version 2 - october 3
please see fix on page 3.

COMP 472/6721 **Project** **Fall 2017**

Due dates:	Wednesday Oct 18 2017 + Monday Nov 6 2017 + Monday Dec 4 2017
Late submission:	20% per day on each deliverable.
Teams:	Students registered in COMP 472 can do the project in teams of at most 4. Students registered in COMP 6721 can do the project in teams of at most 2. Teams must submit only 1 copy of the project.
Purpose:	The purpose of this project is to make you develop a heuristic and the minimax algorithm.

In this project, you will implement a minimax algorithm and design a heuristic to play a game called **Bonzee**.

Game:
Bonzee is an adversary game between 2 players. One player has Red tokens and the other has Green tokens. The game is played on a 9x5 chess-like board. As shown in Figure 1, board cells are numbered horizontally from 1 to 9 and vertically from A to E. Note that the board has 2 types of cells: white cells and black cells (the difference will be explained later).

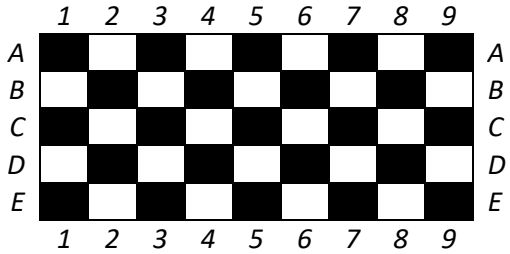

Figure 1: Game board

Initial board:
Initially, the red and green tokens are placed on the board as shown in Figure 2. Notice that cell C6 is left empty.

	1	2	3	4	5	6	7	8	9	
A	R	R	R	R	R	R	R	R	R	A
B	R	R	R	R	R	R	R	R	R	B
C	G	G	G	G		R	R	R	R	C
D	G	G	G	G	G	G	G	G	G	D
E	G	G	G	G	G	G	G	G	G	E
	1	2	3	4	5	6	7	8	9	

Figure 2: Initial configuration of the game

Moves:
Tokens can move to an adjacent cell as long as the destination cell is empty and is within the board (obviously!). When a token is sitting on a black cell, it can move horizontally, vertically or diagonally. However, when a token is sitting on a white cell, it can only move horizontally or vertically (it cannot move diagonally). For example,

- from position A2, a token can move to cells A1, B2 or A3, as long as they are free.
- from position A3, a token can move to cells A2, B2, B3, B4 or A4, as long as they are free.
- from position C4, a token can move to cells B4, D4, C3 or C5, as long as they are free.
- from position C5, a token can move to cells B5, D5, C4, C6, D4, D6, B4 or B6, as long as they are free.

Turns:
Green always plays first, then Red, then Green, ... in an alternate fashion. At each turn, a player choses one of his/her tokens and moves it by a single position to an adjacent cell (straight only on white cells, or either straight or diagonally on black cells).

A player can do 2 types of moves: a sequence of attacks or a defensive move.

1. An attack can be either:
    a) A *forward attack* consists of moving a token 1 cell *toward* an opponent's token(s) and landing next to a sequence of *n* consecutive tokens from the opponent (*n≥1*). In that case, the player can attack all *n* tokens and remove them from the board.
    b) A *backward attack* consists of starting in a cell next to a sequence of *n* consecutive tokens from the opponent (*n≥1*) and moving *away* from them. In that case, the player can attack all *n* tokens and remove them from the board.
2. A defensive move consists of moving a token to an adjacent cell without attacking any opponent token.

Note that if a token moves to a cell that can attack the opponents' tokens, these tokens must be removed from the board.
For example, Figure 3 shows a forward attack, where Green can move the token in D4 to C5. By doing so, the Green token will be moved directly towards the Red tokens in B6 and A7 and will land next to them. This way, both tokens B6 and A7 will be attacked and removed from the board.

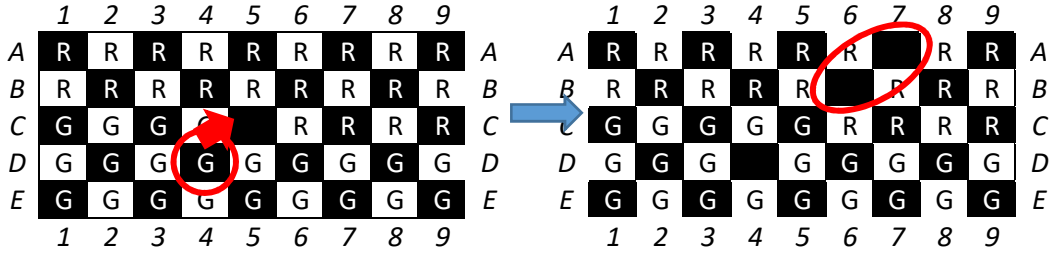

Figure 3: Example of a forward attack: Green in D4 is moved to C5 and attacks B6 and A7

Figure 4 shows an example of a backward attack, where token B7 is moved to A7. By doing this, the Red token will be moved directly away from the 3 Green tokens in C7, D7 and E7. This way, all 3 tokens will be attacked and removed from the board.

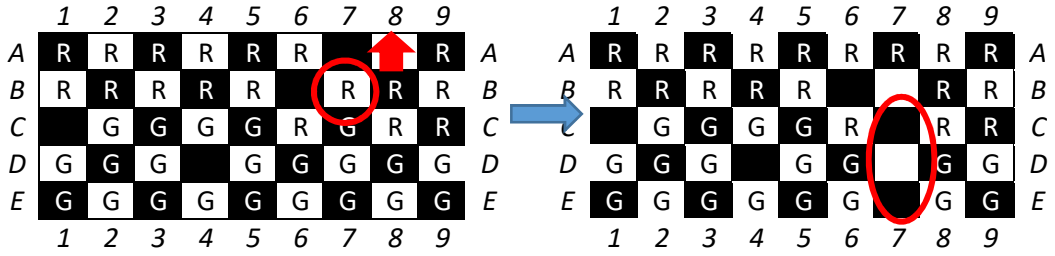

Figure 4: Example of a backward attack: Red in B7 is moved to A7 and attacks C7, D7 and E7

Figure 5 shows an example of a defensive move where the Green token in E4 is moved to E5, and leads no attack. Note, however, that although this is a valid move, it is not the smartest one, because now Red can move D6 to C7 and attack E5 by backward attack.

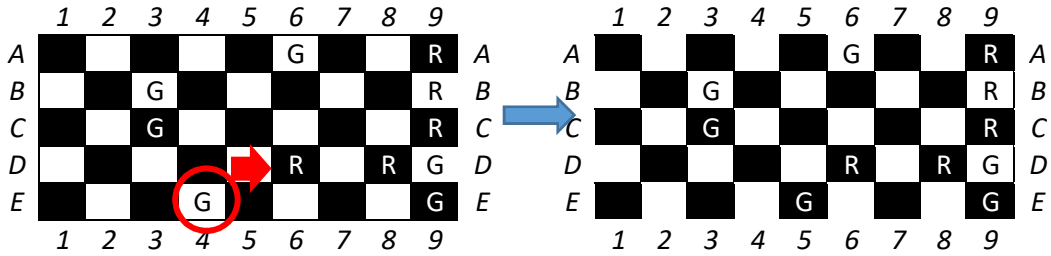

Figure 5: Example of a defensive move: Green in E4 is moved to E5

An Ambiguous Attack is a Forward Attack!

If a move could be used as either a forward attack (and eliminate say, *n* tokens) or a backward attack (and eliminate say, *m* tokens), then the move will, by definition, be considered a forward attack. No, the player cannot apply both types of attacks and eliminate all *n+m* tokens!.

This is illustrated in Figure 6. If C4 is moved to C5, then the move is considered a forward attack, and 3 red tokens will eliminated (C6, C7 and C8), even if the move could also be considered a backward attack.

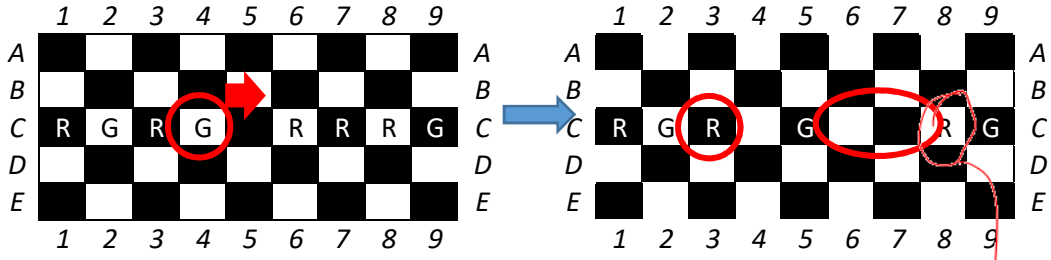

Figure 6: Example of an ambiguous attack: the move will be considered a forward attack

this 'R' should not be there.

End of the game:

To win, a player must either attack all of the opponent's tokens.

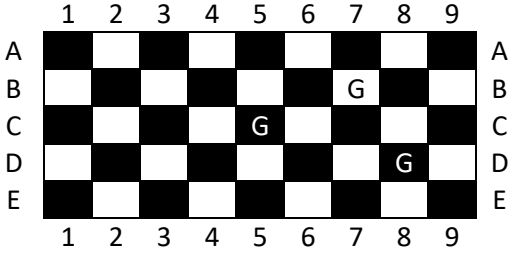

Figure 7: Example of a win for Green.

However, after 5 consecutive non-attacking moves from both players (i.e. 10 consecutive non-attacking moves in total), then the game is considered a draw.

Your Task

In this project, you will implement a minimax algorithm to play the game of Bonzee automatically. You are free to develop the heuristic that you want, but your program must decide on the next move to take in at most 3 seconds (real time in the lab).

Apart from the minimal requirements above (minimax + 3 seconds to decide), your project's grade will depend on the "extra" features you have implemented; for example, the development of a heuristic that takes into account several features, the use of alpha-beta pruning, or...

Play Modes

Your program should be able to run in manual mode and in automatic mode. This means that you should be able to run your program with:

1. manual entry for both players (i.e. 2 humans playing against each other)
2. manual entry for one player, and automatic moves for the other (i.e. one human playing against your "AI")

After each move, your program must display the new configuration of the board.

Programming Details

1. To program the game, you can use Python, Java, C or C++. If you wish to use another language, please check with me first.
2. It is not necessary to have a fancy user-interface. A simple command-line interface with a text-based output is sufficient.
3. In order to facilitate the running of the tournament (see below), we will use the following standards:
   a. The player (human or AI) will indicate its move by entering the coordinates of the moving token followed by the coordinate(s) or where that token should go. For example, **`C5 B6`** indicates that the token at position C5 will move to position B6.
   b. If a <u>human</u> player enters an illegal move (e.g. **`K9`**, **`C5 A2`**), then the player will only be warned and be given a chance to enter another move with no penalty[1]. However, if <u>your AI player</u> generates an illegal move, then it will automatically lose the game.

Tournament

To make the project more fun, we will organize a tournament between all the projects submitted. We will make pairs of AI players play against each other (e.g. your AI against another team's AI). If, at any time, your AI takes more than 3 seconds to decide its next move, it will automatically be eliminated from the game, and your opponent will win.

One bonus point will be allocated to your result in this tournament.

More details about the tournament will be provided later in the semester.

[1] The point of this rule is to avoid losing a game in the tournament because a human did not transcribe the computer's move correctly.

Deliverables

The submission of the project will consist of 3 deliverables + the tournament that will take place in the lab.

Deliverable	***Functionality***	***Due Date***
Deliverable I	manual game (no minimax, no heuristic)	Wednesday Oct 18 2017
Deliverable II	Deliverable I + minimax with a simple heuristic	Monday Nov 6 2017
Tournament (nothing to submit)	Deliverable II + your informed heuristic (if you plan on winning some games!)	Tuesday Nov 28 & Wednesday Nov 29
Deliverable III	Deliverable II + your final heuristic + oral presentation	Monday Dec 4 2017

Oral Presentation

Your final deliverable must be accompanied by an oral presentation of about 5 to 10 minutes. Your presentation should:

1. Describe your program (how to run it, what the main functions and data structures are …).
2. Describe and justify your heuristic. In particular, describe how you came up with your evaluation function. For example, what features of the game state you considered and why you chose them; how you balanced speed of the evaluation function with performance, and generally why you settled on the evaluation function you did.
3. Describe and explain your results either at the tournament or when you played against it (why you think your heuristic makes good or bad decisions).

Evaluation Scheme:

Deliverable I	25%
Deliverable II	40%
Result at the Tournament	10% + [0pt...1pt] bonus
Deliverable III	25%

A detailed evaluation scheme for each deliverable will be posted on Moodle later in the semester.

Submission:

Deliverables I, II and III must be handed-in electronically on the due date.

1. Make sure that you have signed the expectation of originality form (available on the Web page; or at: http://www.encs.concordia.ca/documents/expectations.pdf) and given it to me.
2. In addition, write one of the following statements on your assignment:
   "We certify that this submission is the original work of members of the group and meets the Faculty's Expectations of Originality"
   with the signatures and I.D. #s of all the team members and the date.
3. Hand in each deliverable electronically:
   - Create one zip file, containing all files for your deliverable.
   - Name your zip file:
     deliverable_N_studentID1_studentID2_... (for group work)
     where N=1 or 2 or 3
   - Upload your zip file at: https://fis.encs.concordia.ca/eas/ as project1, project2, and project3.

Have fun!