Libera Università di Bolzano Bozen

Esame Financial Trading Algorithms

# Rules

* + Non è permesso comunicare con altre persone o tra studenti. I telefoni e ogni altro mezzo di comunicazione devono essere spenti. Non è permesso aprire alcun programma di comunicazione sul computer e ciò è considerato frode.
  + Sei responsabile della corretta copia dei tuoi file.

## Esercizio Smart Contracts

## Idea

Il tuo compito è scrivere un contratto per gestire le entrate in una sauna. Gli utenti acquistano il biglietto quando entrano e devono indicare per quanti secondi rimarranno all'interno, pagando un numero appropriato di wei. All'uscita, il biglietto viene controllato per verificare se il tempo è scaduto e, in caso affermativo, l'utente deve integrare il pagamento.

Scrivi un contratto in Solidity chiamato **sauna**.

**Strutture dati suggerite**

## L'unica struttura dati rilevante è l'orario di uscita di ciascuna persona.

## Step 1

## Scrivi un costruttore che imposti tutte le variabili, prestando attenzione affinché il tempo minimo e massimo sia positivo e che il prezzo massimo sia maggiore del prezzo minimo. Il prezzo per secondo può essere zero nel caso in cui l'ingresso sia gratuito.

## Step 2

## Aggiungi una funzione buy\_ticket. L'utente deve indicare per quanti secondi rimarrà all'interno. La funzione verifica che questo numero sia compreso tra il minimo e il massimo e che l'utente abbia pagato abbastanza wei. La funzione aggiorna l'orario di uscita al tempo corrente più il numero di secondi indicato.

## Step 3

Aggiungi una visualizzazione **check\_exit**, che è valida solo se il sistema ha già visto quell'utente in precedenza (usa exit\_time per determinarlo). Restituisce 0 se l'utente esce in tempo, altrimenti restituisce l'importo di wei extra che l'utente deve pagare. *Ad esempio, se il prezzo è 20 e l'utente deve uscire entro il timestamp  
1 000 000 e ora è tempo 1 000 040, la funzione restituisce 20\*40=800.*

**GIRA LA PAGINA 🡪**

## Step 4

## Aggiungi una funzione pay\_to\_exit, che è valida solo se il sistema ha già visto quell'utente in precedenza. Se l'utente sta uscendo in tempo o ha pagato abbastanza per compensare il tempo extra (puoi usare view check\_exit() per ottenere l'importo extra da pagare), restituisce true. Altrimenti, restituisce false.

## Step 5

## Fai deploy del contratto sulla blockchain di Sepolia, acquista un biglietto con un breve periodo, attendi che scada e usa pay\_to\_exit per uscire pagando l'importo extra.

## Step 6

## Aggiungi le funzioni get\_money e change\_price che possono essere utilizzate solo dal proprietario per ottenere tutti i soldi pagati fino a quel momento e per cambiare il prezzo per secondo.

## Step 7

Modifica le funzioni **buy\_ticket** in modo che se l'utente paga troppo, la funzione restituisca l'importo extra pagato. *Ad esempio, se il prezzo è 20 e l'utente paga 1000 per 30 secondi, l'importo previsto è 20\*30=600 e la funzione restituisce 400 all'utente.* Modifica la funzione **pay\_to\_exit** allo stesso modo.

## Salva e restituisci:

* un file di testo **sauna.txt** contenente il codice sorgente Solidity che hai creato e l'indirizzo del contratto Sepolia.