INTEGRATIVE TASK 1

Nicolás Cuéllar Molina – A00394970

Davide Flamini Cazarán - A00381665

Andrés Felipe Cabezas - A00394772

Tarea integradora 1.docx - Statement of Integrator 1

PROBLEM SPECIFICATION TABLE

CLIENT	Snakes and Ladders Inc
USER	Snakes and Ladders Players
FUNCTIONAL REQUIREMENTS	- R1: Create the game with the parameters given by the user - R2: Show the player the game board with the players - R3: Show the player the game board with the , snakes and ladders - R4: Define the development of the game. - R5: Create a turn system. - R6: Calculate the player's score - A7: Store game scores in a binary tree.
CONTEXT OF THE PROBLEM	Develop a program that allows you to play and also simulate the famous game Ladders and Snakes.
NON-FUNCTIONAL REQUIREMENTS	- RN1: The grid must be modeled and implemented using linked lists. - RN2: It is not possible to use any array, or arraylist, or any Java collection in this program. - RN3: It is not possible to use cycles in this program. youAll iterations must be done using recursion. - RN4: All ladders and snakes must be modeled as connections between nodes of the linked structure - RN5: The score log data is deleted as soon as you close the program.

Functional Requirements Analysis Table

<table>
<tr><td>Name or identifier</td><td colspan="3">R1: Create the game with the parameters given by the user</td></tr>
<tr><td>Summary</td><td colspan="3">The program must allow receiving by parameter the number of rows, columns, snakes and stairs of the game. From this data the game will be created and the snakes and ladders will be randomly located.</td></tr>
<tr><td rowspan="5">Inputs</td><td>input name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>rows</td><td>int</td><td></td></tr>
<tr><td>columns</td><td>int</td><td></td></tr>
<tr><td>numLadders</td><td>int</td><td></td></tr>
<tr><td>numSnakes</td><td>int</td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. Ask for the number of columns, rows, snakes and ladders
2. Verify that the data is of integer type
3. Create the board according to the received parameters</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">A game board of snakes and ladders according to the parameters and rules for its creation.</td></tr>
<tr><td rowspan="2">Outputs</td><td>output name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>gameBoard</td><td>String</td><td></td></tr>
</table>

<table>
<tr><td>Name or identifier</td><td colspan="3">R2: Show the player the game board with the players</td></tr>
<tr><td>Summary</td><td colspan="3">The system must allow showing the players on the board. You can represent each player with any of the following symbols: * ! Or X % $ # + &.</td></tr>
<tr><td rowspan="2">Inputs</td><td>input name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>optionSymbol</td><td>int</td><td>That the data be of type int</td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. Ask for the option of the symbol with which the player wishes to be represented
2. Verify that the option entered is of integer type and is within the range of options
3. Show the player on the game board</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">A game board where players are represented by symbols</td></tr>
<tr><td rowspan="2">Outputs</td><td>output name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>gameBoard</td><td>String</td><td>The game has been initialized</td></tr>
</table>

<table>
<tr><td>Name or identifier</td><td colspan="3">R3: Show the player the game board with the snakes and ladders</td></tr>
<tr><td>Summary</td><td colspan="3">The system must allow the user to see the game board, where snakes and ladders are shown, taking into account that no snake starts on the n x m square (board size), and no ladder or snake start or end square must coincide with another start or end of ladder or snake.</td></tr>
<tr><td rowspan="2">Inputs</td><td>input name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>N/A</td><td></td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. Traverse the game board recursively
2. Verify in which nodes there are snakes or ladders, and print them
3. Display the game board with the snakes and ladders on the screen</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">The game board is displayed, where you can see the game's snakes and ladders.</td></tr>
<tr><td rowspan="2">Outputs</td><td>output name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>gameBoard</td><td>String</td><td>The game has been initialized</td></tr>
</table>

<table>
<tr><td>Name or identifier</td><td colspan="3">R4: Define the development of the game.</td></tr>
<tr><td>Summary</td><td colspan="3">Once the game board has been created, the board will be displayed with the position of each player and all the squares on it. In addition, a turn system will be used with a submenu that allows showing a board with the position of the snakes and ladders or moving through the squares until one of them reaches the end.</td></tr>
<tr><td rowspan="2">Inputs</td><td>input name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>N/A</td><td></td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. Show the board with all the cells and the players
2. Show turn system submenu
3. Execute whatever the player has chosen
4. Display the updated board with player positions or a game over in case one of the players reaches the end.</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">Updated game board or a game over.</td></tr>
<tr><td rowspan="2">Outputs</td><td>output name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>gameBoard</td><td>String</td><td>The game has been initialized</td></tr>
</table>

Name or identifier	**R5: Create a turn system**		
Summary	The system must have a defined turn system for the game. On their turn each player has access to a menu that will allow them to roll the dice or see the ladders and snakes.		
Inputs	**input name**	**Datatype**	**Selection or repetition condition**
	option	int	
General activities necessary to obtain the results	1. Print menu on screen 2. Read the option chosen by the user 3. Roll the dice randomly and move the player to the corresponding square. (If you choose 1) 4. Show the board of snakes and ladders (If you choose 2) 5. Continue the turn to the next player, as long as no one has reached the end.		
Result or postcondition	New updated player position or board with the snakes and ladders		
Outputs	**output name**	**Datatype**	**Selection or repetition condition**
	confirmation	String	That the turn system has been created correctly

Name or identifier	**R6: Calculate the player's score**		
Summary	When a player reaches the last square, the score is calculated using the formula (600 - t) / 6. Where t is the time in seconds between the start of the game and when one of the players reaches the finish line. This formula is made so that after 10 minutes, the score starts to be negative.		
Inputs	**input name**	**Datatype**	**Selection or repetition condition**
	t	double	
General activities necessary to obtain the results	1. Receive the time in seconds elapsed between the start of the game and when one of the players reaches the finish line. 2. Calculate the score using the formula proposed 3. Return the calculated score.		
Result or postcondition	A new calculated score		
Outputs	**output name**	**Datatype**	**Selection or repetition condition**
	scorePlayer	double	

Name or identifier	R7: Store the winning player's score in a binary tree		
Summary	This function allows to store the score of the winning players in a binary search tree		
Inputs	**input name**	**Datatype**	**Selection or repetition condition**
	scorePlayer	double	The score was calculated previously
General activities necessary to obtain the results	1. Receive the scorePlayer. 2. Calculate where it should be placed in the binary tree. 3. Enter it in the corresponding position.		
Result or postcondition	A new saved score in a binary tree		
Outputs	**output name**	**Datatype**	**Selection or repetition condition**
	confirmation	String	That the score has been saved correctly in the binary tree.