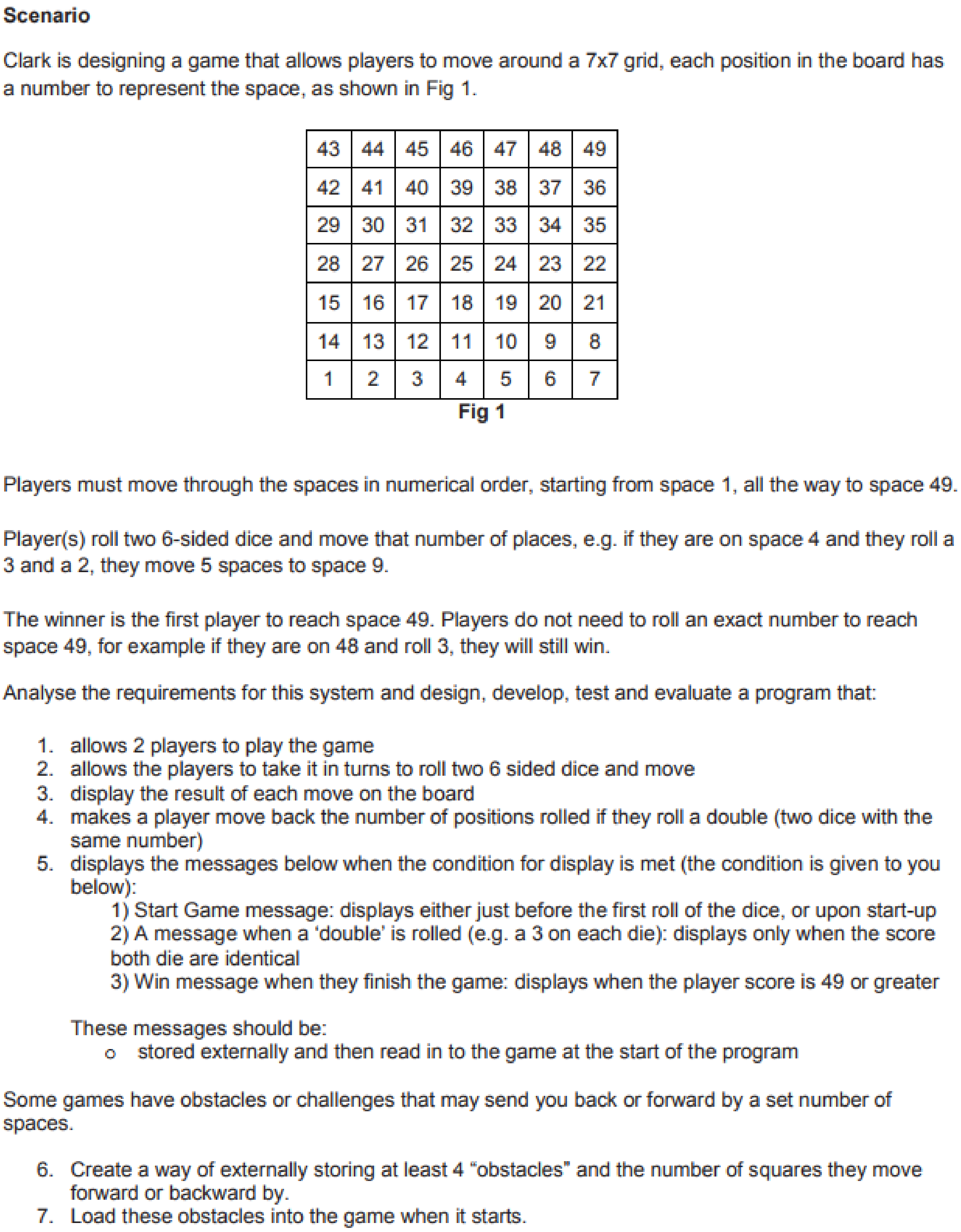
Year 12 Summer Challenge

This MUST follow the Systems Development Lifecycle.

When designing your solution, please ensure that you have completed pseudocode AND flowcharts – this MUST be BEFORE you develop any actual code!!

Python would be preferred, but you are free to choose a specific programming language.



Scenario

Clark is designing a game that allows players to move around a 7x7 grid, each position in the board has a number to represent the space, as shown in Fig 1.

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

Fig 1

Players must move through the spaces in numerical order, starting from space 1, all the way to space 49.

Player(s) roll two 6-sided dice and move that number of places, e.g. if they are on space 4 and they roll a 3 and a 2, they move 5 spaces to space 9.

The winner is the first player to reach space 49. Players do not need to roll an exact number to reach space 49, for example if they are on 48 and roll 3, they will still win.

Analyse the requirements for this system and design, develop, test and evaluate a program that:

1. allows 2 players to play the game
2. allows the players to take it in turns to roll two 6 sided dice and move
3. display the result of each move on the board
4. makes a player move back the number of positions rolled if they roll a double (two dice with the same number)
5. displays the messages below when the condition for display is met (the condition is given to you below):
   1) Start Game message: displays either just before the first roll of the dice, or upon start-up
   2) A message when a 'double' is rolled (e.g. a 3 on each die): displays only when the score both die are identical
   3) Win message when they finish the game: displays when the player score is 49 or greater

   These messages should be:
   - stored externally and then read in to the game at the start of the program

Some games have obstacles or challenges that may send you back or forward by a set number of spaces.

6. Create a way of externally storing at least 4 "obstacles" and the number of squares they move forward or backward by.
7. Load these obstacles into the game when it starts.