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.