MONASH College

MCD4710 - Introduction to Algorithms and Programming
Assignment 2 (10%)

Due: Thursday, January 4, 2023, 11:55 pm - Week 10

Objectives

The objectives of this assignment are:

- To gain experience in designing algorithms for a given problem description and implementing those algorithms in Python.
- To demonstrate your understanding on:
    - Implementing problem solving strategies
    - Recognizing the relationship between a problem description and program design
    - Decomposing code into functions in Python.

Submission Procedure

Your assignment will not be accepted unless it meets these requirements:

1. Your name and student ID must be included at the start of every file in your submission.
2. Save your file(s) into a zip file called YourStudentID.zip
3. Submit your zip file containing your entire submission to Moodle.

Late Submission

Late submission will have 10% deduction of the total assignment marks per day (including weekends). Assignments submitted 7 days after the due date will not be accepted.

Important Notes

- Please ensure that you have read and understood the university's procedure on plagiarism and collusion available at https://www.monashcollege.edu.au/__data/assets/pdf_file/0005/1266845/Student-Academic-Integrity-Diplomas-Procedure.pdf . You will be required to agree to these policies when you submit your assignment.
- Your code will be checked against other students' work and code available online by advanced plagiarism detection systems. Make sure your work is your own. Do not take the risk.
- Your program will be checked against a number of test cases. Do not forget to include comments in your code explaining your algorithm. If your implementation has bugs, you may still get some marks based on how close your algorithm is to the correct algorithm. This is made difficult if code is poorly documented.

MCD4710 - Introduction to Algorithms and Programming
Assignment 2 (10%)

Assignment code interview

Each student will be interviewed during a lab session regarding their submission to gauge your personal understanding of your Assignment code. The purpose of this is to ensure that you have completed the code yourself and that you understand the code submitted. Your assignment mark will be scaled according to the responses provided.

Interview Rubric

0	The student cannot answer even the simplest of questions. There is no sign of preparation. They probably have not seen the code before.
0.25	There is some evidence the student has seen the code. The answer to a least one question contained some correct points. However, it is clear they cannot engage in a knowledgeable discussion about the code.
0.5	The student seems underprepared. Answers are long-winded and only partly correct. They seem to be trying to work out the code as they read it. They seem to be trying to remember something they were told but now can't remember. However, they clearly know something about the code. With prompting, they fail to improve on a partial or incorrect answer.
0.75	The student seems reasonably well prepared. Questions are answered correctly for the most part but the speed and/or confidence they are answered with is not 100%. With prompting, they can add a partially correct answer or correct an incorrect answer.
1	The student is fully prepared. All questions were answered quickly and confidently. It is absolutely clear that the student has performed all of the coding themselves.

Marking Criteria

This assignment contributes to 10% of your final mark.

Task 1 – Create board: 2 marks

Task 2 – Print board: 2 marks

Task 3 – Initialize the board: 4 marks

Task 4 – Place snakes on Board: 3 marks

Task 5 – Place ladders on Board: 3.5 marks

Task 6 – Update player position on Board: 4.5 marks

Task 7 – Check if a game over: 5 marks

Task 8 - Update player current position: 3 marks

Task 9 – Play the game

Task 10 – Continuous snake and ladder game with more difficulty: 8 marks (Task 9 and 10 together)

Programming practice – 4 marks

- Decomposition and use of helper functions (1 mark)
- Variable names and code Documentation (1 mark)
- No hardcoding (2 marks)

Total: 39 marks

MCD4710 - Introduction to Algorithms and Programming
Assignment 2 (10%)

Background Information

In this assignment, you are to implement a computer-based variant of the game Snakes and Ladders. The basic game of Snakes and Ladders is a simple press-your-luck game with one winner. Players score points by moving up on the board. The player who reaches the end of the board first is the winner!

In this assignment you are going to implement a computer-based variant of the game Snakes and Ladders. It is a 2-player game with the following rules.

1. The board is a 5 -by- 10 board (ie. 5 rows and 10 columns) for the basic implementation. The player should be able to create a board of any size.
2. Place three snakes and three ladders randomly on the board. Represent snake's head with an 'S' and tail with an 's'. Similarly represent ladder's top with 'L' and bottom with 'l'. Remember your implementation should allow the user to place any number of snakes and ladders.
3. Initialize human player as P1, P2 and so on depending on number of players and computer player as AI and place them on first square of the board.
4. The players start at the first square, and they have to maneuver through snakes and ladders to get to the last square.
5. Until one of the players wins do the following:
    - Roll the dice
    - Move the player forward for the value got on the dice roll.
    - If the player is on snake's head, move down to its tail
    - If the player is on ladder's bottom, take it to its top
    - else remain there and let the next player roll the dice

The following diagrams show some of the players places.

In this assignment the user is expected to play the game against the computer. The user players will be player 1 ('P1'). player 2 ('P2') depending on the number of players and the computer will be the AI player ('AI').

Task 1: Create the board

Implement a function named `create_board(NUM_ROWS,NUM_COLS)` which creates a 5x10 board (ie. 5 rows and 10 columns) for the basic game but user should be able to input any number. Fill each cell with an empty string (''). Implement the board as a table (ie. a list of lists) in python. This function should return the table (i.e. list of lists). It should not print the board.

Input: No input taken
Output: A table that represents the 5x10 board with all the cells filled with a value.

For example:

```
>>> board = create_board(NUM_ROWS,NUM_COLS)
>>> board
[['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', ''],
    ['', '', '', '', '', '', '', '', '', ''], ['', '', '', '', '', '', '', '', '', '']]
```

Task 2: Print board

Write a function named `print_board(board)` that prints the given board in the format shown in the example below.

Input: The current status of the board
Output: Prints the current board to the screen

For example:

```
>>> board = create_board(NUM_ROWS,NUM_COLS)
>>> print_board(board)
Snakes and Ladders Board:
   0   1   2   3   4   5   6   7   8   9
----------------------------------------
0|   |   |   |   |   |   |   |   |   |   |
----------------------------------------
1|   |   |   |   |   |   |   |   |   |   |
----------------------------------------
2|   |   |   |   |   |   |   |   |   |   |
----------------------------------------
3|   |   |   |   |   |   |   |   |   |   |
----------------------------------------
4|   |   |   |   |   |   |   |   |   |   |
----------------------------------------
```

Task 3: Initialize the board

Write a function named `initialise_board(board,NUM_SQUARES)` that initializes the board with values from 1 to 50 for a 50 square board in a zig zag pattern simulating a snakes and ladders board (see the example doctest below). Also, place the players on the first square.

Input: The current status of the board and number of squares.
Output: A table that represents the 5x10 board with values placed in squares showing a zig zag pattern.

For example:

```
>>> board = initialise_board(board, NUM_SQUARES)
>>> board
Snakes and Ladders Board:
   0    1    2    3    4   5    6    7   8    9
-------------------------------------------------
0| 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 |
-------------------------------------------------
1| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
-------------------------------------------------
2| 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 |
-------------------------------------------------
3| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
-------------------------------------------------
4| 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1P1AI |
-------------------------------------------------
```

Task 4: Place snakes on the Board

Write a function named `place_snakes(board, NUM_SQUARES,NUM_SNAKES)` that initialises the board by placing three snakes randomly. If the cell has a snake head represent it with **'S'** and if a cell has a snake tail represents it with **'s'** and so on. Note that we use this representation for snake and ladder so that when the board is displayed on the screen, the player can clearly see which cells have snakes head and tail.

Input: The current status of the board, number of squares and number of snakes.
Output: A table that represents the 5x10 board with snakes and ladders
placed on the board in specific places.
For example:

```
>>> board = initialise_board (board,NUM_SQUARES)
>>> board = place_ladders(board,NUM_SQUARES,NUM_SNAKES)
>>> print_board(board)
Snakes and Ladders Board:
    0   1    2    3   4    5    6    7    8    9
-------------------------------------------------------
0| 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41S |
-------------------------------------------------------
1| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39S | 40 |
-------------------------------------------------------
2| 30 | 29 | 28S | 27 | 26 | 25 | 24 | 23 | 22 | 21 |
-------------------------------------------------------
3| 11 | 12s | 13 | 14 | 15 | 16s | 17 | 18 | 19 | 20 |
-------------------------------------------------------
4| 10s |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 | 1P1AI |
-------------------------------------------------------
```

Task 5: Place ladders on the Board

Write a function named `place_ladders(board, NUM_SQUARES,NUM_LADDERS)` that adds three ladders to the board from task 4 above. The ladders should be placed randomly. If the cell has a ladder top represent it with **'L'** and ladder bottom with **'l'**. Note that we use this representation for snakes and ladders so that when the board is displayed on the screen, the player can clearly see which cells have ladder top and ladder bottom.

Input: The current status of the board, number of squares and number of ladders.
Output: A table that represents the 5x10 board with snakes and ladders
placed on the board in specific places.
For example:

```
>>> board = initialise_board (board,NUM_SQUARES)
>>> board = place_ladders(board,NUM_SQUARES,NUM_LADDERS)
>>> print_board(board)
```

Snakes and Ladders Board:

```
    0     1  2     3  4     5     6     7     8     9
-------------------------------------------------------
0| 50L| 49 | 48 | 47 | 46 | 45L| 44 | 43 | 42l | 41S |
-------------------------------------------------------
1| 31 | 32 | 33 | 34 | 35 | 36L| 37 | 38 | 39S| 40 |
-------------------------------------------------------
2| 30l| 29 | 28S| 27 | 26 | 25 | 24 | 23 | 22 | 21 |
-------------------------------------------------------
3| 11 | 12s| 13 | 14 | 15 | 16s| 17 | 18 | 19 | 20 |
-------------------------------------------------------
4| 10s|  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 | 1P1AI|
-------------------------------------------------------
```

Task 6: Update player position on board

Write a function named `update_player_position(player,position,board)` that moves the players on the board by moving 'P1' or 'AI' on the board.

Note: You have to remove the player (P1/AI) from the previous square and to reflect its updated position move the player (P1/AI) to their updated square.

Input: The current status of the board, current player, current position of the player
Output: A table that represents the 5x10 board with snakes and ladders placed on the board in specific places and both players.

For example: At the start of the game Pl and AI are on square 1.

Snakes and Ladders Board:

```
    0     1     2     3  4     5     6  7     8     9
-------------------------------------------------------
0| 50L| 49 | 48 | 47 | 46 | 45L| 44 | 43 | 42l| 41S|
-------------------------------------------------------
1| 31 | 32 | 33 | 34 | 35 | 36L| 37 | 38 | 39S| 40 |
-------------------------------------------------------
2| 30l| 29 | 28S| 27 | 26 | 25 | 24 | 23 | 22 | 21 |
-------------------------------------------------------
3| 11 | 12s| 13 | 14 | 15 | 16s| 17 | 18 | 19 | 20 |
-------------------------------------------------------
4| 10s|  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 | 1P1AI|
-------------------------------------------------------
```

After one turn when player (P1) rolled 4 on the dice and AI rolled 5 on the dice their updated positions are square 5 and 6 respectively.

Snakes and Ladders Board:

```
    0   1   2   3   4   5   6   7   8   9
  ------------------------------------------------
  0| 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 |
  ------------------------------------------------------
  1| 31L| 32 | 33 | 34 | 35 | 36 | 37L| 38 | 39L| 40 |
  ------------------------------------------------------
  2| 30 | 29 | 28S| 27 | 26 | 25 | 24s| 23 | 22 | 21S|
  ------------------------------------------------------
  3| 11 | 12l| 13s| 14 | 15 | 16 | 17 | 18l| 19 | 20s|
  ------------------------------------------------
  4| 10 |  9 |  8 |  7 | 6P1| 5AI|  4 |  3 |  2 |  1 |
  ------------------------------------------------
```

Task 7: Check if game is over

Write a function named `check_game_over(currentPlayer,board, NUM_SQUARES)` that checks if one of the players has won. If a player has won, the function should display on the screen which player has won and return `'P1' or'P2'` corresponding to the user who has won and `'AI'` if the computer has won. The function should also check if the player current position goes beyond the board size after the dice roll if yes then the player stays in the same position. Otherwise, call update_current_position function to update the current position.

Input: The current status of the board, currentPlayer and number of squares.
Output: The marker of the player who won.

For example: These are sample screens for the situations where the player either won the game or stays in the same position.

```
AI its your turn
You rolled a 6
You will remain in the same position until you get exact value 2
Snakes and Ladders Board:
   0   1   2   3   4   5   6   7   8   9
------------------------------------------------------
0| 50 | 49 | 48AI| 47 | 46 | 45 | 44 | 43 | 42 | 41 |
------------------------------------------------------
1| 31L| 32 | 33 | 34 | 35 | 36P1| 37L| 38 | 39L| 40 |
------------------------------------------------------
2| 30 | 29 | 28S| 27 | 26 | 25 | 24s| 23 | 22 | 21S|
------------------------------------------------------
3| 11 | 12l| 13s| 14 | 15 | 16 | 17 | 18l| 19 | 20s|
------------------------------------------------------
4| 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |
------------------------------------------------------
Enter r to roll the dice or q to quit the game:
P1 its your turn
You rolled a 5
```

```
Snakes and Ladders Board:
    0    1     2    3    4     5    6    7    8   9
-----------------------------------------------------
0| 50 | 49 | 48AI| 47 | 46 | 45P1| 44 | 43 | 42 | 41 |
-----------------------------------------------------
1| 31L| 32 | 33 | 34 | 35 | 36 | 37L| 38 | 39L| 40 |
-----------------------------------------------------
2| 30 | 29 | 28S| 27 | 26 | 25 | 24s| 23 | 22 | 21S|
-----------------------------------------------------
3| 11 | 12l| 13s| 14 | 15 | 16 | 17 | 18l| 19 | 20s|
-----------------------------------------------------
4| 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |
-----------------------------------------------------
AI its your turn
You rolled a 2
Congratulations AI! You have won the game by reaching the last square first.
Would you like to play another round? If yes enter 'y' or else enter 'n' to exit.
```

Task 8: Update player current position value

Write a function named `update_current_position(currentPlayer,currentPosition,board)` that is called from `check_game_over` **f**unction to update the player's current position value

Input: The current status of the board, currentPlayer and currentPosition.
Output: Current possition is updated on the board

For example: These are sample screens for the situations where the player either lands on a snake or lands on a ladder.

```
>>>
AI its your turn
You rolled a 4
Great! You climbed a ladder.
Move forward to square 31
Snakes and Ladders Board:
   0   1   2   3   4   5   6   7   8   9
-----------------------------------------------------
0| 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 |
-----------------------------------------------------
1| 31LAI| 32 | 33 | 34 | 35 | 36 | 37L| 38 | 39L| 40 |
-----------------------------------------------------
2| 30 | 29 | 28S| 27 | 26 | 25 | 24s| 23 | 22 | 21S|
-----------------------------------------------------
3| 11 | 12l| 13s| 14 | 15 | 16 | 17 | 18l| 19 | 20sP1|
-----------------------------------------------------
4| 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |
-----------------------------------------------------
```

```
P1 its your turn
You rolled a 5
Oops! You landed on a snake.
Move backward to square 24
Snakes and Ladders Board:
   0   1   2   3   4   5   6   7   8   9
-------------------------------------------------
0| 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 |
-------------------------------------------------
1| 31L| 32 | 33 | 34 | 35 | 36AI| 37L| 38 | 39L| 40 |
-------------------------------------------------------
2| 30 | 29 | 28S| 27 | 26 | 25 | 24sP1| 23 | 22 | 21S|
-------------------------------------------------------
3| 11 | 12l| 13s| 14 | 15 | 16 | 17 | 18l| 19 | 20s|
-------------------------------------------------
4| 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |
-------------------------------------------------
AI its your turn
```

Task 9: Play the game

Finally, we can implement the play function. Write a function named `play()` that does the following.

1. Initialize the board, let the user decide number of squares, number of rows and columns.
2. Let the user decide how many players they want in the game.
3. Initialize all the players with player names and place them on the first square.
4. Chose one player randomly to start the game.
5. Display the player name who will start. If user is starting then ask the user to either roll or quit. Print the message “You chose to end the game. Try again later!”, if player choses to quit.
6. Else, follow the game rules until a player wins the game and at the end display the player details who won the game.
7. To win the game, Snakes and Ladders rules require a player to roll the dice and get the exact steps needed to win the game. Say if you are left with two spots, then your dice roll must indicate 2. If you get six, you will stay in the same spot and cannot move.

Note: you might use helper functions to display player details who won the game, or swap the players after one player finishes their turn or to roll a dice.

Test your code interactively by playing the game.

Task 10: Continuous game

Continuous Snakes and Ladders alternative game play option. The players must select which version of the game (such as how many squares, snakes and ladders) to play at the start of the game. This version is almost the same as the basic one, but it is a single long round, rather than several short ones. When the player reaches the end of the board then the player can select to play again but with increased difficulty such as next level with a greater number of boards such as 10x10 board and more hazards etc.

Note:

- The board will be initialized in the similar fashion as previous with a greater number of squares, snakes and ladders placed.

For Example: if user selects to exit:

```
>>>
AI its your turn
You rolled a 2
Congratulations AI! You have won the game by reaching the last square first.
Would you like to play another round? If yes enter 'y' or else enter 'n' to exit.
Thank you for playing this game. Bye!
```

For Example: if user selects to continue:

```
>>>
Congratulations P1! You have won the game by reaching the last square first.
Would you like to play another round? If yes enter 'y' or else enter 'n' to exit. y
Enter the number of squares you want on the board: 100
Enter the number of rows you want the board to have: 10
Enter the number of cols you want the board to have: 10
Enter the number of snakes you want the board to have: 4
Enter the number of ladders you want the board to have: 4
Enter the number of players you want to play the game with: 3
Enter1players name such as P1,P2 for player one, two and so on and AI for computer: P1
Enter2players name such as P1,P2 for player one, two and so on and AI for computer: P2
Enter3players name such as P1,P2 for player one, two and so on and AI for computer: AI
9
Snakes and Ladders Board:
     0    1     2     3     4     5     6     7     8     9
-----------------------------------------------------------
0| 91 | 92 | 93 | 94 | 95 | 96S | 97 | 98 | 99 | 100 |
-----------------------------------------------------------
1| 90 | 89 | 88 | 87S| 86 | 85 | 84 | 83 | 82 | 81 |
-----------------------------------------------------------
2| 71L| 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79S| 80 |
-----------------------------------------------------------
3| 70 | 69 | 68 | 67 | 66 | 65 | 64 | 63 | 62 | 61 |
-----------------------------------------------------------
4| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
-----------------------------------------------------------
5| 50S| 49 | 48 | 47 | 46 | 45 | 44 | 43s| 42 | 41 |
-----------------------------------------------------------
6| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40l|
-----------------------------------------------------------
7| 30 | 29 | 28 | 27 | 26 | 25 | 24L| 23 | 22L| 21 |
-----------------------------------------------------------
8| 11 | 12 | 13s| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
-----------------------------------------------------------
9| 10 |   9 |    8 |   7 |  6 |   5 |   4 |   3 |   2s| 1P1P2AI|
-----------------------------------------------------------
```

---------- END OF ASSIGNMENT ----------