Esercitazione 2
Programmazione Avanzata ed Elementi di Ingegneria del Software
02/12/2022

1) Considerare una lista concatenata di record di esemplari di piante in possesso di un vivaio. Questi sono strutturati come:

```
typedef struct {
      char *szGenere;
      char *szSpecie;
      _Bool bFiore;
} rPianta;
```

Implementare, avvalendosi dello `STRATEGY` pattern, una funzione `massimo(lista, campo)` che, presa in ingresso una lista di record, restituisca il numero massimo di occorrenze di quel campo nel record. Per esempio, nella lista:

```
[{“Aloe”, “Vera”, true},
 {“Rosa”, “Canina”, true},
 {“Rosa”, “Gallica”, true},
 {“Cycas”, “Revoluta”, false}]
```

il confronto per “Fiore” dovrà restituire 3.

2) Considerare un gioco in cui due individui, *Risky* e *Careful*, si sfidano secondo le seguenti regole:

- All’inizio di ogni turno, *Risky* deve tirare una moneta e dichiarare un numero da 1 a 50; *Careful* può tirare un dado a sei facce e dichiarare un numero da 1 a 20.
- Se *Risky* ottiene “Testa” guadagna punti uguali al numero dichiarato ma, in caso contrario, ne perde il doppio.
- Se *Careful* ottiene un numero minore di 5, guadagna punti uguali al numero dichiarato ma, in caso contrario, ne perde altrettanti.
- Ogni giocatore scommette una sola volta per turno. Chi raggiunge per primo 500 vince.

Implementare la funzione `Play(Player *)` usando lo STRATEGY Pattern sia per *Risky* che per *Careful*.

3) Stimare la complessità computazionale della funzione `unknown()` riportata in seguito. Opzionalmente, indicare quale algoritmo implementa.

```
int unknown (int iN) {
    int iK = 1;
    int iT = 1;
    while (iK++ < iN) {
        iT *= iK;
    }
    return iT;
}
```