**Tehtävät 2 (Ennakointi & tekoälyn parametrien optimointi)**

1. Toteuta 4-gram -tekoäly kolikkopelillle (tekoäly arvaa pidätkö kolikkoa vasemmassa vai oikeassa kädessäsi). Testaa tekoälyä pelaamalla sitä vastaan riittävän pitkiä (esim. 100 kierroksen) otteluita. Onnistutko voittamaan yhtään ottelua, vaikka tiedät tekoälyn toimintaperiaatteen? (3 pistettä)

Toteutettu. Liitteenä palautuksessa nimellä *02\_01.zip*. En onnistunut voittamaan yhtään ottelua.

2. Esitä valistunut arvaus siitä, mikä on ihmisen optimaalinen strategia kolikkopelissä. Miksi ihmisen on käytännössä lähes mahdotonta pelata optimaalista strategiaa? Entä onko toteuttamasi 4-gram - tekoälyn strategia optimaalinen? Perustele. (1 piste)

Optimaalinen strategia KPS-pelissä on pelata jokaista vaihtoehtoa tasaisesti, kuitenkin pyrkien välttämään toistuvia kuvioita (pattern). Ihmisen on vaikea pitää lukua kunkin strategisen option pelatuista määristä.

Luomani 4-gram-tekoäly ei pelaa optimaalista strategiaa, vaan pyrkii maksimaalisesti hyödyntävään strategiaan, sillä se ei pelaa tasaisesti jokaista strategista optiota, vaan muokkaa strategiaansa kunkin botin käyttäytymisen mukaisesti.

3. Rakenna KPS-turnaukseen N-gram -perusteinen botti (valitse sopiva N itse). Toteuta bottisi MY\_BOT.h -tiedostoon ja testaa sitä turnauksessa, johon osallistuvat kaikki muut botit (opettajan + muiden opiskelijojiden), ks. roshambo\_kaikki\_botit.zip. Bottisi tulisi olla niin vahva, että se voittaa turnauksen (4 pistettä)

Botti liitteenä palautuksessa nimellä *MY\_BOT.h*.

4. Etsi funktion f(x) = x3 + x, maksimiarvo käyttäen Hill Climbing -algoritmia (x:n arvo väliltä [-1, 1]). (2 pistettä)

5. Ota tehtävän pohjaksi optimize-solution (Visual Studio). Käytettävissäsi on rajapinta kuvitteelliseen tekoälyyn, jonka toimintaa ohjaa kaksi parametria, attack\_range ja defense\_range. Rajapinnassa on kaksi funktiota:

// Set parameters that guide AI's decisions.

void set\_parameters(double attack\_range, double defense\_range);

// Test AI. Returns score that reflects AI's strenght, // the bigger the better.

double test\_AI();

set\_parameters() -funktiolla asetetaan halutut parametrien arvot. Tämän jälkeen test\_AI() testaa (oletettavasti automatisoidusti) tekoälyn vahvuuden näillä parametriasetuksilla. Tekoälyn vahvuus palautetaan kutsujalle paluuarvona (mitä suurempi, sitä parempi).

Optimoi parametrit käyttäen Hill Climbing -algoritmia. Tee toteutus optimize\_ai\_main.cpp - tiedostoon. (3 pistettä)