CW2: Adversarial Search Report

(Question 1): Look at the attached python file in lines: 4 - 266.

(Question 2): Look at the attached python file in lines 274 - 437.

(Question 3): To answer this question I used the time module in python as shown in lines 237-241 for MiniMax without pruning, and lines 410-414 for Minimax with pruning. In conclusion, there is a significant difference in the time taken for action selection for the MiniMax algorithm with and without alpha-beta pruning. When using alpha-beta pruning the action selection time decreases significantly. To demonstrate this, I have provided the action selection times for the (3,3,3) game with and without pruning in the table below. Note that this type of game will always end in a draw due to the MiniMax algorithm.

Move	1st	2nd	3rd	4th	5th
Alpha-Beta	0.696	0.0321	0.0024	0.0011	0.0002
No Pruning	19.18	0.326	0.01	0.00164	0.0002

Table 1: Time taken to select an action (s)

There is a significant difference in time because when using alpha-beta pruning the algorithm does not have to run through every single iteration of the game. As soon as the algorithm knows one is better than a previous iteration, it can just ignore it and move onto the next one. Additionally, the time taken to make an action decreases as more moves are made on the board. This is because there are less positions that we have to check. Once we get to the end of the game, the time taken with and without pruning is similar, as less pruning is needed. At the start of the game, the difference in time taken is very significant however.

(Question 4): When the values of (m, n, k) are scaled up the time taken to select an action increases significantly. The time taken for the alpha-beta pruning version increases by a significant amount, but the time taken without pruning increases much more significantly. The time taken when there is no pruning increases so much that for any game above $(4, 3, 3)$ I do not have enough memory/time for my laptop to run the game. For alpha-beta pruning I can run the games for $(4, 4, 4)$ and $(5, 5, 3)$ however if I increase any of the values more than this I no longer have enough memory/time on my laptop to fully run the game. When I use alpha-beta pruning, I find that scaling the value of k up has more of an effect on the time taken to select an action than scaling m or n up.

Below are tables to demonstrate how the time taken for action selection changes as we scale up the values of (m, n, k). Also, I put them in order of time taken.

(3,3,3); Draw:

Move	1st	2nd	3rd	4th	5th
Alpha-Beta	0.696	0.0321	0.0024	0.0011	0.0002
No Pruning	19.18	0.326	0.01	0.00164	0.0002

Table 2: Time taken to select an action (s)

(4,3,3); Player 'X' wins:

Move	1st	2nd	3rd	4th
Alpha-Beta	0.9536	0.206	0.006	0.002
No Pruning	52961	70.6	0.296	0.003

Table 3: Time taken to select an action (s)

(4,4,3); Player 'X' wins; Time taken with no pruning is too long to compute:

Move	1st	2nd	3rd	4th
Alpha-Beta	1.84	0.38	0.17	0.0023

Table 4: Time taken to select an action (s)

(5,4,3); Player 'X' wins; Time taken with no pruning is too long to compute:

Move	1st	2nd	3rd	4th
Alpha-Beta	6.53	2.29	0.77	0.11

Table 5: Time taken to select an action (s)

(4,3,4); Draw; Time taken with no pruning is too long to compute:

Move	1st	2nd	3rd	4th	5th	6th
Alpha-Beta	18.11	1.38	0.15	0.015	0.002	0.0011

Table 6: Time taken to select an action (s)

(5,5,3); Player 'X' wins; Time taken with no pruning is too long to compute:

Move	1st	2nd	3rd	4th
Alpha-Beta	329.88	11.38	1.17	0.063

Table 7: Time taken to select an action (s)

(4,4,4); Draw; Time taken with no pruning is too long to compute:

Move	1st	2nd	3rd	4th	5th	6th	7th	8th
Alpha-Beta	1432	179.06	8.502	2.652	0.223	0.653	0.0451	0.0031

Table 8: Time taken to select an action (s)

As seen from the tables, the time taken to select an action increases significantly as (m, n, k) increase. The value of k seems to increase the time by more than the values of m and n. Unfortunately I cannot get times for $(5, 4, 4)$ and $(5, 5, 4)$ or anything above this as it takes too long for my computer to get an answer. Increasing (m, n, k) when I use the no pruning algorithm escalates the time taken to ludicrous levels. For example, to get the time for $(4, 3, 3)$ with no pruning I had to let my algorithm run overnight.

(Question 5): Unfortunately for this question I could not reproduce the results as my computer did not have enough power to be able to run the algorithms.