Casi di test avventura testuale **Gruppo 11**

1 – leggere_nome_giocatore

Nome	Descrizione	Tipo	Vincoli
nome_giocatore	Nome del giocatore che l'utente ha immesso	array 1 dimensione di caratteri (statico)	dimensione=10

Nome	Descrizione	Tipo	Vincoli
esito	flag che indica se nome_giocatore immesso dall'utente e' corretto	booleano	vero=corretto falso=errato

Input da testare	Output Funzione	Output desiderato?	Commento
(stringa vuota)	esito := falso	si	CORRETTO
▸	esito := falso	si	CORRETTO
stringaconpiudi10caratteri	esito := falso	si	CORRETTO
test 12	esito := falso	si	CORRETTO
test	esito := vero	si	CORRETTO
1test2444	esito := vero	si	CORRETTO
gianni ciao	esito := falso	si	CORRETTO

2 – eseguire_scanner

Nome	Descrizione	Tipo	Vincoli
comando_utente	Comando immesso dall'utente	array 1 dimensione di caratteri (statico)	dimensione=64

Nome	Descrizione	Tipo	Vincoli
regex_match	flag che indica se una o più parole del linguaggio descritto dall'espressione regolare [^a-zA-Z][^a-zA-Z]* e' sottostringa del comando	booleano	vero=c'è un match falso= non c'è un match

Nome [DATI LAVORO]	Descrizione	Tipo	Vincoli
verbo_comando	Verbo estratto da comando_utente	array 1 dimensione di caratteri (statico)	dimensione=64
argomento_ comando	Argomento estratto da comando_utente	array 1 dimensione di caratteri (statico)	dimensione=64

Poiche' la funzione eseguire_scanner modifica i dati di lavoro verbo_comando e argomento_comando che saranno l'input della funzione eseguire_parser, facciamo un controllo anche su quelli.

Input da testare	Output Funzione	Output desiderato?	Commento
1243	regex_match := vero	si	CORRETTO
!£%%	regex_match := vero	si	CORRETTO
(stringa vuota)	regex_match := falso	si	CORRETTO
guarda	regex_match := falso verbo_comando := "guarda" argomento_comando := ""	si	CORRETTO
APRI carovana	regex_match := falso verbo_comando := "apri" argomento_comando := "carovana"	si	CORRETTO
prendere MELA	regex_match := falso verbo_comando := "prendere" argomento_comando := "mela"	si	CORRETTO
PREN\\ER\\E M11111LA	regex_match := vero	si	CORRETTO
ucCIDERE tesT	regex_match := falso verbo_comando := "uccidere" argomento_comando := "test"	si	CORRETTO

3 – eseguire_parser

Nome	Descrizione	Tipo	Vincoli
verbo_comando	Verbo estratto dal comando immesso dall'utente	array 1 dimensione caratteri (statico)	dimensione=64
argomento_comando	Argomento estratto dal comando immesso dall'utente	array 1 dimensione caratteri (statico)	dimensione=64

Nome	Descrizione	Tipo	Vincoli
regex_match	flag che indica se una o più parole del linguaggio descritto dalle espressione regolari memorizzate nella struttura dati array di tipo comando sono sottostringhe di verbo_comando	booleano	vero=c'è un match falso= non c'è un match

Input da testare **verbo_comando** **argomento_comando**	**Output Funzione**	**Output desiderato?**	**Commento**
Esempio di stanza dove e' stata testata la funzione: stanza 18. Oggetti: carovana (attributi i,a), mela (attributi i,p) (presente negli oggetti contenuti della stanza)			
"scendi" "stqlla"	regex_match := vero (significa che il comando "scendi" esiste) *Non vedo nessun stqlla nella stanza*	si	CORRETTO
"prendi" "carovana"	regex_match := vero (significa che il comando "prendi" esiste) *Non posso fare questa azione su carovana*	si	CORRETTO
"scendi" "carovana"	regex_match := vero (significa che il comando "scendi" esiste) *Non posso fare questa azione su carovana*	si	CORRETTO
"soffoca" "carovana"	regex_match := vero (significa che il comando "soffoca" esiste) *Non posso fare questa azione su carovana*	si	CORRETTO
"scendi" ""	regex_match := vero (significa che il comando "scendi" esiste) *Su quale oggetto devo fare l'azione?*	si	CORRETTO
"blalblalal" "test test g "	regex_match := falso (significa che il comando "blalblalal" non esiste) *Non capisco cosa vuoi fare*	si	CORRETTO
"equipaggia" "carro armato"	regex_match := vero (significa che il comando "equipaggia" esiste) *Non vedo nessun carro armato nell'inventario*	si	CORRETTO

“salta” “”	regex_match := vero (significa che il comando “salta” esiste) *Si esegue la funzione eseguire_salta*	si	CORRETTO
“salta” “bla bla”	regex_match := vero (significa che il comando “salta” esiste) *Devo solo saltare. Digita ‘salta’*	si	CORRETTO
PRIMO CASO DI TEST SUL COMANDO “prendi mela”: Prima di eseguire il parser con input “prendi” e “mela”, si esegue il parser con input “apri” e “carovana”			
“apri” “carovana”	regex_match := vero (significa che il comando “apri” esiste) *Si rendono visibili gli oggetti contenuti della stanza.*	si	CORRETTO
“prendi” “mela”	regex_match := vero (significa che il comando “prendi” esiste) *Preso*	si	CORRETTO
SECONDO CASO DI TEST SUL COMANDO “prendi mela”: Si esegue il parser con input “prendi” e “mela” senza aver prima eseguito il parser con input “apri” e “carovana”			
“prendi” “mela”	regex_match := vero (significa che il comando “prendi” esiste) *Non vedo nessun mela nella stanza*	si	CORRETTO

4 – avviare_sistema_combattimento

Nome	Descrizione	Tipo	Vincoli
protagonista	Protagonista del gioco che deve battersi contro il nemico	oggetto_personaggio	deve avere gli attributi salute e forza >0
oggetto_nemico	Nemico del protagonista	oggetto_personaggio	deve avere gli attributi salute e forza >0

Nome	Descrizione	Tipo	Vincoli
danno	Danno che protagonista subisce battendosi contro il nemico	intero	>= salute protagonista significa che protagonista perde, vince altrimenti

Poiche' la funzione avviare_sistema_combattimento basa il suo funzionamento su numeri casuali, nella colonna "Output Funzione" sono mostrati piu' output (separati dalla virgola)

Input da testare **1^ riga: salute/forza Protagonista** **2^ riga: salute/forza Nemico**	**Output Funzione**	**Output desiderato?**	**Commento**
28,9 40,8	9,9,7,11,6,10	si	CORRETTO
20,5 30,1	4,8,8,8,7,6,3,8,7	si	CORRETTO
30,5 30,1	9,9,10,6,5,8,7,6,7,5,9	si	CORRETTO
26,9 40,8	9,7,7,8,9,9,8,6,8	si	CORRETTO
10,2 40,8	16,12,11,14,11,12	si	CORRETTO
10,15 40,8	13,3,10,9,11,9,8	si	CORRETTO