Krieg

This document is a step by step guide to Krieg and mentions the basic game rules. The participants have to submit their codes in C/C++/Python 2.

1 Terminologies

1.1 Game Board

The game consists of 5 rows and 5 columns. The rows have been initialized from 0 to 4 (from top to bottom). The columns have been initialized from 0 to 4 (from left to right). Row 0 contains the Sentinel bombers, row 1 contains the Sentinel stingers, row 2 is empty, row 3 contains the Scourge stingers and row 4 contains the Scourge bombers.

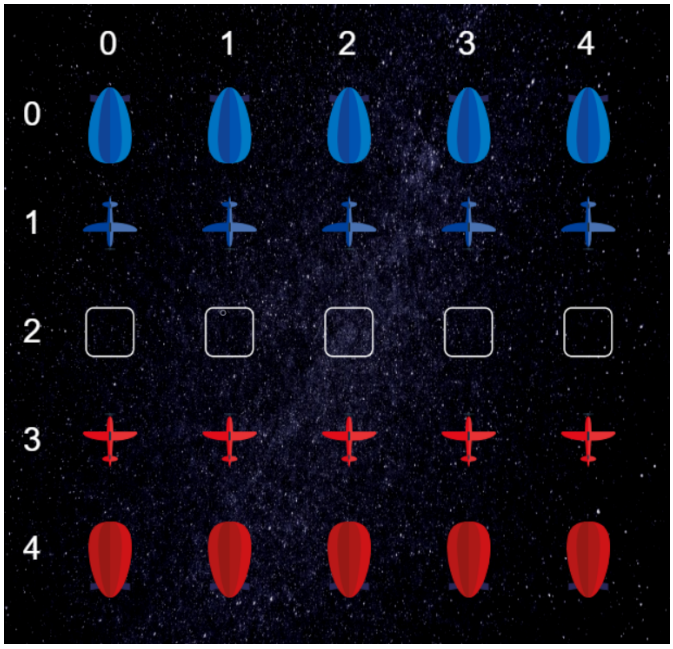

Figure 1

1.2 Stingers

The game has 10 stingers. Each player has 5 stingers, all placed in a single row.

1.3 Bombers

The game has 10 bombers. Each player has 5 bombers, all placed in a single row.

1.4 Initial Input

The first player will be chosen at random and will receive input as 1 (tokens-Sentinels) while the second player will receive input as 0 (tokens-Scourges). No output is expected at this stage.

Note : Your base will be row 0 if your tokens are Sentinels and row 4 if Scourges.

1.5 Input Format

The input specifies the opponents previous move. The input your bot will receive is a 7 character string (including spaces) of the format :-
"r1 c1 r2 c2"
r1 c1 = initial position
r2 c2 = final position
r: row number, c: column number

1.6 Output Format

The output specifies your bot's move. The output your bot will give is a 7 character string (including spaces) of the format :-
"r1 c1 r2 c2"
r1 c1 = initial position
r2 c2 = final position
r: row number, c: column number

2 Game Play

The game is turn based. Each player gets alternate turns and each turn is composed of 1 move. Numbers '1' and '2' would be assigned to the bots, where '1' denotes Player 1 and '2' denotes Player 2.

Note : All moves are only in the forward direction. There are no moves in the backward direction.

2.1 Stinger Movement

Stingers can move one space at a time only diagonally, eliminations being the exception (see subsection 2.3 for reference).

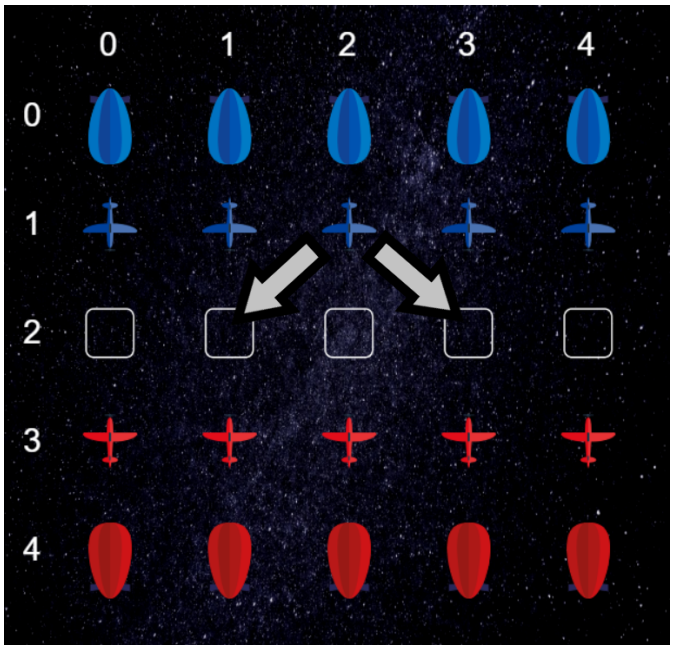

Figure 2

2.2 Bomber Movement

Bombers can move one place at a time in any direction, eliminations being the exception (see subsection 2.3 for reference).

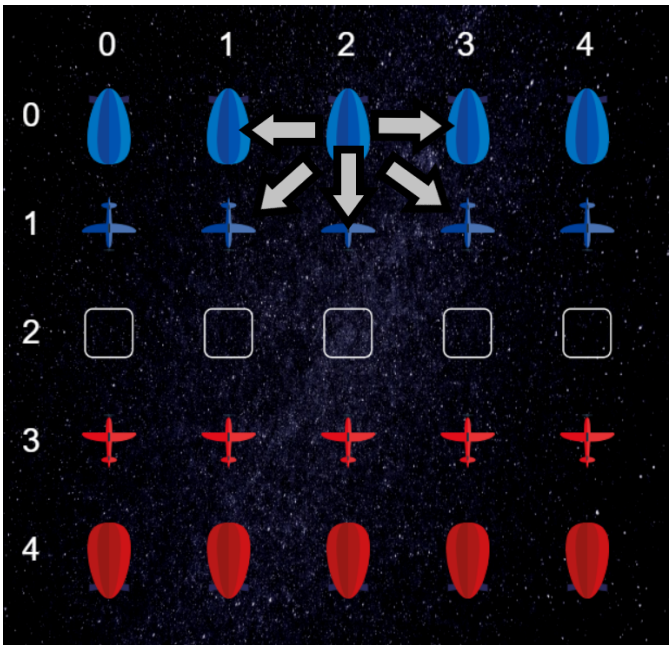

Figure 3

2.3 Elimination

Case 1: Vertical Elimination
When your token (stinger or bomber) comes face to face with the opponent's token and the space behind the opponent's token is empty, then the player can eliminate the opponent's token.

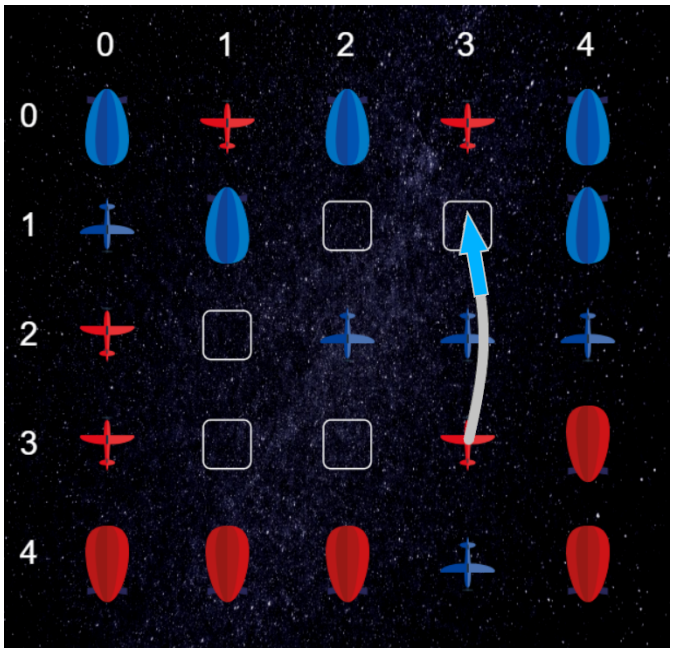

Figure 4

Case 2: Horizontal Elimination
When your token (stinger or bomber) has an enemy token on its left or right side, and the space next to the enemy token is empty, then the player can eliminate the opponent's token.
Note: Horizontal eliminations in row 0 and row 4 are not allowed.

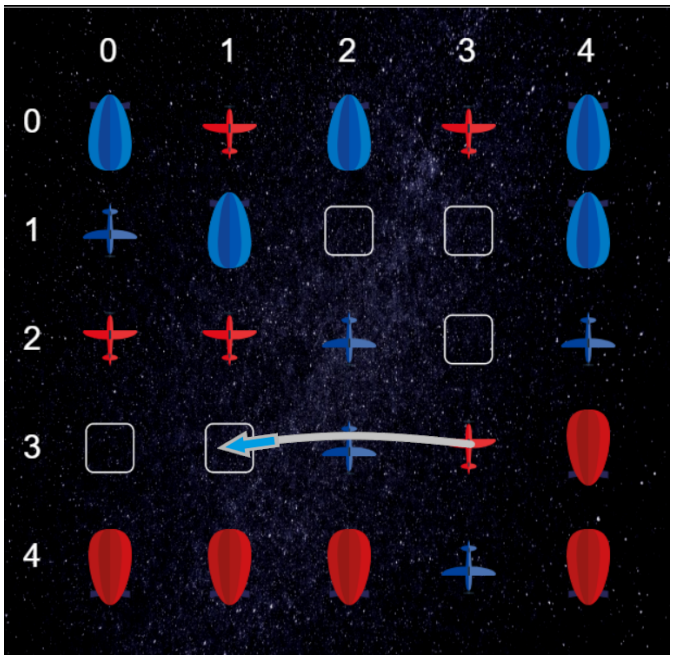

Figure 5

2.4 End Of The Game

- The game will end when one of the bombers of either bot reaches the opponent's base (row 4 if player is blue, row 0 if player is red).
- If a player loses all five bombers, the opponent wins by default.
- If all tokens of a player are blocked (no valid moves remaining), the game will terminate. In this case the player with more number of bombers on the board wins.

Note: The bot program will be terminated automatically when game is over.

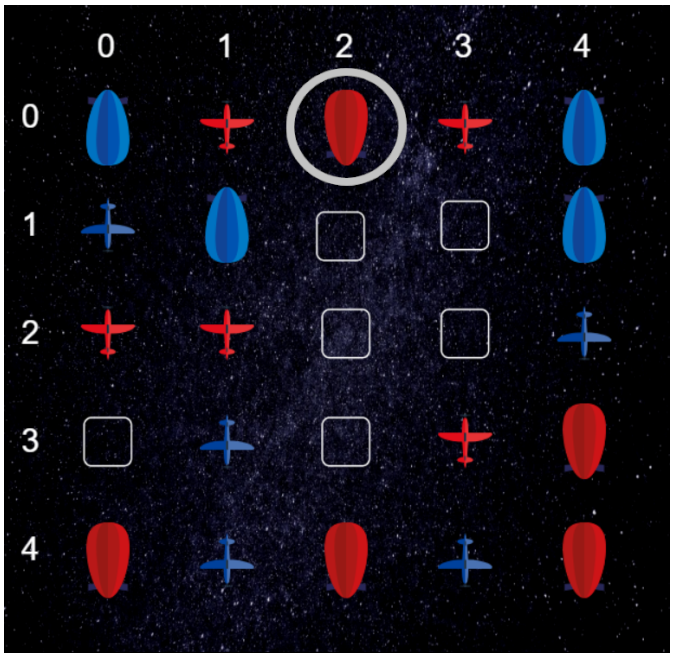

Figure 6

2.5 Bot Format

The player's bot (code) is run only once, with his/her logic executed in an unconditional while loop.
The format of the code should be as follows:

```
    main ()
{

/*initial input scanned by the player to receive player number: whoAmI*/
/*if turn = 1, you are Sentinel and it's your turn first*/
/*if turn = 0, you are Scourge and it's opponent's turn first*/

turn = whoAmI
while(1)
{

/* if(turn!=1)

{
```

```
/*input scanned by the player specifying the opponent's previous move: x*/
/*player's logic to make changes in his board according to the input*/

}

/*player's logic to make the next move*/
/*output printed by the player at the end of every iteration specifying the current move: y*/



    turn=0
}
}
```

whoAmI=1 (Sentinel)
whoAmI=0 (Scourge)
x- move input received
y- move output to be printed

Important Note: Sentinel starts from row 0 and Scourge starts from row 4.

2.6 Disqualification Criteria

Bot will be disqualified due to any of the following reasons:

- Bot does not respond within 2 seconds (for C and C++)/6 seconds(for python 2) of its execution.
- Bot returns an invalid move.
- Syntactical errors in program.
- Excessive resource usage (Bot should consume less than 20MB memory and max file size must be less than 1MB).
- Any malicious activity encountered in the code (The latest version of the bot would be taken into consideration).

Note: In case of any disputes, the decision of the XOdia team will be final.