بسمه تعالی

تکلیف اختیاری سری اول

موعد تحویل ۱۴۰۲/۰۹/۳۰

(۰.۵ نمره اضافه)

در این تکلیف قصد داریم با مفاهیم بیان شده در فصل هفتم بیشتر آشنا شویم. بدین منظور شما بایستی با بررسی کد پیوست شده، گراف CFG مربوط به آن را استخراج نموده و با محاسبه DU-Pathهای موجود حداقل ۳ عدد Test Case تولید نمایید. به عنوان خروجی بایستی یک PDF شامل، گراف به دست آمده، مراحل محاسبه DU-Pathها و همچنین اطلاعات دقیق مربوط به Test Caseها را گزارش نمایید.

موفق باشید

شعرباف

پیوست:

```
public List<Game> buildInitialRound(List<Participant> participants) {

    if (!Util.isPowerOfTwo(participants.size())) {
        throw new IllegalArgumentException("The number of participants is not a power of two");
    }

    List<Participant> remainingRankedParticipants = new ArrayList<>(participants);
    List<Game> initialRound = new ArrayList<>();

    int amountOfInitialGames = participants.size() / 2;
    for (int i = 0; i < amountOfInitialGames; i++) {
        Game game = new Game();
        Scanner scanner = new Scanner(System.in);
        while (!game.isGameOver()) {
            System.out.println(game.toString());
            int chosenint;
            for (chosenint = -10; chosenint >= Direction.values().length
                || chosenint < 0; chosenint = scanner.nextInt()) {
                int counter = 0;
                for (Direction d : Direction.values()) {
                        System.out.println(counter + ": " + d.name());
                        counter++;
                }
                Logger.log("Please chose a direction: ");
            }

        try {
                game.playRound(Direction.values()[chosenint]);
            } catch (ImpossibleActionException e) {
                Logger.log("Cannot perform action (" + e.getMessage() + "). Try again.");
            }

        }
        scanner.close();

        try {
            game.addParticipant(remainingRankedParticipants.remove(0));
            game.addParticipant(remainingRankedParticipants.remove(0));
        } catch (TournamentException e) {
            assert false : "INTERNAL ERROR: a game was not constructed correctly! This should never happen.";
        }
        initialRound.add(game);
    }
    assert remainingRankedParticipants.size() > 0
            : "INTERNAL ERROR: there are participants remaining! This should never happen.";
    return initialRound;
}
```