatori.
Output atteso: Nessuna modifica al valore del turno corrente.
Caso di test 3:
Descrizione: Fissare il turno iniziale quando il numero di giocatori è minore di 2. Input:
turno: valore intero rappresentante il turno corrente. num_giocatori: valore intero indicante il numero totale di giocatori. Output atteso: Il valore del turno viene generato casualmente tra 1 e il numero totale di giocatori.
Funzione gestire_caselle_speciali:
Caso di test 1:
Descrizione: Gestire una casella speciale di tipo "Oca" con i lanci dei dadi.
Input:

casella_speciale: valore rappresentante il tipo di casella speciale.
lancio_1: valore intero del primo lancio del dado. lancio_2: valore intero del secondo lancio del dado.
Output atteso: Viene chiamata la funzione "gestire_casella_oca" con i lanci dei dadi come argomenti.
Caso di test 2:
Descrizione: Gestire una casella speciale di tipo "Ponte" con un lancio del dado.
Input:
casella_speciale: valore rappresentante il tipo di casella speciale.
lancio: valore intero del lancio del dado.
Output atteso: Viene chiamata la funzione "gestire_casella_ponte" con il lancio del dado come argomento.

Caso di test 3:
Descrizione: Gestire una casella speciale di tipo "Locanda".
Input:
casella_speciale: valore rappresentante il tipo di casella speciale.
Output atteso: Viene chiamata la funzione "gestire_casella_locanda".
Caso di test 4:
Descrizione: Gestire una casella speciale di tipo "Pozzo".
Input:
casella_speciale: valore rappresentante il tipo di casella speciale.
Output atteso: Viene chiamata la funzione "gestire_casella_blocco".
Caso di test 5:
Descrizione: Gestire una casella speciale di tipo "Labirinto".
Input:
casella_speciale: valore rappresentante il tipo di casella speciale.
Output atteso: Viene chiamata la funzione "gestire_casella_labirinto".
Caso di test 6:
Descrizione: Gestire una casella speciale di tipo "Scheletro".
Input:
casella_speciale: valore rappresentante il tipo di casella speciale.
Output atteso: Viene chiamata la funzione "gestire_casella_scheletro".
Funzione gestire_casella_oca:
Caso di test 1:
Descrizione: Gestire una casella "Oca" quando il giocatore corrente è il primo giocatore.
Input:

partita_in_corso: oggetto rappresentante la partita in corso.
lanci_giocatore: lista dei lanci dei dadi del giocatore corrente.
Output atteso: La posizione del giocatore viene aggiornata alla posizione speciale 1 (26).
Caso di test 2:
Descrizione: Gestire una casella "Oca" quando il giocatore corrente è il primo giocatore.
Input:
partita_in_corso: oggetto rappresentante la partita in corso.
lanci_giocatore: lista dei lanci dei dadi del giocatore corrente.

Output atteso: La posizione del giocatore viene aggiornata alla posizione speciale 2 (53).
Caso di test 3:
Descrizione: Gestire una casella "Oca" quando il giocatore corrente non è il primo giocatore.
Input:
partita_in_corso: oggetto rappresentante la partita in corso.
lanci_giocatore: lista dei lanci dei dadi del giocatore corrente. Output atteso: La posizione del giocatore viene aggiornata sommando i lanci dei dadi. Caso di test 4:
Descrizione: Gestire una casella "Oca" quando il giocatore corrente non è il primo giocatore.
Input:
partita_in_corso: oggetto rappresentante la partita in corso.
lanci_giocatore: lista dei lanci dei dadi del giocatore corrente. Output atteso: La posizione del giocatore viene aggiornata sommando i lanci dei dadi. Caso di test 5:
Descrizione: Gestire una casella "Oca" quando il giocatore corrente non è il primo giocatore.
Input:
partita_in_corso: oggetto rappresentante la partita in corso.
lanci_giocatore: lista dei lanci dei dadi del giocatore corrente. Output atteso: La posizione del giocatore viene aggiornata sommando i lanci dei dadi. Caso di test 6:
Descrizione: Gestire una casella "Oca" quando il giocatore corrente non è il primo giocatore.
Input:
partita_in_corso: oggetto rappresentante la partita in corso.
lanci_giocatore: lista dei lanci dei dadi del giocatore corrente.
Output atteso: La posizione del giocatore viene aggiornata sottraendo la somma dei lanci dei dadi alla posizione attuale del giocatore.
Funzione gestire_casella_ponte:
Caso di test 1:
Descrizione: Gestire una casella "Ponte" quando il turno corrente del giocatore è 0.
Input:

partita_in_corso: oggetto rappresentante la partita in corso.
lancio: valore intero rappresentante il lancio del dado del giocatore corrente.
Output atteso: La posizione del giocatore viene aggiornata sommando il lancio alla posizione attuale.
Caso di test 2:
Descrizione: Gestire una casella "Ponte" quando il turno corrente del giocatore non è 0.
Input:
partita_in_corso: oggetto rappresentante la partita in corso. lancio: valore intero rappresentante il lancio del dado del giocatore corrente.
Output atteso: La posizione del giocatore viene aggiornata sommando il lancio alla posizione attuale.
Caso di test 3:
Descrizione: Gestire una casella "Ponte" quando il turno corrente del giocatore non è 0.
Input:
partita_in_corso: oggetto rappresentante la partita in corso. lancio: valore intero rappresentante il lancio del dado del giocatore corrente.
Output atteso: La posizione del giocatore viene aggiornata sommando il lancio alla posizione attuale.
Funzione gestire_casella_locanda:
Caso di test 1:
Descrizione: Gestire una casella "Locanda" quando il turno corrente del giocatore è 0.
Input:

partita_in_corso: oggetto rappresentante la partita in corso. Output atteso: La variabile "blocco" del giocatore corrente viene aggiornata a -3, indicando che il giocatore è bloccato per 3 turni.
Caso di test 2:
Descrizione: Gestire una casella "Locanda" quando il turno corrente del giocatore non è 0.

Input:
partita_in_corso: oggetto rappresentante la partita in corso. Output atteso: La variabile "blocco" del giocatore corrente viene aggiornata a -3, indicando che il giocatore è bloccato per 3 turni.
Caso di test 3:
Descrizione: Gestire una casella "Locanda" quando il turno corrente del giocatore non è 0.
Input:
partita_in_corso: oggetto rappresentante la partita in corso. Output atteso: La variabile "blocco" del giocatore corrente viene aggiornata a -3, indicando che il giocatore è bloccato per 3 turni.
Funzione liberare_giocatore:
Caso di test 1:
Descrizione: Liberare un giocatore bloccato sulla stessa casella di un altro giocatore.
Input:

partita_in_corso: oggetto rappresentante la partita in corso.
giocatore_bloccato: indice del giocatore bloccato sulla casella di blocco.
giocatore_liberato: indice del giocatore che si trova sulla stessa casella.
Output atteso: La variabile "blocco" del giocatore liberato viene aggiornata a 0, indicando che il giocatore è stato liberato dal blocco.
Caso di test 2:
Descrizione: Liberare un giocatore bloccato sulla stessa casella di un altro giocatore.
Input:
partita_in_corso: oggetto rappresentante la partita in corso.
giocatore_bloccato: indice del giocatore bloccato sulla casella di blocco.
giocatore_liberato: indice del giocatore che si trova sulla stessa casella.
Output atteso: La variabile "blocco" del giocatore liberato viene aggiornata a 0, indicando che il giocatore è stato liberato dal blocco.
Caso di test 3:
Descrizione: Liberare un giocatore bloccato sulla stessa casella di un altro giocatore.
Input:

partita_in_corso: oggetto rappresentante la partita in corso.
giocatore_bloccato: indice del giocatore bloccato sulla casella di blocco.
giocatore_liberato: indice del giocatore che si trova sulla stessa casella.
Output atteso: La variabile "blocco" del giocatore liberato viene aggiornata a 0, indicando che il giocatore è stato liberato dal blocco.

Funzione gestire_casella_blocco:

Caso di test 1:
Descrizione: Gestire una casella di blocco (Pozzo o Prigione) quando un giocatore finisce su di essa.
Input:

partita_in_corso: oggetto rappresentante la partita in corso. giocatore: indice del giocatore che finisce sulla casella di blocco.
Output atteso: La variabile "blocco" del giocatore viene impostata a un valore negativo (-INFINITO nel caso del pozzo o prigione) per indicare che il giocatore è bloccato finché non si verifichi una particolare condizione.
Caso di test 2:
Descrizione: Gestire una casella di blocco (Pozzo o Prigione) quando un giocatore finisce su di essa.
Input:
partita_in_corso: oggetto rappresentante la partita in corso. giocatore: indice del giocatore che finisce sulla casella di blocco. Output atteso: La variabile "blocco" del giocatore viene impostata a un valore negativo (-INFINITO nel caso del pozzo o prigione) per indicare che il giocatore è bloccato finché non si verifichi una particolare condizione. Caso di test 3:
Descrizione: Gestire una casella di blocco (Pozzo o Prigione) quando un giocatore finisce su di essa e c'è un altro giocatore bloccato sulla stessa casella.
Input:
partita_in_corso: oggetto rappresentante la partita in corso. giocatore: indice del giocatore che finisce sulla casella di blocco.
Output atteso: La variabile "blocco" del giocatore che finisce sulla casella di blocco viene impostata a un valore negativo (-INFINITO nel caso del pozzo o prigione) per indicare che il giocatore è bloccato finché non si verifichi una particolare condizione. La variabile "blocco" dell'altro

giocatore viene impostata a 0, indicando che il giocatore è stato liberato dal blocco.

Funzione gestire_casella_labirinto:

Caso di test 1:

Descrizione: Gestire una casella "Labirinto" quando un giocatore finisce su di essa.

Input:

partita_in_corso: oggetto rappresentante la partita in corso. giocatore: indice del giocatore che finisce sulla casella labirinto.

Output atteso: La variabile "posizione" del giocatore viene impostata a un valore specifico (calcolato in base alla posizione di ritorno) per indicare che il giocatore è mandato alla casella di ritorno dopo essere finito sulla casella labirinto.

Caso di test 2:

Descrizione: Gestire una casella "Labirinto" quando un giocatore finisce su di essa.

Input:

partita_in_corso: oggetto rappresentante la partita in corso. giocatore: indice del giocatore che finisce sulla casella labirinto.

Output atteso: La variabile "posizione" del giocatore viene impostata a un valore specifico (calcolato in base alla posizione di ritorno) per indicare che il giocatore è mandato alla casella di ritorno dopo essere finito sulla casella labirinto.

Caso di test 3:

Descrizione: Gestire una casella "Labirinto" quando un giocatore finisce su di essa.

Input:

partita_in_corso: oggetto rappresentante la partita in corso. giocatore: indice del giocatore che finisce sulla casella labirinto.

Output atteso: La variabile "posizione" del giocatore viene impostata a un valore specifico (calcolato in base alla posizione di ritorno) per indicare che il giocatore è mandato alla casella di ritorno dopo essere finito sulla casella labirinto.

Funzione gestire_casella_scheletro:

Caso di test 1:

Descrizione: Gestire una casella "Scheletro" quando un giocatore finisce su di essa.
Input:

partita_in_corso: oggetto rappresentante la partita in corso.
giocatore: indice del giocatore che finisce sulla casella scheletro.
Output atteso: La variabile "posizione" del giocatore viene impostata a 1 per indicare che il giocatore è mandato alla casella di ritorno dopo essere finito sulla casella scheletro.
Caso di test 2:
Descrizione: Gestire una casella "Scheletro" quando un giocatore finisce su di essa.
Input:
partita_in_corso: oggetto rappresentante la partita in corso. giocatore: indice del giocatore che finisce sulla casella scheletro.
Output atteso: La variabile "posizione" del giocatore viene impostata a 1 per indicare che il giocatore è mandato alla casella di ritorno dopo essere finito sulla casella scheletro.
Caso di test 3:
Descrizione: Gestire una casella "Scheletro" quando un giocatore finisce su di essa.
Input:
partita_in_corso: oggetto rappresentante la partita in corso. giocatore: indice del giocatore che finisce sulla casella scheletro.
Output atteso: La variabile "posizione" del giocatore viene impostata a 1 per indicare che il giocatore è mandato alla casella di ritorno dopo essere finito sulla casella scheletro.
Funzione gestire_tabellone_lancio_vittoria:
Caso di test 1:
Descrizione: Gestire il movimento del giocatore quando si verifica un lancio che supera la dimensione del tabellone.
Input:

posizione_attuale: posizione attuale del giocatore.
dimensione: dimensione del tabellone. lancio: valore intero rappresentante il lancio del dado.

Output atteso: La posizione attuale del giocatore rimane invariata.
Caso di test 2:
Descrizione: Gestire il movimento del giocatore quando si verifica un lancio che supera la dimensione del tabellone.
Input:
posizione_attuale: posizione attuale del giocatore.
dimensione: dimensione del tabellone. lancio: valore
intero rappresentante il lancio del dado.
Output atteso: La posizione attuale del giocatore viene aggiornata sottraendo il valore del lancio dalla posizione attuale.
Caso di test 3:
Descrizione: Gestire il movimento del giocatore quando si verifica un lancio che supera la dimensione del tabellone.
Input:
posizione_attuale: posizione attuale del giocatore.
dimensione: dimensione del tabellone. lancio: valore
intero rappresentante il lancio del dado.
Output atteso: La posizione attuale del giocatore viene aggiornata sottraendo il valore del lancio dalla posizione attuale.
Caso di test 4:
Descrizione: Gestire il movimento del giocatore quando si verifica un lancio che supera la dimensione del tabellone.
Input:
posizione_attuale: posizione attuale del giocatore.
dimensione: dimensione del tabellone. lancio: valore
intero rappresentante il lancio del dado.
Output atteso: La posizione attuale del giocatore viene aggiornata sottraendo il valore del lancio dalla posizione attuale.
Funzione liberare_prigione_lancio:
Caso di test 1:
Descrizione: Liberare un giocatore dalla prigione in base al valore del lancio.
Input:

lancio: valore intero rappresentante il lancio del dado (5).
giocatore_prigione: indice del giocatore bloccato in prigione (3). Output

atteso: La variabile "blocco" del giocatore_prigione viene aggiornata a 0.
Caso di test 2:
Descrizione: Liberare un giocatore dalla prigione in base al valore del lancio.
Input:

lancio: valore intero rappresentante il lancio del dado (3).
giocatore_prigione: indice del giocatore bloccato in prigione (2). Output atteso: Nessuna modifica alla variabile "blocco" del giocatore_prigione.
Caso di test 3:
Descrizione: Liberare un giocatore dalla prigione in base al valore del lancio.
Input:

lancio: valore intero rappresentante il lancio del dado (7).
giocatore_prigione: indice del giocatore bloccato in prigione (1).
Output atteso: La variabile "blocco" del giocatore_prigione viene aggiornata a 0.
Caso di test 4:
Descrizione: Liberare un giocatore dalla prigione in base al valore del lancio.
Input:

lancio: valore intero rappresentante il lancio del dado (4).
giocatore_prigione: indice del giocatore bloccato in prigione (0). Output atteso: Nessuna modifica alla variabile "blocco" del giocatore_prigione.
Funzione configurare_partita_classica:
Caso di test 1:
Descrizione: Configurare una partita classica con impostazioni standard.
Input: scelta = 1.
Output atteso: La partita viene configurata con le impostazioni standard.

Caso di test 2:
Descrizione: Configurare una partita classica con impostazioni personalizzate. Input: scelta = 2.

Output atteso: La partita viene configurata con le impostazioni personalizzate.

Caso di test 3:
Descrizione: Richiedere nuovamente la scelta all'utente in caso di input non valido.
Input: scelta = a.
Output atteso: È richiesta nuovamente la scelta all'utente.

Caso di test 4:
Descrizione: Richiedere nuovamente la scelta all'utente in caso di input non valido.
Input: scelta = 8.
Output atteso: È richiesta nuovamente la scelta all'utente.

Funzione configurare_partita_personalizzata:
Caso di test 1:
Descrizione: Configurare una partita personalizzata con dimensione=70 e num_giocatori=3.
Input: dimensione = 70, num_giocatori = 3.
Output atteso: La partita viene configurata con le impostazioni personalizzate: dimensione=70, num_giocatori=3, viene generato un nuovo tabellone e configurata la lista dei giocatori.

Caso di test 2:
Descrizione: Richiedere nuovamente la dimensione all'utente in caso di input non valido.
Input: dimensione = 100, num_giocatori = 2.
Output atteso: È richiesta nuovamente la dimensione all'utente.

Caso di test 3:
Descrizione: Richiedere nuovamente il num_giocatori all'utente in caso di input non valido.
Input: dimensione = 60, num_giocatori = 5.
Output atteso: È richiesto il num_giocatori all'utente.

Caso di test 4:
Descrizione: Configurare una nuova partita personalizzata con dimensione=80 e num_giocatori=2. Input: dimensione = 80, num_giocatori = 2.
Output atteso: La partita viene riconfigurata con nuovi valori per dimensione e num_giocatori, viene generato un nuovo tabellone e configurata la lista dei giocatori.

Caso di test 5:
Descrizione: Configurare una nuova partita personalizzata con dimensione=60, mantenendo il numero di giocatori.
Input: dimensione = 60, num_giocatori = 4.
Output atteso: La partita viene riconfigurata con un nuovo valore per dimensione, mantenendo il numero di giocatori e generando un nuovo tabellone.

Funzione configurare_giocatori:
Caso di test 1:
Descrizione: Configurare una partita con una lista di 3 giocatori.
Input: num_giocatori = 3.
Output atteso: La partita viene configurata con una lista di 3 giocatori.

Caso di test 2:
Descrizione: Configurare una partita con una lista di 2 giocatori.
Input: num_giocatori = 2.
Output atteso: La partita viene configurata con una lista di 2 giocatori.

Caso di test 3:
Descrizione: Richiedere nuovamente l'inserimento del nome in caso di input non valido.
Input: num_giocatori = 4.
Output atteso: È richiesto nuovamente l'inserimento del nome.

Caso di test 4:
Descrizione: Riconfigurare la partita con una nuova lista di 2 giocatori, sostituendo i giocatori preesistenti.

Input: num_giocatori = 2.
Output atteso: La partita viene riconfigurata con una nuova lista di 2 giocatori, sostituendo i giocatori preesistenti.

Caso di test 5:
Descrizione: Riconfigurare la partita con una nuova lista di 3 giocatori, sostituendo i giocatori preesistenti.
Input: num_giocatori = 3.
Output atteso: La partita viene riconfigurata con una nuova lista di 3 giocatori, sostituendo i giocatori preesistenti.

Funzione gestire_turno_blocco:
Caso di test 1:
Descrizione: Gestire il blocco del turno in base alla casella "Pozzo" quando il blocco_turno è -INFINITO.
Input:

casella: valore costante rappresentante la casella "Pozzo".
blocco_turno: valore costante rappresentante il blocco del turno (INFINITO).
Output atteso: Il blocco_turno rimane -INFINITO.
Caso di test 2:
Descrizione: Gestire il blocco del turno in base alla casella "Locanda" quando il blocco_turno è 3.
Input:

casella: valore costante rappresentante la casella "Locanda". blocco_turno: valore intero rappresentante il blocco del turno (3).
Output atteso: Il blocco_turno diviene 2.
Caso di test 3:
Descrizione: Gestire il blocco del turno in base alla casella "Locanda" quando il blocco_turno è 1.
Input:

casella: valore costante rappresentante la casella "Locanda".
blocco_turno: valore intero rappresentante il blocco del turno (1).
Output atteso: Il blocco_turno diviene 0 e il giocatore è liberato.

Caso di test 4:
Descrizione: Gestire il blocco del turno in base alla casella "Prigione" quando il blocco_turno è -INFINITO.
Input:

casella: valore costante rappresentante la casella "Prigione".
blocco_turno: valore costante rappresentante il blocco del turno (INFINITO).
Output atteso: In base al risultato del lancio del dado nella funzione liberare_prigione_lancio, il blocco_turno rimane -INFINITO o diviene 0.

Funzione gestire_turno_generale:

Caso di test 1:
Descrizione: Gestire il turno generale in base alla posizione attuale, la dimensione del tabellone e il lancio del dado.
Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (88). dimensione: valore intero rappresentante la dimensione del tabellone (90). lancio: valore intero rappresentante il lancio del dado (4).
Output atteso: La posizione attuale rimane 88.
Caso di test 2:
Descrizione: Gestire il turno generale in base alla posizione attuale, la dimensione del tabellone e il lancio del dado.
Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (86). dimensione: valore intero rappresentante la dimensione del tabellone (90).
lancio: valore intero rappresentante il lancio del dado (4).
Output atteso: La variabile "fine" diviene 1, indicando la fine del gioco.
Caso di test 3: (Il giocatore si trova su una casella speciale con spostamento all'indietro negativo)
Descrizione: Gestire il turno generale in base alla posizione attuale, l'effetto di spostamento all'indietro e la casella "Oca". Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (10). indietro: valore intero rappresentante l'effetto di spostamento all'indietro (3). casella: valore costante rappresentante la casella "Oca".
Output atteso: La posizione attuale diviene 7.
Caso di test 4:
Descrizione: Gestire il turno generale in base alla posizione attuale, la dimensione del tabellone e il lancio del dado.
Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (45). dimensione: valore intero rappresentante la dimensione del tabellone (68). lancio: valore intero rappresentante il lancio del dado (5).
Output atteso: La posizione attuale diviene 50.
Caso di test 5:
Descrizione: Gestire il turno generale in base alla posizione attuale, la dimensione del tabellone e il lancio del dado con un effetto a catena di caselle "Oca".
Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (89). dimensione: valore intero rappresentante la dimensione del tabellone (90). lancio: valore intero rappresentante il lancio del dado (10).
Output atteso: La posizione attuale diviene 1.
Caso di test 6:
Descrizione: Gestire il turno generale in base alla posizione attuale, la dimensione del tabellone e il lancio del dado con una casella "Scheletro" in posizione 57.
Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (52). dimensione: valore intero rappresentante la dimensione del tabellone (90). lancio: valore intero rappresentante il lancio del dado (5).

Output atteso: La posizione attuale diviene 1.
Caso di test 7:
Descrizione: Gestire il turno generale in base alla posizione attuale, la dimensione del tabellone e il lancio del dado con una casella "Pozzo" in posizione 50.
Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (43). dimensione: valore intero rappresentante la dimensione del tabellone (83). lancio: valore intero rappresentante il lancio del dado (7).
Output atteso: La posizione attuale diviene 50 e il blocco_turno diviene INFINITO.
Caso di test 8:
Descrizione: Gestire il turno generale in base alla posizione attuale, la dimensione del tabellone e il lancio del dado con una casella "Prigione" in posizione 50.
Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (43). dimensione: valore intero rappresentante la dimensione del tabellone (83). lancio: valore intero rappresentante il lancio del dado (7).
Output atteso: La posizione attuale diviene 50 e il blocco_turno diviene INFINITO.
Caso di test 9:
Descrizione: Gestire il turno generale in base alla posizione attuale, la dimensione del tabellone e il lancio del dado con una casella "Locanda" in posizione 50.
Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (43). dimensione: valore intero rappresentante la dimensione del tabellone (83).
lancio: valore intero rappresentante il lancio del dado (7).
Output atteso: La posizione attuale diviene 50 e il blocco_turno diviene -3.

Caso di test 10:
Descrizione: Gestire il turno generale in base alla posizione attuale, la dimensione del tabellone e il lancio del dado con una casella "Ponte" in posizione 6.
Input:

posizione_attuale: valore intero rappresentante la posizione attuale del giocatore (4). dimensione: valore intero rappresentante la dimensione del tabellone (73).
lancio: valore intero rappresentante il lancio del dado (2). Output atteso: La posizione attuale diviene 8.

CASI DI TEST MODULO – GESTIRE_MENU

Funzione gestire_menu_partita:

Caso di test 1:
Descrizione: Avvio di una nuova partita.
Input: file_menu: percorso del file contenente le opzioni del menu (file di testo) Output atteso:
Messaggio di avvio di una nuova partita

Caso di test 2:
Descrizione: Tentativo di salvare la partita corrente.
Input: file_menu: percorso del file contenente le opzioni del menu (file di testo) Output atteso:
flag: valore intero 1

Caso di test 3:
Descrizione: Abbandono della partita corrente.
Input: file_menu: percorso del file contenente le opzioni del menu (file di testo) Output atteso: flag: valore intero 0 (indicante l'uscita dal menu principale)

Caso di test 4:

Descrizione: Uscita dal menu principale.
Input: file_menu: percorso del file contenente le opzioni del menu (file di testo) Output atteso: flag: valore intero 0 (indicante l'uscita dal menu principale)

Caso di test 5:

Descrizione: File del menu non valido o mancante.
Input:
file_menu: percorso del file contenente le opzioni del menu (file di testo)
Output atteso:
La funzione gestisce correttamente la mancanza o l'errore del file.

Caso di test 6:

Descrizione: Selezione di una scelta non valida.
Input:
file_menu: percorso del file contenente le opzioni del menu (file di testo)
scelta: valore intero diverso da SCELTA_INIZIO, SCELTA_SALVA, SCELTA_ABBANDONA e SCELTA_ESCI
Output atteso:
Richiesta di inserire nuovamente la scelta del menu.

Funzione gestire_partita_in_corso:

Caso di test 1:

Descrizione: Scelta di lanciare i dadi per continuare la partita.
Input: partita: partita attualmente in corso (di tipo Record competizione_oca) scelta_menu_g: carattere 'l'
Output atteso: Nessun output atteso specifico. La funzione dovrebbe procedere con il lancio dei dadi e la continuazione della partita.

Caso di test 2:

Descrizione: Scelta di aprire il menu di pausa e selezionare l'opzione "Salva partita".
Input: partita: partita attualmente in corso (di tipo Record competizione_oca) scelta_menu_g: valore intero corrispondente a SCELTA_SALVA Output atteso: La funzione dovrebbe salvare la partita e restituire il valore "fine" pari a 0.

Caso di test 3:

Descrizione: Scelta di aprire il menu di pausa e selezionare l'opzione "Abbandona partita".
Input: partita: partita attualmente in corso (di tipo Record competizione_oca) scelta_menu_g: valore intero corrispondente a SCELTA_ABBANDONA Output atteso: La funzione dovrebbe impostare il valore "fine" a 1, indicando che la partita è stata abbandonata.

Caso di test 4:

Descrizione: Scelta di aprire il menu di pausa e selezionare l'opzione "Esci".
Input:
partita: partita attualmente in corso (di tipo Record competizione_oca) scelta_menu_g: valore intero corrispondente a SCELTA_ESCI Output atteso: La funzione dovrebbe impostare il valore "flag" a 0, indicando di ritornare al menu principale.

Caso di test 5:

Descrizione: File del menu di nuova partita non valido o mancante.
Input:
partita: partita attualmente in corso (di tipo Record competizione_oca) file_partita_in_corso: valore NULL o percorso non valido del file Output atteso: La funzione dovrebbe impostare il valore "fine" a ERRORE_FILE, indicando un errore nel recupero del file del menu

Funzione salvare_partita:

Caso di test 1:

Descrizione: Salvataggio di una partita valida nello slot corretto.
Input: partita_salvata: partita che si intende salvare nello slot scelto (di tipo Record competizione_oca) slot: numero intero compreso tra 1 e MAX_SLOT (inclusi) Output atteso: fine: valore intero diverso da ERRORE_FILE

Caso di test 2:

Descrizione: Tentativo di salvare una partita quando il numero di slot è negativo o zero.
Input: partita_salvata: partita che si intende salvare nello slot scelto (di tipo Record competizione_oca) slot: numero intero negativo o zero
Output atteso: Richiesta di inserire nuovamente la scelta dello slot.

Caso di test 3:
Descrizione: Tentativo di salvare una partita quando il numero di slot è maggiore di MAX_SLOT.
Input: partita_salvata: partita che si intende salvare nello slot scelto (di tipo Record competizione_oca)
slot: numero intero maggiore di MAX_SLOT
Output atteso: Richiesta di inserire nuovamente la scelta dello slot.

Caso di test 4:
Descrizione: Tentativo di salvare una partita quando il file di salvataggio corrispondente allo slot selezionato non esiste. Input: partita_salvata: partita che si intende salvare nello slot scelto (di tipo Record competizione_oca) slot: numero intero compreso tra 1 e MAX_SLOT (inclusi) Output atteso:
fine: valore intero diverso da ERRORE_FILE Osservazione:
Il file è creato dall'apertura in modalità scrittura.

CASI DI TEST MODULO – GESTIRE_FILE

Funzione leggere_file_di_testo:
Caso di test 1:
Input:

file_testo = "prova.txt" num_caratteri
= 5
Descrizione: Leggere un file di testo con 5 caratteri.
Output atteso: Una stringa di lunghezza 5 contenente i primi 5 caratteri del file di testo.

Caso di test 2:

Input:
file_testo = "prova.txt" num_caratteri
= 0
Descrizione: Leggere un file di testo con 0 caratteri.
Output atteso: Una stringa vuota (lunghezza 0).

Caso di test 3:

Input:
file_testo = "vuoto.txt" num_caratteri
= 2
Descrizione: Leggere un file di testo vuoto con 2 caratteri. Output atteso: Una stringa di lunghezza 1 contenente solo il carattere FINE_STRINGA ('\0').
Output effettivo: Una stringa di lunghezza 5 contenente caratteri casuali e in ultimo carattere FINE_STRINGA
Risoluzione: I controlli sull'input di lunghezza sono effettuati a priori della chiamata della funzione Caso di test 4:

Input:
file_testo = "prova_lunghezza.txt"
num_caratteri = 700
Descrizione: Leggere un file di testo lungo 700 caratteri
Output atteso: Una stringa di lunghezza 700 contenente i primi 700 caratteri del file di testo.

Funzione scrivere_file_di_testo:

Caso di test 1:

Input: file_testo =
"scrivere.txt" stringa =
"Prova testo"
Descrizione: Scrivere la stringa "Prova testo" sul file di testo.

Output atteso: Il file di testo "scrivere.txt" contiene la stringa "Prova testo".

Caso di test 2:

Input:
file_testo = "sovrascrivere.txt" stringa
= "Messaggio nuovo"
Descrizione: Scrivere la stringa "Messaggio nuovo" sul file di testo esistente, sovrascrivendo il contenuto precedente.
Output atteso: Il file di testo "sovrascrivere" contiene la stringa "Messaggio nuovo".
Output atteso: Il file di testo "sovrascrivere" contiene la stringa "Messaggio nuovo".

Caso di test 3:

Input:
file_testo = "caratteri_txt" stringa
= "???---0dc!£$"
Descrizione: Scrivere una stringa contenente caratteri speciali sul file di testo.
Output atteso: Il file di testo "caratteri.txt" contiene la stringa "???--0dc!£$"

Funzione calcolare_lunghezza_file_di_testo:

Caso di test 1:

Input:
file = "prova.txt" Output
atteso:
num_caratteri = 50.

Caso di test 2:

Input:
file = "vuoto.txt" Output
atteso: num_caratteri =
0.

Funzione ricercare_slot:

Caso di test 1:

Input: slot =
1

tipo_apertura = "rb"
Descrizione: Recuperare lo slot corrispondente alla prima riga del file dei percorsi dei binari utilizzando l'apertura in modalità di lettura.
Output atteso: Apre il file "slot_1.bin" in modalità di lettura e lo restituisce come output. Caso di test 2: Input: riga = 2 tipo_apertura = "r"
Descrizione: Recuperare lo slot corrispondente alla seconda riga del file dei percorsi dei binari utilizzando l'apertura in modalità r. Il numero di righe nel file è 4
Output atteso: Apre il file "slot_2.bin" in modalità di lettura e lo restituisce come output.

Funzione ricercare_il_menu:
Caso di test 1:
Input: riga = 5, tipo_apertura = "r"
Descrizione: Recuperare il menu corrispondente alla quinta riga del file dei percorsi dei menu utilizzando l'apertura in modalità di lettura.
Output atteso: Apre il file "menu_5.txt" in modalità di lettura e lo restituisce come output. Caso di test 2:
Input: riga = 1, tipo_apertura = "r"
Descrizione: Recuperare il menu corrispondente alla prima riga del file dei percorsi dei menu utilizzando l'apertura in modalità r. Il numero di righe nel file è 2
Output atteso: Apre il file "menu_1.txt" in modalità di lettura e lo restituisce come output.

Funzione spostare_a_riga:
Caso di test 1:
Input: file = "prova.txt" riga_attuale = 1
Descrizione: Posizionare il cursore del file "prova.txt" all'inizio del testo.
Output atteso: Il puntatore del file "prova.txt" viene spostato all'inizio del file.

Caso di test 2:

Input: file = "prova.txt" riga_attuale = 5 Descrizione: Spostare il cursore del file "prova.txt" alla quinta riga. Output atteso: Il puntatore del file "prova.txt" viene spostato alla quinta riga del testo.

Caso di test 3:

Input: file = "vuoto.txt" riga_attuale = 1

Descrizione: Spostarsi alla prima riga di un file di testo vuoto.

Output atteso: Nessun movimento, poiché il file "vuoto.txt" è vuoto. Funzione leggere_riga_di_file:

Caso di test 1:

Input: file = "prova.txt"

Descrizione: Leggere il contenuto della prima riga dal file di testo. Output atteso: La stringa "riga" contiene il contenuto della prima riga del file.

Caso di test 2:

Input: file = "prova.txt"

Descrizione: Leggere il contenuto della seconda riga dal file di testo.

Output atteso: La stringa "riga" contiene il contenuto della seconda riga del file.

Funzione leggere_riga_di_file:

Caso di test 1:

Input:

file = "prova.txt"

Descrizione: Leggere la prima riga del file di testo.

Output atteso: La stringa riga contiene il contenuto della prima riga del file.

Caso di test 2:

Input:

file = "prova.txt"

Descrizione: Leggere la seconda riga del file di testo.

Output atteso: La stringa riga contiene il contenuto della seconda riga del file.

Output effettivo: La stringa riga contiene il contenuto della prima riga del file.

Risoluzione: Lo spostamento della riga è effettuato da un'altra funzione

Caso di test 3:

Input: file = "vuoto.txt" (file
vuoto)
Descrizione: Leggere la riga di un file di testo vuoto.
Output atteso: La stringa riga è vuota.

Funzione calcolare_potenza:
Caso di test 1:
Input: base_numero = 3, esponente = 2
Descrizione: Calcolare la potenza di 3 elevato a 2. Output atteso: potenza = 8
Caso di test 2:
Input: base_numero = 0, esponente = 5
Descrizione: Calcolare la potenza di 0 elevato a 5.
Output atteso: potenza = 0 Caso di test 3:
Input: base_numero = -2, esponente = 3
Descrizione: Calcolare la potenza di -2 elevato a 3.
Output atteso: potenza = -8 Caso di test 4:
Input: base_numero = 4, esponente = 0
Descrizione: Calcolare la potenza di 4 elevato a 0.
Output atteso: potenza = 1 Caso
di test 5:
Input: base_numero = 1.5, esponente = 2
Descrizione: Calcolare la potenza di 1.5 elevato a 2.
Output atteso: potenza = 2.25

Funzione scegliere_opzione_di_menu:
Caso di test 1:
Input: file_menu = "menu_principale.txt", min_scelta = 1, max_scelta = 4
Descrizione: Stampare il menu principale e richiedere all'utente una scelta compresa tra 1 e 4.
Input da tastiera: 3
Output atteso: Restituire 3 come scelta.

Caso di test 2:

Input: file_menu = "menu_principale.txt", min_scelta = 1, max_scelta = 4
Descrizione: Stampare il menu principale e richiedere all'utente una scelta compresa tra 1 e 4.
Input da tastiera: 5
Output atteso: Ripetere la richiesta di inserimento della scelta.

Caso di test 3:

Input: file_menu = "menu_opzioni.txt", min_scelta = 0, max_scelta = 2
Descrizione: Stampare il menu delle opzioni e richiedere all'utente una scelta compresa tra 0 e 2.
Input da tastiera: 1
Output atteso: Restituire 1 come scelta.

Caso di test 4:

Input: file_menu = "menu_opzioni.txt", min_scelta = 0, max_scelta = 2
Descrizione: Stampare il menu delle opzioni e richiedere all'utente una scelta compresa tra 0 e 2.
Input da tastiera: -1 Output atteso: Ripetere la richiesta di inserimento della scelta. Caso di test 5:

Input: file_menu = "menu_principale.txt", min_scelta = 1, max_scelta = 3
Descrizione: Stampare il menu principale e richiedere all'utente una scelta compresa tra 1 e 3. Input da tastiera: 'b'
Output atteso: Ripetere la richiesta di inserimento della scelta.

Caso di test 6:

Input: file_menu = "menu_opzioni.txt", min_scelta = 0, max_scelta = 2
Descrizione: Stampare il menu delle opzioni e richiedere all'utente una scelta compresa tra 0 e 2.
Input da tastiera: 'y'
Output atteso: Ripetere la richiesta di inserimento della scelta.

CASI DI TEST MODULO – GESTIRE_CLASSIFICA

Funzione aggiornare_classifica:

Caso di test 1:

Input:

vincitore = "Jasmine" giocatore
= Record("Jasmine", 3) Risultato
atteso:
Creazione del file classifica, Jasmine prima classificata con 3 tiri. Caso di test 2:
Input:
vincitore = "Max", giocatore
= Record("Max", 5) Classifica
iniziale:
["Jasmine" (3), "Max" (5), "Alex" (9)] Risultato
atteso:
La funzione aggiorna il numero di tiri di "Max" a 5 e aggiorna la classifica nel file di testo.
Caso di test 3:
Input: vincitore = "Bull",
giocatore = Record("Bull", 7)
Classifica iniziale:
["Jasmine" (3), "Max" (5), "Bull" (7), "Alex" (9)] Risultato
atteso:
La funzione inserisce "Bull" come nuovo classificato, inserendolo al posto corretto.

Funzione scrivere_classifica_vuota :
Caso di test 1:
Input:
vincitore = Record("Jasmine ", 3),
lista_classificati = [] Risultato
atteso:
Viene creato il file della classifica e inserito il vincitore come primo elemento della classifica.
Caso di test 2:
Input:
vincitore = Record("Max", 5),
lista_classificati = ["Jasmine " (3), " Max " (5), "Alex" (9)] Risultato
atteso:

Non viene creato un nuovo file di classifica, si riscrive la classifica nel file di testo e viene restituito lista_classificati con il vincitore.

Caso di test 3:

Input:
vincitore = Record("Bull", 7),
lista_classificati = [] Risultato
atteso:
La funzione non scrive nel file di classifica, restituisce lista_classificati vuota e si verifica un errore.

Caso di test 4:

Input:
vincitore = Record("Fabio", 0)
lista_classificati = [] Risultato
atteso:
Viene creato il file della classifica, inserito il vincitore con zero tiri come primo elemento della classifica.

Caso di test 5:

Input:
vincitore = Record("Dusan",19)
lista_classificati = ["Jasmine " (3), " Max " (5), "Alex" (9)] Risultato
atteso:
La funzione non crea un nuovo file di classifica, restituisce lista_classificati con il vincitore.

Funzione inserire_classifica:

Caso di test 1:

Input:
vincitore = Record("Alex", 2)
lista_classificati = ["Fabio" (5) "Niga" (6), "Dave" (8)] Risultato
atteso:
Inserisce il vincitore come primo classificato nella lista nella posizione corretta.

Caso di test 2:

Input:
vincitore = Record("Luca", 9)
lista_classificati = ["Fabio" (5) "Niga" (6), "Dave" (8), "Steve" (9)] Risultato atteso:
La funzione inserisce il vincitore nella posizione immediatamente successiva al classificato con lo stesso numero di tiri.

Caso di test 3:

Input:
vincitore = Record("Franco", 12)
lista_classificati = ["Fabio" (5) "Niga" (6), "Dave" (8), "Dave" (9)] Risultato atteso:
La funzione non modifica la lista_classificati perché il vincitore ha un numero di tiri più alto rispetto agli altri giocatori classificati.

Caso di test 4:

Input:
vincitore = Record("Jasmine", 3)
lista_classificati = [] Risultato atteso:
La funzione inserisce il vincitore come unico classificato nella lista vuota.

Funzione aggiornare_classificato:

Caso di test 1:

Input:
pos_classificato = 2, vincitore = Record("Nico", 5)
lista_classificati = ["Jasmine" (6), "Giuseppe" (7), "Alice" (8), "Davide" (10)]
Risultato atteso:
La funzione confronta il numero di tiri del secondo classificato con il numero di tiri del vincitore. Il vincitore è inserito nella lista_classificati nella posizione corretta perché ha un numero inferiori di numeri dei lanci.

Caso di test 2:

Input:

pos_classificato = 2, vincitore =
Record("Emiliano", 7)`,
lista_classificati = ["Jasmine" (6), "Giuseppe" (7), "Alice" (8), "Davide" (10)]
Risultato atteso:
La funzione confronta il numero di tiri del secondo classificato con il numero di tiri del vincitore. Siccome il numero dei tiri è uguale non viene aggiornata la lista_classifica.

Caso di test 3:

Input:
pos_classificato = 4, vincitore =
Record("Arianna", 9)`,
lista_classificati = ["Jasmine" (6), "Giuseppe" (7), "Alice" (8), "Davide" (10)]
Risultato atteso:
La funzione confronta il numero di tiri del quarto classificato con il numero di tiri del vincitore. Siccome il numero dei tiri è inferiore, il vincitore viene inserito nella lista_classifica nella posizione corretta.

Caso di test 4:

Input:
pos_classificato = 0, vincitore
= Record("Alice", 3)`,
lista_classificati = ["Jasmine" (6), "Giuseppe" (7), "Alice" (8), "Davide" (10)]
Risultato atteso:
La funzione confronta il numero di tiri del classificato in posizione 0 con il numero di tiri del vincitore. Siccome è il vincitore stesso ad essere nella posizione 0, non viene aggiornata la lista_classifica.

Caso di test 5:

Input:
pos_classificato = NO_RICERCA,
vincitore = Record("Jasmine", 5)
lista_classificati = []` Risultato
atteso:
La funzione non effettua alcuna modifica alla lista poiché non è presente alcun classificato. La lista_classificati non viene modificata.

Funzione convertire_giocatore:

Caso di test 1:

Input:
giocatore = Record("Alessia", 3)` Risultato
atteso:
La funzione legge il nome del giocatore e lo inserisce nel campo "nome_classificato" del record classificato; legge il numero di tiri del giocatore e lo inserisce nel campo "lanci " nel record classificato.

Caso di test 2:

Input:
giocatore = Record("", 5)
Risultato atteso:
La funzione non legge il nome del giocatore poiché è vuoto. Inserisce il carattere di fine stringa nel campo "nome_classificato" del nuovo classificato; legge il numero di tiri del giocatore e lo inserisce nel campo "lanci" del record classificato.

Caso di test 3:

Input:
giocatore = Record("A", 8)` Risultato
atteso:
La funzione legge il nome del giocatore "A" e lo inserisce nel campo "nome_classificato" del record classificato; legge il numero di tiri del giocatore e lo inserisce nel campo "lanci " del record classificato.

Funzione ricercare_classificati:

Caso di test 1:

Input: file_classifica: ["Jasmine", 4], ["Oshimen", 7],
["Bob", 3] Risultato atteso:
La funzione legge il file_classifica e li inserisce nell'array lista_classificati nella posizione corretta.

Caso di test 2:

Input: file_classifica:
vuoto Risultato
atteso:
La funzione non legge nulla dal file perché è vuoto.
Restituisce la lista_classificati vuota.

Caso di test 3:

Input:
file_classifica: ["Simba", 5] Risultato
atteso:
La funzione legge il file_classifica e lo inserisce nella lista_classificati.
Restituisce lista_classificati con il l'unico classificato.

Funzione confrontare_lista:

Caso di test 1:

Input: lista_classifica: ["Shiva", 4], ["Sfera", 6],
["Charlie", 3] vincitore: "Sfera" Risultato atteso:
La funzione confronta il nome del vincitore con i nomi presenti nella lista dei classificati. Trova una corrispondenza con il nome nella posizione 2 della lista dei classificati. Restituisce `pos_corrispondenza` con il valore 2.

Caso di test 2:

Input: lista_classifica: ["Shiva", 4], ["Sfera", 6],
["Charlie", 3] vincitore: "Simba" Risultato atteso:
La funzione confronta il nome del vincitore con i nomi presenti nella lista_classificati. Non trova corrispondenze e restituisce pos_corrispondenza con il valore `NO_RICERCA`.

Caso di test 3:

Input: lista_classifica:
vuota vincitore:
"Alice"
Risultato atteso:
Siccome la lista_classificati è vuota la funzione non trova corrispondenze.
Restituisce pos_corrispondenza con il valore `NO_RICERCA`.

Caso di test 4:

Input: lista_classifica: ["Shiva", 4], ["Sfera", 6], ["Charlie", 3], ["Leo", 8], vincitore: "Leo"
Risultato atteso:
La funzione confronta il nome del vincitore con i nomi presenti nella lista dei classificati. Trova una corrispondenza con il nome nella posizione 4 della lista dei classificati. Restituisce `pos_corrispondenza` con il valore 4.

Caso di test 5:

Input: lista_classifica: vuota vincitore: ("")
Risultato atteso:
Siccome la lista_classificati è vuota e anche il vincitore è vuoto la funzione restituisce pos_corrispondenza con il valore `NO_RICERCA`.

Funzione confrontare_numero_tiri:

Caso di test 1:

Input: lista_classifica: vuota vincitore` con numero di tiri 7
Risultato atteso:
Siccome la lista_classifica è vuota, la funzione restituisce `pos_corrispondenza` con il valore `NO_RICERCA`.

Caso di test 2:

Input: lista_classifica:["Alice", 5] vincitore` con numero di tiri 5 Risultato atteso:
La funzione confronta il numero di tiri del vincitore con il numero di tiri dei classificati presenti nella lista. Trova una corrispondenza e restituisce `pos_corrispondenza` con il valore 1.

Caso di test 3:

Input: lista_classifica: ["Alice", 3] vincitore` con numero di tiri 5 Risultato atteso:

La funzione confronta il numero di tiri del vincitore con il numero di tiri dei classificati presenti nella lista. Siccome non trova corrispondenze, restituisce `pos_corrispondenza` con il valore `NO_RICERCA`.

Caso di test 4:

Input: lista_classifica: ["Ghali", 5], ["TaxiB", 7], ["Tony", 4], ["Davide", 6] vincitore` con numero di tiri 5 Risultato atteso:
La funzione confronta il numero di tiri del vincitore con i numeri di tiri presenti nella lista dei classificati. Trova una corrispondenza in cui il numero di tiri del classificato è maggiore del vincitore nella posizione 2 della lista dei classificati. Restituisce `pos_corrispondenza` con il valore 2.

Funzione scrivere_classifica:

Caso di test 1:

Input:
file_classifica: vuoto
lista_classifica: vuota Risultato atteso:
La funzione non scrive sul file della classifica e restituisce file_classifica senza modifiche.

Caso di test 2:

Input:
file_classifica` contenente dei record di classificati
lista_classifica: ["Simba", 2] Risultato atteso:
La funzione scrive la lista_classifica sul file della classifica e restituisce file_classifica aggiornato.

Caso di test 3:

Input:
file_classifica contenente dei record di classificati
lista_classifica = ["Paky", 4], ["Shiva", 5], ["Charlie", 6] Risultato atteso:
La funzione scrive la lista_classifica sul file della classifica in sequenza e restituisce `file_classifica` aggiornato.

Caso di test 4:

Input: file_classifica:
vuoto lista_classifica:
["Alice", 5]
Risultato atteso: La funzione scrive il record del classificato ["Alice", 5] sul file della classifica e restituisce `file_classifica` con il nuovo record.

Caso di test 5:

Input: file_classifica` contenente dei record di
classificati lista_classifica: vuota Risultato atteso:
La funzione cancella tutti i record presenti nel file della classifica, lasciando il file vuoto, e restituisce file_classifica senza modifiche.

Funzione inserire_elemento:

Caso di test 1:

Input:
lista_classifica: vuota
classificato: ["Shiva", 3]
pos_inserimento: 0 Risultato
atteso:
lista_classificati diventa ["Shiva", 3] come unico elemento.

Caso di test 2:

Input:
lista_classifica: ["Shiva", 3], ["Evelyn", 4], ["Paky", 5]
classificato: ["Davide", 2] pos_inserimento: 1
Risultato atteso:
lista_classificati [“Davide”, 2], ["Shiva", 3], ["Evelyn", 4] Caso di test 3:

Input: lista_classifica: ["Keys", 4],
[FINE_LISTA] classificato: ["Max", 3]
pos_inserimento:1 Risultato atteso:
lista_classificati ["Keys", 4], ["Max", 3], [FINE_LISTA].

Caso di test 4:

Input: lista_classificati: ["Shiva", 4], ["Max”, 3],
[FINE_LISTA] classificato: ["Giulia", 6]
pos_inserimento: 0

Risultato atteso: `lista_classificati` diventa ["Giulia", 6], ["Shiva", 4], ["Max", 3].

Funzione scrivere_classifica_testo:

Caso di test 1:

Input:
lista_classificati: vuota Risultato
atteso:
flag = 1 e il file di testo rimane vuoto.

Caso di test 2:

Input:
lista_classificati: ["Shiva", 7], ["Rondo", 4], ["Simba", 5], [FINE_LISTA]
Risultato atteso:
flag = 1 e il file di testo contiene il seguente contenuto:

!!! CLASSIFICA VINCITORI !!!

Pos Nome Lanci

1 Shiva 7

2 Rondo 4 3 Simba 5 Caso di test 3:

Input:
lista_classificati:["Shiva", 7], ["Rondo", 4], [FINE_LISTA], [FINE_LISTA]
Risultato atteso:
flag = 1 e il file di testo contiene il seguente contenuto:

!!! CLASSIFICA MIGLIORI OCHE !!!

Pos Nome N.Tiri

1 Shiva 7

2 Rondo 4

Caso di test 4:

Input: lista_classificati: ["Shiva", 7], ["Rondo", 4], ["Simba", 5], [FINE_LISTA] file_classifica_testo: restituisce `ERRORE_FILE` Risultato atteso:
flag = 0 e il file di testo rimane vuoto.

Caso di test 5:

Input:
lista_classificati: vuota
file_classifica_testo: restituisce 0 Risultato
atteso:
flag = 0 e il file di testo rimane vuoto.

Funzione aggiornamento_classifica:

Caso di test 1:

Input:
vincitore con numero di tiri = 3 Risultato
atteso:
flag = 0

Caso di test 2:

Input:
vincitore con numero di tiri = PARTITA_INTERROTTA Risultato
atteso:
flag != 0 Caso
di test 3:

Input:
vincitore con numero di tiri = ERRORE_FILE
Risultato atteso: flag = ERRORE_FILE.

Caso di test 4:

Input:
vincitore con numero di tiri = 6
flag: ERRORE_FILE Risultato
atteso: flag = ERRORE_FILE.

Caso di test 5:

Input:
vincitore con numero di tiri = 5
flag != ERRORE_FILE Risultato
atteso:
flag = 0.

4. Presentazione della soluzione

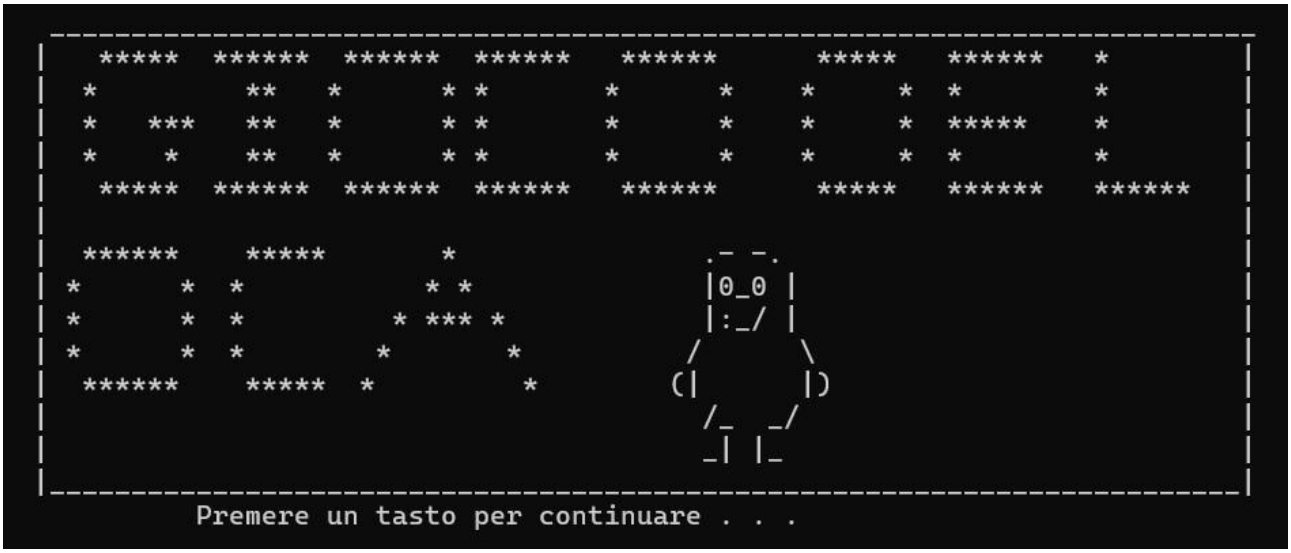
4.1 Funzionalità Fondamentali

La sezione della documentazione che illustra le funzionalità del gioco attraverso degli screenshot dell'eseguibile (exe) è denominata "Funzionalità Fondamentali". Questa sezione fornisce una visualizzazione visiva delle diverse schermate e delle interfacce utente del gioco per consentire la comprensione dell'aspetto e la navigazione dell'applicazione.

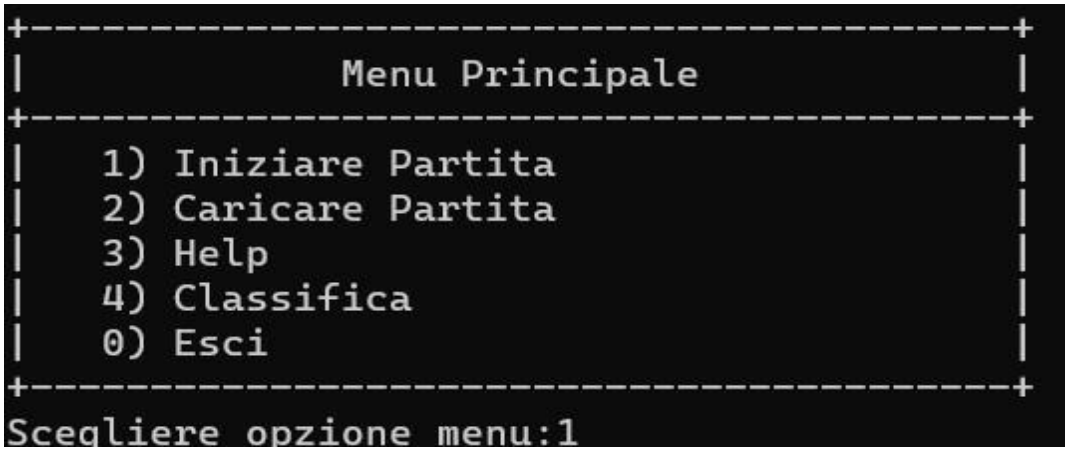
La galleria di screenshots del gioco include una serie di immagini che rappresentano le schermate principali e le funzionalità chiave del gioco. Questi screenshot catturano le interfacce utente pertinenti, come il menu principale, le opzioni di gioco, il tabellone di gioco, i messaggi di sistema e altre schermate rilevanti per l'esperienza di gioco.

Ogni screenshot nella galleria è accompagnato da una breve descrizione che spiega l'obiettivo e la funzionalità rappresentata nell'immagine. Queste descrizioni forniscono informazioni aggiuntive per aiutare a comprendere le azioni che possono compiere o le informazioni che possono visualizzare in quella specifica schermata.

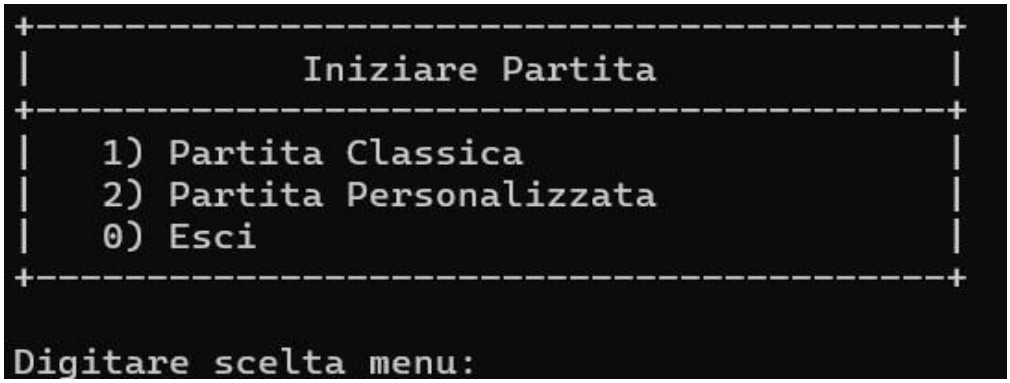
- Introduzione all'interfaccia iniziale



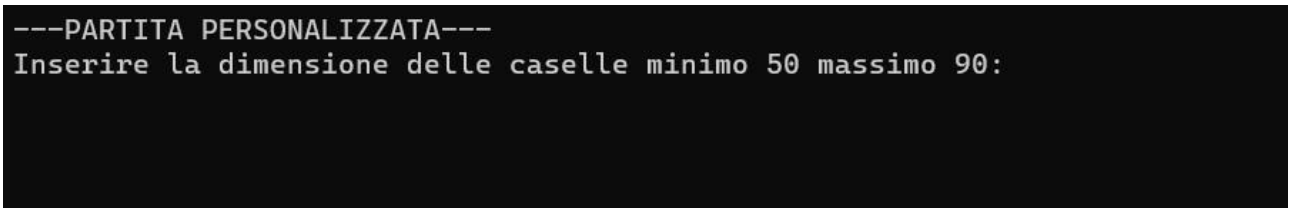
- Introduzione al menu principale



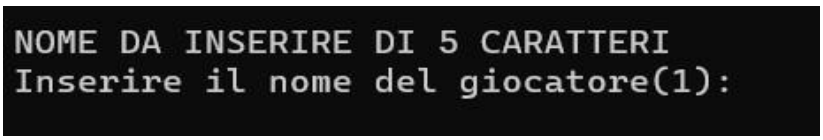
- Scelta “iniziare partita”, visualizzazione dell’interfaccia annessa :



- Interfaccia con la scrittura della dimensione del tabellone della partita personalizzata :



- Interfaccia Partita Personalizzata , con richiesta del nome dei giocatori :



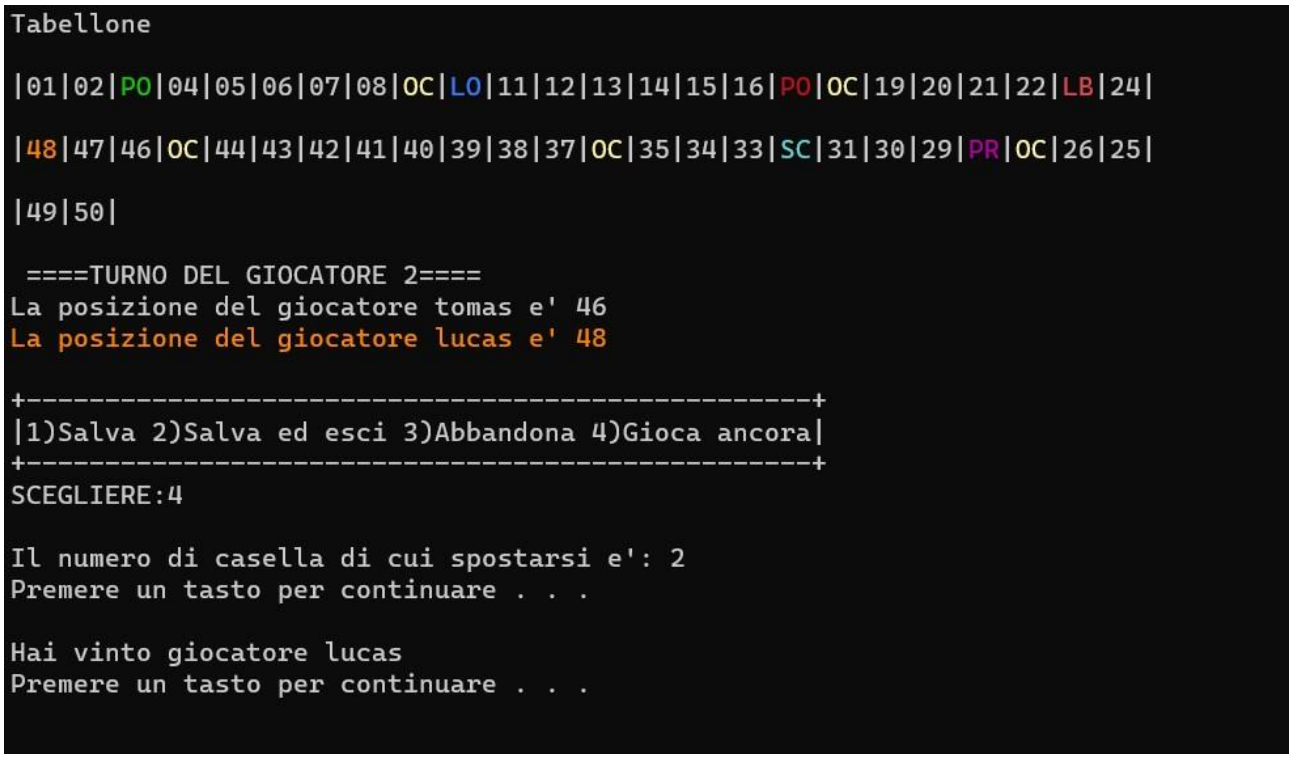
- Partita iniziata :

```
Tabellone

|01|02|PO|04|05|06|07|08|OC|LO|11|12|13|14|15|16|PO|OC|19|20|21|22|LB|24|

|48|47|46|OC|44|43|42|41|40|39|38|37|OC|35|34|33|SC|31|30|29|PR|OC|26|25|

|49|50|

 ====TURNO DEL GIOCATORE 2====
La posizione del giocatore tomas e' 0
La posizione del giocatore lucas e' 0

+-----------------------------------------------+
|1)Salva 2)Salva ed esci 3)Abbandona 4)Gioca ancora|
+-----------------------------------------------+
SCEGLIERE:
```

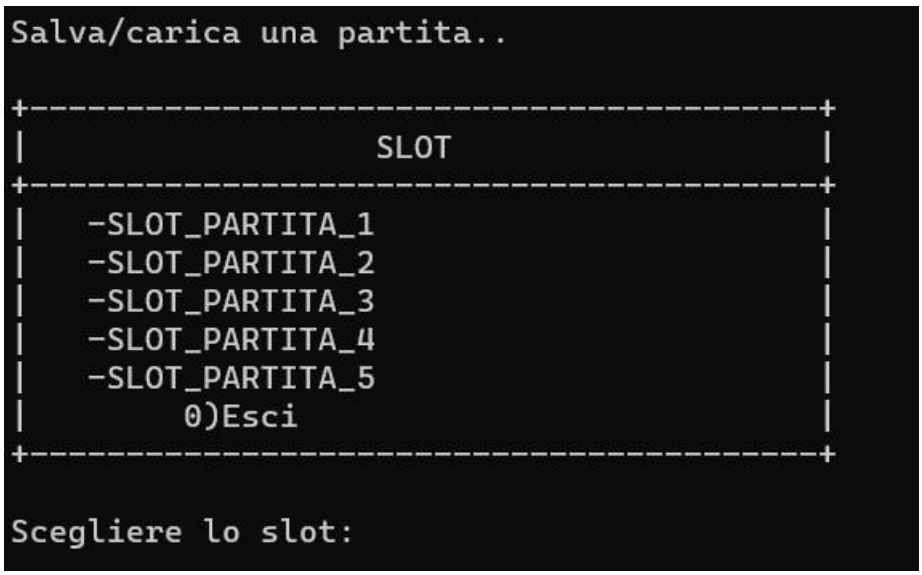
- Partita con ritrovamento delle caselle speciali :

```
Tabellone

|01|02|PO|04|05|06|07|08|OC|LO|11|12|13|14|15|16|PO|OC|19|20|21|22|LB|24|

|48|47|46|OC|44|43|42|41|40|39|38|37|OC|35|34|33|SC|31|30|29|PR|OC|26|25|

|49|50|

 ====TURNO DEL GIOCATORE 2====
La posizione del giocatore tomas e' 6
La posizione del giocatore lucas e' 18


Sei finito su un oca
Avanza ancora
Premere un tasto per continuare . . .
```

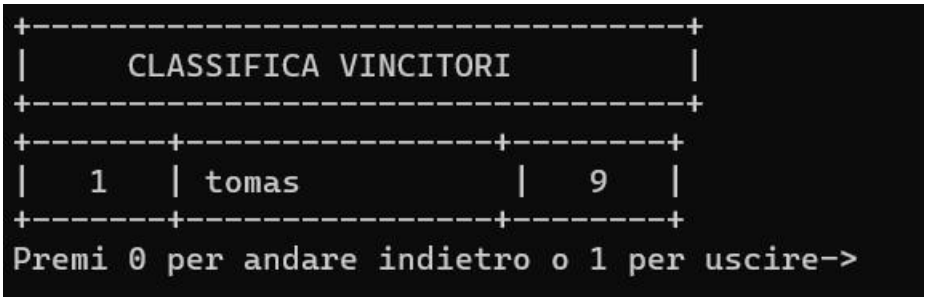
- Partita finita con il raggiungimento della vittoria di un giocatore :



- Introduzione interfaccia caricamento della partita da uno dei cinque slot :



- Introduzione interfaccia Classifica dei giocatori che hanno vinto la partita con il numero dei lanci :



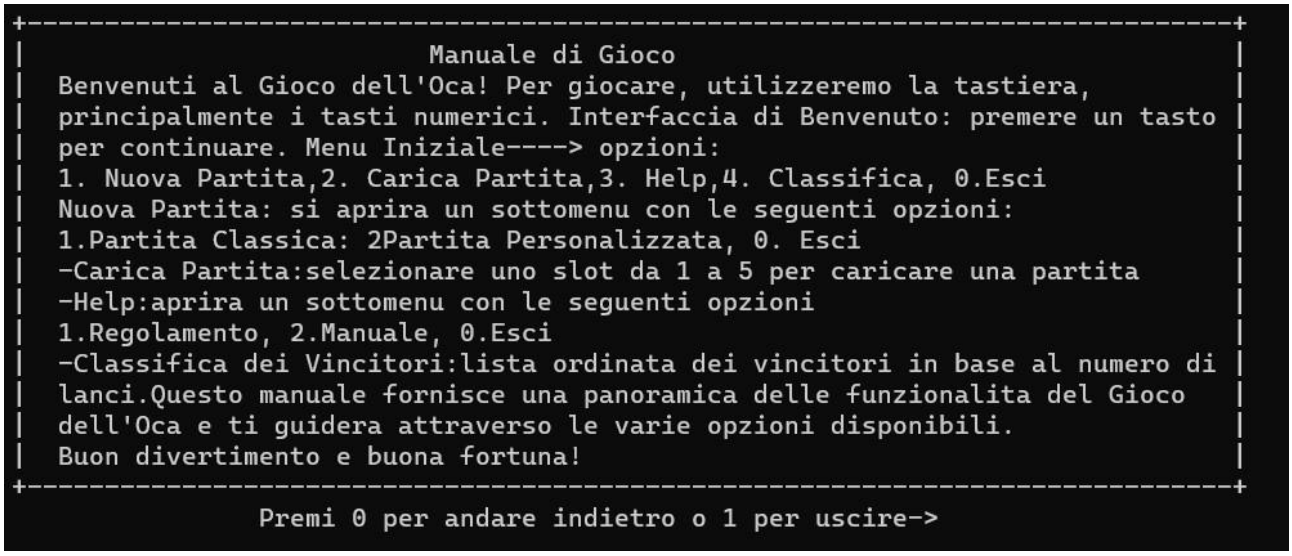
- Visualizzazione interfaccia menu aiuto :

```
+-----------------------------------------+
|                 Aiuto                   |
+-----------------------------------------+
|   1) Regolamento                        |
|   2) Manuale                            |
|   0) Esci                               |
+-----------------------------------------+

Scegliere opzione menu:
```

- Visualizzazione del regolamento del gioco dell'oca :

```
+-------------------------------------------------------------------------------+
|                      Regolamento del Gioco dell'Oca                           |
|                                                                               |
| Il gioco procede nel senso orario. Il primo giocatore lancia i dadi e sposta  |
| la sua pedina in avanti di un numero di caselle corrispondente alla somma     |
| ottenuta. Il gioco inizia dalla casella numero 1. Il numero di giocatori puo  |
| variare da 2 a 4. Lungo il percorso ci sono caselle speciali con effetti      |
| particolari:                                                                  |
| Le caselle "oca" permettono al giocatore di avanzare di un numero di caselle  |
| pari al risultato del lancio dei dadi. Solitamente sono posizionate ogni nove |
| caselle normali. La casella "ponte" (casella 6) richiede al giocatore di      |
| pagare una posta e di ripetere il suo movimento come se fosse su una casella  |
| "oca". La casella "locanda" (casella 19) obbliga il giocatore a fermarsi per  |
| tre turni consecutivi. Il "pozzo" (casella 31) e la "prigione" (casella 52)   |
| bloccano il giocatore finche un'altra pedina raggiunge la stessa casella e    |
| lo sostituisce.Il "labirinto" (casella 42) riporta il giocatore alla casella  |
|33.Lo "scheletro" (casella 58) fa retrocedere il giocatore fino alla casella 1.|
| La casella finale deve essere raggiunta con un lancio di dadi esatto.         |
| Se si supera il numero di caselle necessarie, il giocatore retrocede di       |
|un numero di caselle corrispondente all'eccesso.Il gioco continua fino a quando|
| un giocatore raggiunge la casella finale. Il vincitore e il primo giocatore   |
| a completare il percorso esattamente.                                         |
+-------------------------------------------------------------------------------+
                            Premi 0 per andare indietro o 1 per uscire->
```

- Visualizzazione del manuale del gioco dell'oca


Manuale di Gioco
Benvenuti al Gioco dell'Oca! Per giocare, utilizzeremo la tastiera, principalmente i tasti numerici. Interfaccia di Benvenuto: premere un tasto per continuare. Menu Iniziale----> opzioni:
1. Nuova Partita,2. Carica Partita,3. Help,4. Classifica, 0.Esci
Nuova Partita: si aprira un sottomenu con le seguenti opzioni:
1.Partita Classica: 2Partita Personalizzata, 0. Esci
-Carica Partita:selezionare uno slot da 1 a 5 per caricare una partita
-Help:aprira un sottomenu con le seguenti opzioni
1.Regolamento, 2.Manuale, 0.Esci
-Classifica dei Vincitori:lista ordinata dei vincitori in base al numero di lanci.Questo manuale fornisce una panoramica delle funzionalita del Gioco dell'Oca e ti guidera attraverso le varie opzioni disponibili.
Buon divertimento e buona fortuna!
Premi 0 per andare indietro o 1 per uscire->