Simulazione Prog. avanzata - 01

Teoria

1. Spiega cosa sono gli smart pointers; di' quanti tipi di smart pointers conosci, puntualizzane le differenze e fornisci esempi di codice.
2. Spiega cosa si intende con *ereditarietà multipla* e come viene realizzata in `C++`; spiega inoltre cos'è il *diamond problem* e come viene approcciato in dettaglio, con esempi.
3. Spiega le eccezioni: cosa sono, come vengono propagate e come vengono utilizzate, con esempi.
4. Spiega cosa sono i thread, come li possiamo usare e perché sono utili; spiega il data race problem e come possiamo risolverlo.

Pratica

5. Supponi di avere un vettore di interi casuali positivi (inventalo); crea una funzione `booleanizzazione` che riceve in input il vettore e ne sostituisca ogni elemento con *0* oppure *1*, a seconda che il numero sia rispettivamente pari o dispari; deve essere utilizzata una lambda expression.
6. [NOTA: questo esercizio è un filo da ripensare]
   - Un Antropomorfo ha un certo numero di arti; un Umano ha un nome e un'età; un Megazord ha una *coolness* e un animale associato; un Power Ranger è un Umano e anche un Megazord.
   - Un Power Ranger tutta una sua lista di Ammiratori, e ognuno di questi Ammiratori ammira diversi Power Ranger.

   Modella queste relazioni in maniera che un main con un'istanza di Power Ranger e relativa lista di ammiratori compili (puoi supporre la classe Ammiratore già implementata).

   Nota: il secondo punto di questo esercizio mira a gestire la lettura dei riferimenti incrociati da parte del linker in un progetto multifile. L'argomento è stato trattato a lezione, ma non può essere richiesto in sede d'esame dal momento che il codice andrà copiato in una rich text area e il compilatore online utilizzabile non permette di gestire progetti multifile; per questi motivi, chi si trovasse ad avere poco tempo a disposizione per preparare l'esame potrebbe valutare di saltare questo esercizio. Questo naturalmente resta valido fino al cambio della modalità d'esame.
7. Crea una lambda expression `stampaBizzarra` che riceve in input una stringa e un intero *n* e stampa la stringa *n* volte; crea due thread a cui passi questa funzione con le stringe `"tanti"` e `"ionico"`; assicurati che non ci siano inconsistenze nelle stampe.
8. Considera una `unordered_map<int, int>`; scrivi una lampa expression che crei un vettore, una lista e un set di interi che contengano esattamente i valori della mappa suddetta nella posizione espressa dalla chiave; si gestiscano opportunemente le eccezioni nei casi in cui dovessero rendersi necessarie.