RAZIONALE CASI DI TEST

Nome caso di test: **prenotazioni_test1**

Funzionalità testata: aggiungi_prenotazione(int codice_abbonamento, int codice_lezione)

1. Introduzione

Questo caso di test verifica il comportamento corretto della funzione aggiungi_prenotazione quando vengono forniti in input un codice abbonamento valido (cioè presente nel sistema e non scaduto) e un codice lezione associato a una lezione con posti disponibili. L'obiettivo è assicurarsi che la prenotazione venga eseguita con successo in condizioni ideali.

2. Obiettivi del test

Verificare che la funzione:

- Riconosca correttamente un abbonamento attivo e presente nella lista
- Verifichi la disponibilità di posti per la lezione selezionata
- Registri correttamente la prenotazione nel sistema
- Aggiorni coerentemente il numero di posti occupati della lezione

3. Criteri di selezione dei test

Il test è stato progettato secondo i seguenti criteri:

- Classe di equivalenza valida: l'abbonamento è attivo e la lezione ha disponibilità
- Percorso principale del flusso: si intende verificare il comportamento della funzione in assenza di errori
- Verifica dello stato finale del sistema: si confronta la disponibilità iniziale e finale della lezione per assicurare che sia diminuita di una unità

4. Strategia di test

- I parametri della funzione (codice abbonamento e codice lezione) vengono letti da un file di input con estensione .in
- La funzione viene eseguita in modalità silenziosa, ovvero senza interazione da tastiera o stampa a video in caso di success

- Il file di output generato (prenotazioni.txt) viene confrontato automaticamente con un file oracle (.out) di riferimento
- La verifica viene eseguita tramite uno script automatizzato che utilizza comandi di confronto come diff

5. Ambiente di test

- Sistema operativo: Linux Ubuntu (tramite WSL - Windows Subsystem for Linux)
- Compilatore: gcc
- **File coinvolti:**
- abbonamenti.txt: contiene l'abbonamento con codice 4
- lezioni.txt: contiene la lezione con codice 25
- prenotazioni.txt: inizialmente vuoto o con eventuali prenotazioni precedenti
- **File di confronto:**
    - oracle/prenotazioni_test1.txt.out
    - output/prenotazioni_test1.txt.out

6. Risorse necessarie

- File di configurazione con estensione .in
- File abbonamenti.txt, lezioni.txt e prenotazioni.txt coerenti e aggiornati
- Accesso in scrittura alla cartella output
- Script runner per l’automazione del test e confronto dei file di output

7. Rischi e mitigazioni

Rischio: il file abbonamenti.txt non contiene il codice specificato
Mitigazione: verificare manualmente che il codice 4 sia presente nel file

Rischio: differenze di formattazione tra file oracle e file di output
Mitigazione: utilizzare diff -b oppure applicare una normalizzazione delle righe prima del confronto

Rischio: codice lezione non valido o duplicato
Mitigazione: controllare che lezioni.txt contenga il codice corretto in forma univoca

8. Criteri di accettazione

Il test si considera superato se:

- La funzione stampa un messaggio di conferma della prenotazione
- Il file output/prenotazioni_test1.txt.out contiene le seguenti informazioni:
  - Prenotazione aggiunta con successo
  - Disponibilità iniziale: 20
  - Disponibilità finale: 19
  - Codice abbonamento utilizzato: 4
  - Codice lezione utilizzato: 25
- La differenza tra disponibilità iniziale e finale della lezione è esattamente pari a 1
- Nessun messaggio di errore viene stampato durante l'esecuzione

Nome caso di test: **prenotazioni_test_abbonamento_scaduto**
Funzionalità testata: aggiungi_prenotazione(int codice_abbonamento, int codice_lezione)

Introduzione

Questo caso di test verifica il comportamento della funzione aggiungi_prenotazione quando viene fornito in input un codice abbonamento **scaduto**, anche se il codice è corretto e presente nel file degli abbonamenti. L'obiettivo è assicurarsi che il sistema **blocchi la prenotazione** e **non alteri lo stato** della lezione o della lista prenotazioni.

1. Obiettivi del test

Verificare che la funzione:

- Riconosca correttamente un abbonamento scaduto.
- Impedisca la prenotazione per abbonamenti non più validi.
- Non alteri il numero di posti occupati nella lezione.
- Non registri nessuna nuova prenotazione nella lista.
- Non produca alcun output se eseguita in modalità test (entrambi i parametri ≠ -1).

2. Criteri di selezione dei test

Il test è stato progettato secondo i seguenti criteri:

- **Classe di equivalenza non valida**: abbonamento esistente ma scaduto.
- **Percorso alternativo**: il flusso si interrompe quando viene rilevata la scadenza dell'abbonamento.
- **Verifica dello stato del sistema**: si confrontano i file prenotazioni e lezioni prima e dopo il test per assicurarsi che siano **immutati**.

3. Strategia di test

- I dati di input (codice abbonamento e codice lezione) vengono letti da un file .in.
- La funzione viene eseguita in modalità silenziosa (entrambi i parametri sono != -1).
- Il file prenotazioni.txt non deve subire modifiche.
- Nessun output deve essere generato sullo schermo.
- I file output/ e oracle/ vengono confrontati tramite uno script automatico con diff.

4. Ambiente di test

- **Sistema operativo:** Linux Ubuntu (tramite WSL)
- **Compilatore:** gcc
- **File coinvolti:**
- abbonamenti.txt: contiene un abbonamento scaduto con **codice 6**
- lezioni.txt: contiene una lezione valida e disponibile con **codice 30**
- prenotazioni.txt: inizialmente vuoto o con prenotazioni precedenti
-
- **File di confronto:**
- oracle/prenotazioni_test_abbonamento_scaduto.txt.out
- output/prenotazioni_test_abbonamento_scaduto.txt.out

5. Risorse necessarie

- File .in contenente i codici abbonamento (6) e lezione (30)
- File abbonamenti.txt contenente un abbonamento **attivo ma con scadenza superata**
- File lezioni.txt con lezione disponibile
- Script di test runner
- Permessi in scrittura sulla directory output/

6. Rischi e mitigazioni

Rischio	**Mitigazione**
L'abbonamento non è effettivamente scaduto	Verificare data inizio e durata per garantire la scadenza
Codice abbonamento non presente	Controllare manualmente il file abbonamenti.txt
Output inatteso su schermo	Assicurarsi che la funzione operi in modalità test
Alterazione del file prenotazioni.txt	Confrontare l'hash o usare diff tra file prima e dopo

7. Criteri di accettazione

- Nessuna nuova riga deve essere aggiunta al file prenotazioni.txt.
- Il numero di posti occupati nella lezione non deve cambiare.
- Nessun messaggio di errore deve essere stampato (in modalità test).
- L'output output/prenotazioni_test_abbonamento_scaduto.txt.out deve essere **vuoto** o identico all'oracle.
- Lo stato del sistema (lezioni e prenotazioni) deve essere **invaria**

RAZIONALE CASO DI TEST

Nome caso di test: test3

Funzionalità testata: **reportDisciplineUltimoMese**(listP* lista_prenotazioni, listL* lista_lezioni, FILE* fout)

1. Introduzione

Questo caso di test verifica il comportamento corretto della funzione reportDisciplineUltimoMese, che stampa un report contenente la classifica delle discipline più frequentate nell'ultimo mese. L'obiettivo è assicurarsi che il report contenga dati corretti, ordinati e coerenti rispetto ai dati delle prenotazioni e delle lezioni fornite.

2. Obiettivi del test

Verificare che la funzione:

- Scorra correttamente la lista delle prenotazioni e consideri solo quelle riferite alle lezioni dell'ultimo mese
- Calcoli correttamente il numero di prenotazioni per ciascuna disciplina
- Ordini il report in modo decrescente rispetto al numero di prenotazioni
- Stampa il report correttamente su file di output
- Gestisca correttamente il caso in cui non siano presenti lezioni nell'ultimo mese

3. Criteri di selezione dei test

Il test è stato progettato secondo i seguenti criteri:

- Classe di equivalenza valida: prenotazioni e lezioni correttamente formattate e datate nell'ultimo mese
- Percorso principale del flusso: stampa del report in condizioni standard senza errori
- Verifica dell'ordinamento e accuratezza dei conteggi nel report

4. Strategia di test

- I dati di input (liste di prenotazioni e lezioni) vengono caricati da file test predefiniti contenuti nella cartella input
- La funzione reportDisciplineUltimoMese viene eseguita fornendo un file di output, in modo che il report venga salvato su file anziché stampato a schermo
- Il file di output generato viene confrontato automaticamente con un file oracle di riferimento (.out) mediante uno script di test automatizzato

5. Ambiente di test

• Sistema operativo: Linux Ubuntu (tramite WSL)

• Compilatore: gcc

• File coinvolti:

test3.txt: contiene prenotazioni distribuite su discipline diverse nell'ultimo mese

test3.txt: contiene lezioni associate alle discipline corrispondenti

• File di confronto:

oracle/oracle_test3.txt.out

output/output_test31.txt.out

6. Risorse necessarie

• File di input coerenti e aggiornati (prenotazioni e lezioni)
• Accesso in scrittura alla cartella output
• Script runner per automazione del test e confronto dei file di output

7. Rischi e mitigazioni

Rischio: dati di input non corretti o assenti nell'ultimo mese
Mitigazione: verificare manualmente che i file di input contengano dati validi e recenti
Rischio: differenze di formattazione tra oracle e output
Mitigazione: utilizzare diff -b o normalizzare le righe prima del confronto Rischio: errori di ordinamento o conteggio nel report
Mitigazione: controllare la logica di aggregazione e ordinamento durante la revisione del codice

8. Criteri di accettazione

Il test si considera superato se:
• Il file output/report_discipline_test1.txt.out contiene un elenco ordinato delle discipline con i relativi conteggi di prenotazioni, corrispondente al file oracle
• Il report riporta solo le discipline con prenotazioni nell'ultimo mese
• Il numero di prenotazioni per ciascuna disciplina è corretto e coerente con i dati di input
• In caso di assenza di prenotazioni nell'ultimo mese, il report riporta un messaggio chiaro di "Nessuna lezione svolta nell'ultimo mese"
• Nessun errore o messaggio anomalo viene generato durante l'esecuzione