Proiect PA - Anul 2021
Echipa MEM

Membrii echipei:

- *Timpuriu Mircea*
- *Zaharov Evghenii*
- *Oprea Mihail*
- *Ghițan Răzvan*

ETAPA II

➔ *Instrucțiuni de compilare:*

Arhiva noastră a fost realizată în conformitate cu cerințele din enunț. De aceea, în rădăcina arhivei veți găsi un fișier Makefile care conține regulile ***build***, ***run***, ***clean***. Pentru a putea compila programul, este necesară compilarea surselor cu ajutorul comenzii ***make build***. După aceea, vom rula programul propriu-zis prin intermediul comenzii ***xboard -fcp "make run" -debug***.

Pentru realizarea acestui proiect, ne-am hotărât să utilizăm ***limbajul de programare Java*** în detrimentul limbajului C++, deoarece am considerat mult mai ușoară folosirea acestuia, mai ales datorită cursului de POO din cadrul semestrului I.

➔ *Detalii despre structura proiectului:*

Structura proiectului nostru include următoarele clase:

- ***Main*** - în care citim și prelucrăm comenzile de la xboard;
- ***Logic*** - în care păstrăm informații generale despre starea jocului (e.g. culoarea care mută, numărul de mișcări efectuate);
- ***Board*** - reprezentarea internă a tablei;
- ***Piece*** - clasă abstractă care reprezintă o piesă;
- ***Pawn*** - una din clasele care moștenește clasa Piece și implementează comportamentul specific unui pion;
- ***Rook***, ***Knight***, ***Bishop***, ***King***, ***Queen*** - restul de clase care moștenesc clasa Piece și care descriu/realizează mutările posibile ale respectivelor piese;
- ***Coordonate*** - clasă pentru reprezentarea coordonatei;
- **Move** - clasa care stochează informațiile despre o mutare;
- **En passant** - clasă care stochează coordonata la care se poate captura pionul care realizează en passant;
- **Ray** - clasă care permite parcurgerea boardului pe o linie și determinarea amenințării căsuței inițiale de către celelalte piese.

➔ *Abordarea algoritmică a etapei:*

În timpul receptionarii comenzilor primite la stdin de la xboard, noi le vom analiza și vom putea interpreta comanda sau simula mișcarea primită în structura internă a programului, acest lucru fiind realizat în complexitate O(1).

Pentru a realiza o mișcare (calculul unei mișcării și trimiterea ei la stdout), noi vom genera lista de piese care pot fi mutate, acest lucru fiind realizat printr-o parcurgere a unei liste de piese aflate pe tablă, căutare ce se realizează în complexitate O(n), n fiind numărul de piese de pe tablă.

În implementarea noastră, piesa care urmează a fi mutată este aleasă în mod aleator dintre piesele disponibile. Această selecție este realizată în O(1), deci nu influențează complexitatea algoritmului.

Parcurgerea board-ului în implementarea clasei Ray pentru a determina dacă o piesă se află în șah sau în pericolul de a intra în șah este realizata în O(n), n fiind dimensiunea tablei.

➔ *Surse de inspirație și responsabilitatea fiecărui membru:*

După ce toți membrii echipei s-au familiarizat cu interfața xboard în prima etapă, am decis să împărțim în mod egal sarcinile, care au constat în implementarea posibilităților de mutare a pieselor, implementarea cazurilor speciale de mutări (rocadă, en passant), și, mai ales, algoritmica și analizarea cazurilor în care suntem în șah, sau în care suntem în pericolul de a intra într-o poziție de șah, și acționarea în consecință în aceste cazuri.

➔ *Bibliografie:*

https://makefiletutorial.com/ - pentru Makefile.
https://docs.oracle.com/en/java/ - pentru probleme / detalii despre metode si clase.
https://www.gnu.org/software/xboard/engine-intf.html - pentru a înțelege mai bine algoritmica din spatele acestei etape.
https://en.wikipedia.org/wiki/En_passant - pentru a înțelege toate regulile en passant-ului.
https://en.wikipedia.org/wiki/Castling - pentru a afla detalii despre condițiile rocadelor.