PROCEDURAL PROGRAMMING

PROJECT – SNACKS AND LADDERS

Alexander Souza – G00317835@GMIT.IE

2nd year Software Development

GMIT

PROCEDURAL PROGRAMMING

PROJECT – SNACKS AND LADDERS

CONTENTS

CONTENTS

INTRODUCTION

Snakes and Ladders is a quite simple board game.

It has been around for ages. First instance of the game played was recoded in 2nd century BC in India where it was know as Moksha-patamu. The game was discovered by Europeans during the colonization of India and spread widely around the world. It has been originally used to teach children about good and bad as ladders represented good deeds and snakes punishment for the bad. Nowadays, though, the game does not carry any ethical or religious meaning.

Rules

Players

Snakes and Ladders is played by two to six players, each with her own token to move around the board.

Moving

Players roll a dice, then move the designated number of spaces, between one and six. Once they land on a space, they have to perform any action designated by the space.

Ladders

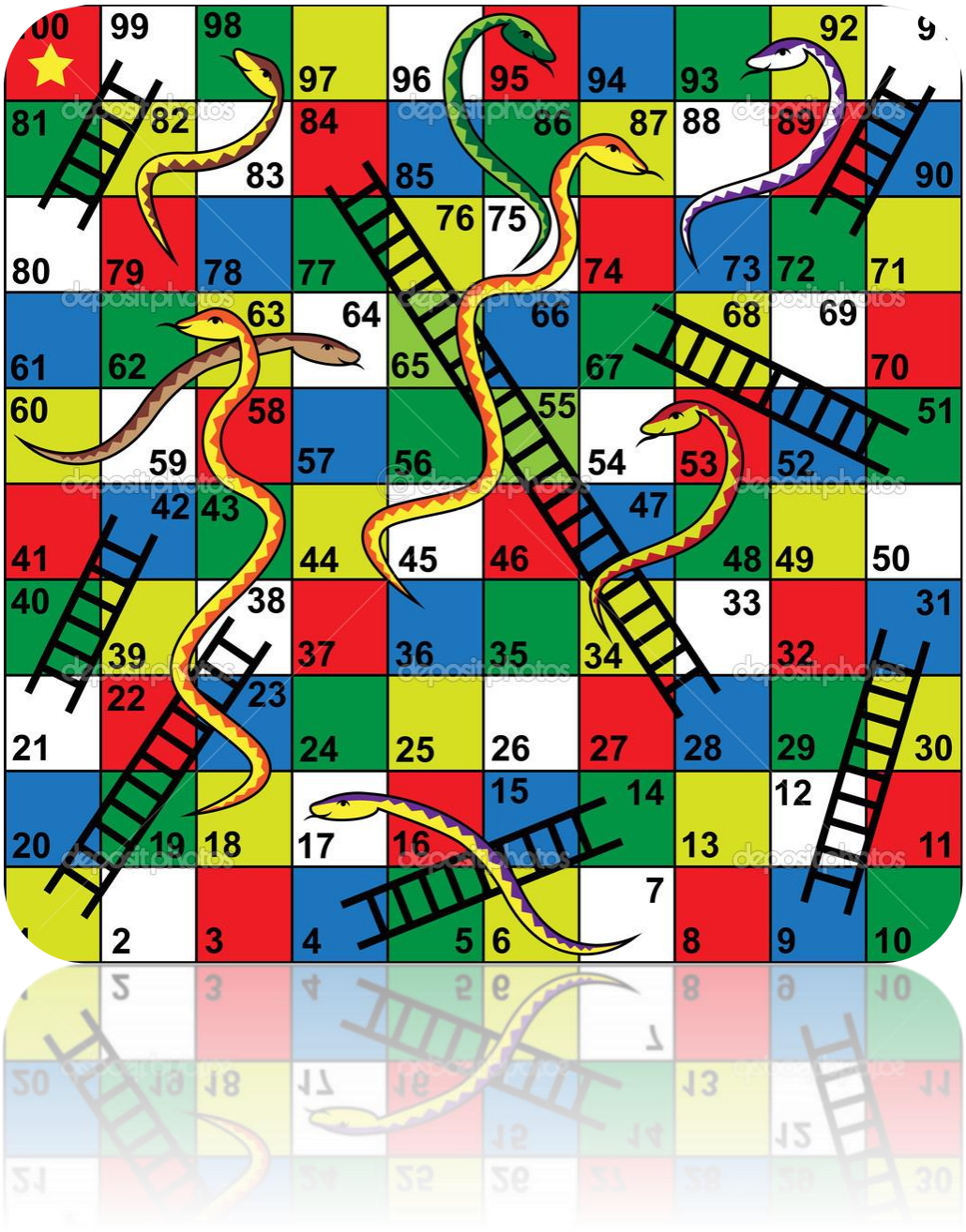
If the space a player lands on is at the bottom of a ladder, he should climb the ladder, which brings him to a space higher on the board.

Snakes/Chutes

If the space a player lands on is at the top of a snake/chute, she must slide down to the bottom of it, landing on a space closer to the beginning.

Winning

The winner is the player who gets to the last space on the board first, whether by landing on it from a roll, or by reaching it with a ladder.



ABSTRACT

This project aims to bring the fun and simplicity of snake game with some new features. It will include computer controlled intelligent opponents whose aim will be to challenge the human players. It will also have the multiplayer feature that will allow more than one players to play the game.

This project explores a new dimension in the traditional snake game to make it more interesting and challenging. The simplicity of this game makes it an ideal candidate for a minor project as we can focus on advanced topics like multiplayer functionality

OBJECTIVES

This game aims to change the way people think of traditional snake game. It will offer the experience of commercial multilayer games to the player retaining the simplicity of traditional snake game.

The major objectives of this project are:

- Create a snake game that will have all the functionality of traditional snake games.
- Introduce multilayer functionality in the game that will allow several players to play a game simultaneously. It should be able to give the experience of a real time multiplayer game to the players.

PROGRAMMING ENVIRONMENT

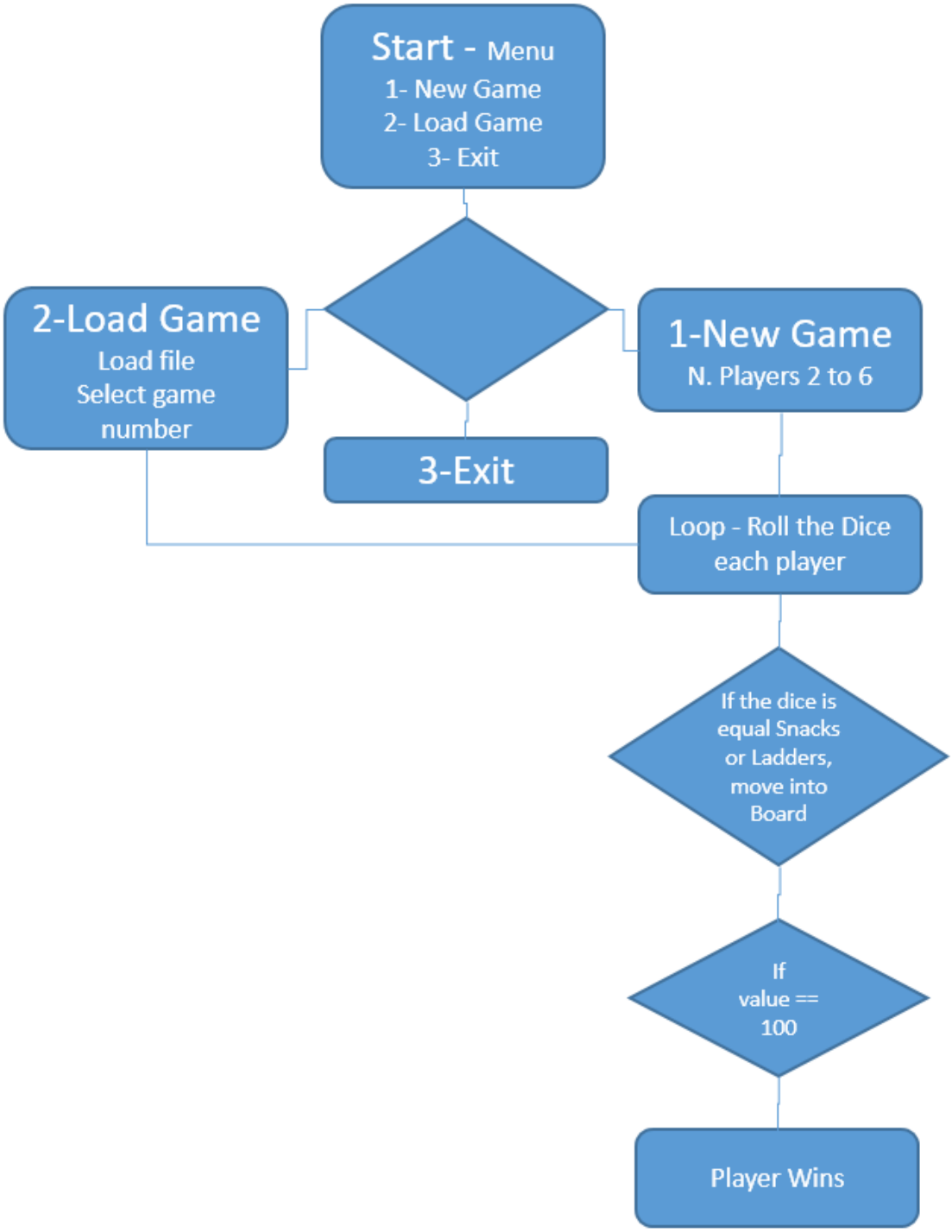
We used several open source tools to develop this project:

- Visual Studio 2015

All the developers of Game used Visual Studio 2015 for the development of this project.

METHODOLOGY

Pseudocode



Methods

printLogo()

```
94   /*
95   Prints the logo of the game
96   snake in ASCII code.
97   */
98   void printLogo()
99   {
100      clrscr();
101      printf("*********************************************************************************\n");
102      printf("                    Snakes and Ladders - Alexander Souza G00317835\n");
103      printf("*********************************************************************************\n");
104      printf("\n                      /^\\/^\\");
105      printf("\n                    _|__|  O|");
106      printf("\n           \\/     /~     \\_/ \\");
107      printf("\n            \\____|__________/  \\");
108      printf("\n                   \\_______      \\");
109      printf("\n                           `\\     \\                 \\");
110      printf("\n                             |     |                 \\");
111      printf("\n                            /      /                  \\");
112      printf("\n                           /     /                     \\");
113      printf("\n                         /      /                       \\ \\");
114      printf("\n                        /     /                          \\  \\");
115      printf("\n                      /     /             _----_            \\   \\");
116      printf("\n                     /     /           _-~      ~-_         |   |");
117      printf("\n                    (      (        _-~    _--_    ~-_     _/   |");
118      printf("\n                     \\      ~-____-~    _-~    ~-_    ~-_-~    /");
119      printf("\n                       ~-_           _-~          ~-_       _-~");
120      printf("\n                          ~--______-~                ~-___-~\n\n");
121  }
122
```

printBoard()

```
124
125  //Prints on the screen the board positions
126  void printBoard()
127  {
128      clrscr();
129      printf(" ------------------------------------------------------------------         Status ==> %s\n", statusGame);
130      printf("| 100  |  99  |  98  |  97  |  96  |  95  |  94  |  93  |  92  |  91  |\n");
131      printf("|  81  |  82  |  83  |  84  |  85  |  86  |  87  |  88  |  89  |  90  |\n");
132      printf("|  80  |  79  |  78  |  77  |  76  |  75  |  74  |  73  |  72  |  71  |\n");
133      printf("|  61  |  62  |  63  |  64  |  65  |  66  |  67  |  68  |  69  |  70  |\n");
134      printf("|  60  |  59  |  58  |  57  |  56  |  55  |  54  |  53  |  52  |  51  |\n");
135      printf("|  41  |  42  |  43  |  44  |  45  |  46  |  47  |  48  |  49  |  50  |\n");
136      printf("|  40  |  39  |  38  |  37  |  36  |  35  |  34  |  33  |  32  |  31  |\n");
137      printf("|  21  |  22  |  23  |  24  |  25  |  26  |  27  |  28  |  29  |  30  |\n");
138      printf("|  20  |  19  |  18  |  17  |  16  |  15  |  14  |  13  |  12  |  11  |\n");
139      printf("|   1  |   2  |   3  |   4  |   5  |   6  |   7  |   8  |   9  |  10  |\n");
140      printf(" ------------------------------------------------------------------\n");
141
142      printf("Ladders #%d=> ", choiceLadSna);
143
144      for (i = 0; i < 7; i++) {
145          printf(" | %d to %d", locationLaddersSnakes[i], jumpLaddersSnakes[i]);
146      }
147
148      printf("\nSnakes  #%d=> ", choiceLadSna);
149      for (i; i < 14; i++) {
150          printf(" | %d to %d", locationLaddersSnakes[i], (locationLaddersSnakes[i] + jumpLaddersSnakes[i]));
151      }
152      printf("\n");
153  }
154
```

PrintMenu()

```
156     //Game menu
157     void PrintMenu() {
158         choice = 0;
159         while (!choice) {
160 
161             printf("Let's go play?\n");
162             printf("--------------------------------------------------------------------------\n");
163             printf("1. New Game\n");
164             printf("2. Load Game\n");
165             printf("3. Exit \n");
166             scanf("%d", &choice);
167 
168             if (choice <= 0 || choice > 3) {
169                 choice = 0;
170                 printf("Invalid option\n");
171             }
172         }
173     }
174 
```

StartNewGame()

```
177     //Initializes the variables needed to start a new game
178     void StartNewGame() {
179         //Ask for the number of Players
180         do
181         {
182             printf("Please enter the number of Players (2 to 6) => ");
183             scanf("%d", &nPlayers);
184 
185         } while (!(nPlayers >= 2 && nPlayers <= 6));
186 
187         // create the player to get the array
188         player = (int*)malloc(sizeof(int) * nPlayers);
189 
190         // Populate the variable player to be 0 zero
191         for (i = 0; i < nPlayers; i++) {
192             *(player + i) = 0;
193         }
194 
195         // Random 1 to 10 for options board
196         choiceLadSna = randTen();
197 
198         // Populate array locationLaddersSnakes and jumpLaddersSnakes pre load values diferents positions
199         for (i = 0; i < 14; i++)
200         {
201             locationLaddersSnakes[i] = valuesLocationLaddersSnakes[choiceLadSna - 1][i];
202             jumpLaddersSnakes[i] = valuesJumpLaddersSnakes[choiceLadSna - 1][i];
203         }
204 
205         // Game Status
206         statusGame = "New Game";
207 
208         getchar();
209     }
```

LoadGame()

```
212     //Initializes the variables required to load an existing game
213     void LoadGame() {
214         int count;
215         int nGame;
216 
217         // open the file
218         cfPtr = fopen("game.txt", "r");
219         int valuePlayer[10000][8];
220         count = 0;
221         printf("Game - Players   Board  Player1  Player2  Player3  Player4  Player5  Player6");
222         printf("\n================================================================================\n");
223 
224         // Read the file and load the variable
225         while (!feof(cfPtr)) {
226             fscanf(cfPtr, "%d %d ", &valuePlayer[count][0], &valuePlayer[count][1]);
227             printf("%4d %7d %7d ", count + 1, valuePlayer[count][0], valuePlayer[count][1]);
228 
229             for (i = 0; i < valuePlayer[count][0]; i++) {
230                 fscanf(cfPtr, "%d ", &valuePlayer[count][i + 2]);
231                 printf("%8d ", valuePlayer[count][i + 2]);
232             }
233             printf("\n");
234             count++;
235         }
236 
237         //Clouse the file
238         fclose(cfPtr);
239 
240         //Ask for the number of Game to load
241         do
242         {
243             printf("\nEnter the number Game you would like to load (1 to %d) => ", count);
244             scanf("%d", &nGame);
245         } while (!(nGame >= 1 && nGame <= count));
246 
247         // create the player to get the array
248         nPlayers = valuePlayer[nGame - 1][0];
249         player = (int*)malloc(sizeof(int) * nPlayers);
250 
251         // Populate the variable player
252         for (i = 0; i < nPlayers; i++) {
253             *(player + i) = valuePlayer[nGame - 1][2 + i];
254         }
255 
256         // options board
257         choiceLadSna = valuePlayer[nGame - 1][1];
258 
259         // Populate array locationLaddersSnakes and jumpLaddersSnakes pre load values diferents positions
260         for (i = 0; i < 14; i++)
261         {
262             locationLaddersSnakes[i] = valuesLocationLaddersSnakes[choiceLadSna - 1][i];
263             jumpLaddersSnakes[i] = valuesJumpLaddersSnakes[choiceLadSna - 1][i];
264         }
265 
266         // Game Status
267         statusGame = "Load Game";
268 
269         getchar();
```

SaveGame()

```
274     //Save the game in file
275     void SaveGame() {
276         // open the file
277         cfPtr = fopen("game.txt", "a");
278 
279         // Save n. playes and number correponds board
280         fprintf(cfPtr, "%d %d ", nPlayers, choiceLadSna);
281 
282         for (i = 0; i < nPlayers; i++) {
283             fprintf(cfPtr, "%d ", *(player + i));
284         }
285 
286         // print blank line
287         fprintf(cfPtr, "\n");
288 
289         statusGame = "Saved";
290 
291         //Clouse the file
292         fclose(cfPtr);
293     }
```

Method main()

```
296  /*
297  Starts the main method
298  */
299  void main()
300  {
301      while (1) {
302
303          int inGame = 1; // Define value to 1 to stay in Game
304          int oldValue;
305
306          printLogo();  //Print the game logo
307          PrintMenu(); // Call Menu Game
308
309          // Verif option correpond in PrintMenu()
310          switch (choice) {
311          case(1) :
312              StartNewGame(); //New Game - Setup all variables to star a new game
313              break;
314          case(2) :
315
316              // open the file
317              cfPtr = fopen("game.txt", "r");
318
319              // Verified if the file not exist
320              if (cfPtr == NULL) {
321                  printf("No records are stored, start a new game.\n\n");
322
323                  keyEnter = fgetc(stdin);
324                  StartNewGame(); //New Game - Setup all variables to star a new game
325                  break;
326              }
327              else {
328                  LoadGame(); //Load Game - Setup all variables to load a saved game
329                  break;
330              }
331
332          case(3) :
333              return 0; // Exit the Game
334          default:
335              return 1;
336          }
337
```

```
338          //Keep in game until InGame == 1
339          while (inGame == 1) {
340
341              printBoard(); // Print Board for Game
342
343              //Print number of players on Screen
344              for (i = 0; i < nPlayers; i++) {
345                  printf("\nPlayer %d value is ==> %d", i + 1, *(player + i));
346              }
347
348              printf("\n--------------------------------------------------------------------------");
349
```

```c
            // This loop correpond each player for roll the Dice
            for (i = 0; i < nPlayers; i++) {
                printf("\nPlayer %d  ==> Press Enter to roll the dice", i + 1);

                //Pause waiting press ENTER to roll the dice
                keyEnter = fgetc(stdin);

                //dice = 0;
                //get value for method rollDice (1 to 6)
                dice = rollDice();

                //Storage old value player before set a new value
                oldValue = *(player + i);

                //Add a new value for the player
                *(player + i) += dice;
                printf("Player %d  ==> The dice is ==> %d <== , now your new value is %d\n", i + 1, dice, *(player + i));
```

```c
            // Verifies that the data value and 6 and if the value of the player and less than 100
            if (*(player + i) < 100 && dice == 6) {
                do {
                    //Pause waiting press ENTER to roll the dice
                    printf("Player %d  ==> Press Enter to roll the dice again ", i + 1);
                    keyEnter = fgetc(stdin);

                    //get value for method rollDice (1 to 6)
                    dice = rollDice();

                    //Storage old value player before set a new value
                    oldValue = *(player + i);

                    //Add a new value for the player
                    *(player + i) += dice;
                    printf("Player %d  ==> The dice is ==> %d <== , now your new value is %d\n", i + 1, dice, *(player + i));

                    // Verifies player got 100 to win the game
                    if (*(player + i) == 100) {
                        break; // Exit the loop
                    }

                    // If player is over 100, the player still in game and back the old value
                    if (*(player + i) > 100) {
                        *(player + i) = oldValue;
                    }

                } while (!(dice < 6) && *(player + i) >= 100); // Loop different value of 6 and greater than 100
            }
```

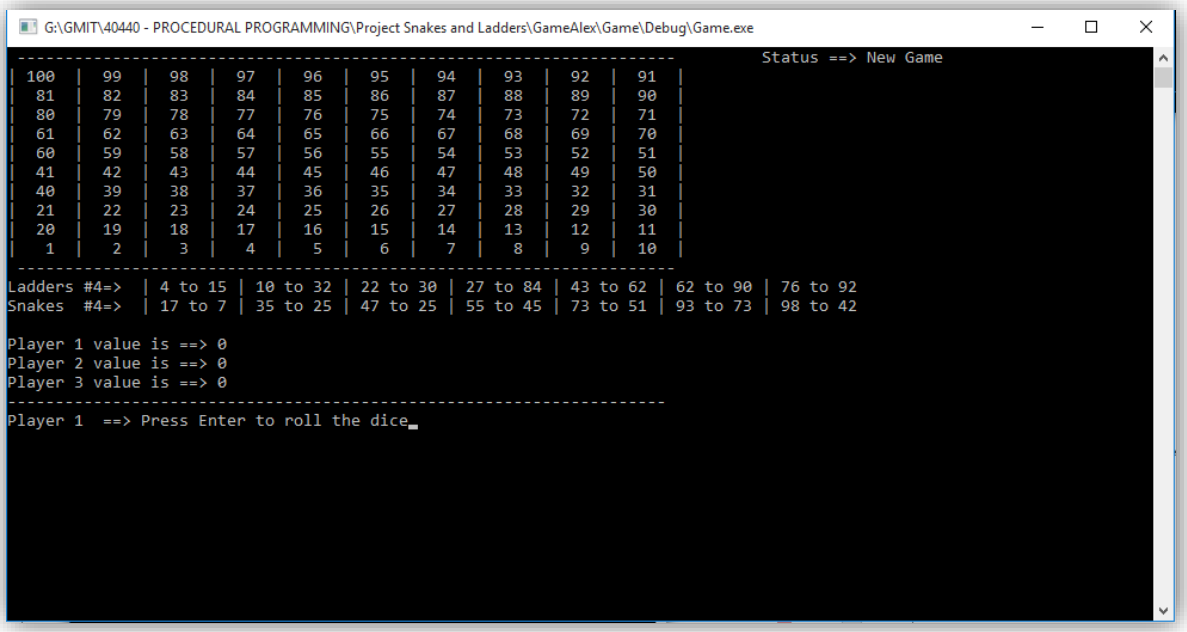
```c
            // Checks if the player has reached the value 100
            if (*(player + i) == 100) {

                printf("\n*****************************************************************");
                printf("\n**************   Player %d  ==> IS THE WINNER :)  **************", i + 1);
                printf("\n*****************************************************************\n\n");
                i = nPlayers;
                inGame = 0;
            }

            // If player is over 100, the player still in game and back the old value
            else if (*(player + i) > 100) {
                *(player + i) = oldValue;
            }

            //This loop checks if the player has reached a value that correponde the head of the snake or ladder
            for (j = 0; j <= 7; j++) {

                if (*(player + i) == locationLaddersSnakes[j]) {
                    oldValue = *(player + i);

                    if (jumpLaddersSnakes[j] < 1) {
                        *(player + i) += jumpLaddersSnakes[j];
                        printf("Player %d  ==> You got a Snake to back from %d to %d\n", i + 1, oldValue, *(player + i));
                    }
                    else {
                        *(player + i) = jumpLaddersSnakes[j];
                        printf("Player %d  ==> You got a ladder to jump from %d to %d\n", i + 1, oldValue, *(player + i));
                    }
                }
            }
```

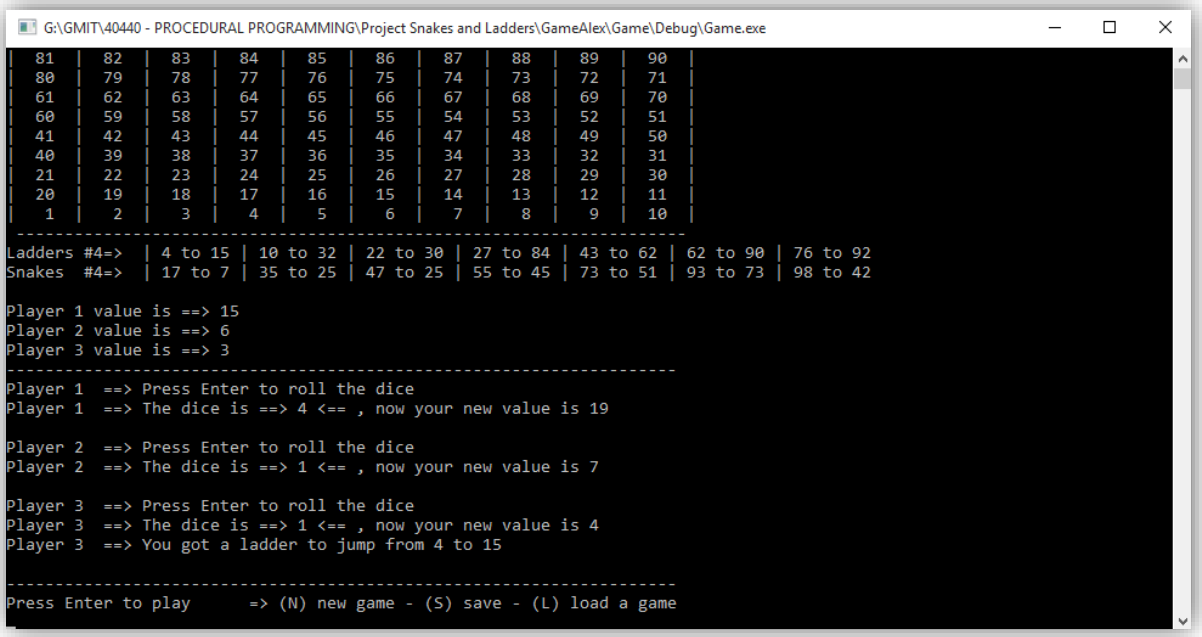
```
431             // Finsh the firt round - this stage the player can start a new game, save or Load a game
432             printf("\n--------------------------------------------------------------------------------");
433             printf("\nPress Enter to play      => (N) new game - (S) save - (L) load a game\n");
434
435             // loop waiting press any key
436             while (!kbhit()) {}
437
438             int ch;
439             ch = getch(); // Storage the key pressed
440
441             // Correpond a new game (n or N)
442             if (ch == 78 || ch == 110) // Represents N and n in ASCII table
443             {
444                 clrscr();       //Cleanner Screen
445                 StartNewGame(); //New Game
446             }
447
448             // Correpond save game (s or S)
449             if (ch == 83 || ch == 115) // Represents S and s in ASCII table
450             {
451                 SaveGame(); //Save the Game
452             }
453
454             // Correpond load game (l or L)
455             if (ch == 76 || ch == 108) // Represents L and l in ASCII table
456             {
457                 clrscr();   //Cleanner Screen
458                 LoadGame(); //Load the Game
459             }
460         }
461     }
462 } //End main
463
```

TESTING THE GAME

New Game

```
Let's go play?
--------------------------------------------------------------------------
1. New Game
2. Load Game
3. Exit
1
Please enter the number of Players (2 to 6) => 3
```





Load Game

```
Let's go play?
--------------------------------------------------------------------
1. New Game
2. Load Game
3. Exit
2
Game - Players   Board  Player1  Player2  Player3  Player4  Player5  Player6
=========================================================================
   1       2       3       19       10
   2       2       3       27       35
   3       6       1       16       17       17       14       31       11
   4       3       4       19        7       15

Enter the number Game you would like to load (1 to 4) => 2
```

```
G:\GMIT\40440 - PROCEDURAL PROGRAMMING\Project Snakes and Ladders\GameAlex\Game\Debug\Game.exe
 ------------------------------------------------------------------          Status ==> Load Game
| 100  |  99  |  98  |  97  |  96  |  95  |  94  |  93  |  92  |  91  |
|  81  |  82  |  83  |  84  |  85  |  86  |  87  |  88  |  89  |  90  |
|  80  |  79  |  78  |  77  |  76  |  75  |  74  |  73  |  72  |  71  |
|  61  |  62  |  63  |  64  |  65  |  66  |  67  |  68  |  69  |  70  |
|  60  |  59  |  58  |  57  |  56  |  55  |  54  |  53  |  52  |  51  |
|  41  |  42  |  43  |  44  |  45  |  46  |  47  |  48  |  49  |  50  |
|  40  |  39  |  38  |  37  |  36  |  35  |  34  |  33  |  32  |  31  |
|  21  |  22  |  23  |  24  |  25  |  26  |  27  |  28  |  29  |  30  |
|  20  |  19  |  18  |  17  |  16  |  15  |  14  |  13  |  12  |  11  |
|   1  |   2  |   3  |   4  |   5  |   6  |   7  |   8  |   9  |  10  |
 ------------------------------------------------------------------
Ladders #3=>  | 6 to 17 | 15 to 31 | 23 to 33 | 28 to 89 | 44 to 66 | 66 to 92 | 70 to 95
Snakes  #3=>  | 16 to 6 | 33 to 23 | 45 to 23 | 58 to 48 | 75 to 53 | 91 to 71 | 95 to 39

Player 1 value is ==> 27
Player 2 value is ==> 35
--------------------------------------------------------------------
Player 1  ==> Press Enter to roll the dice
```

REFERENCES

To develop this project, we referred to some labs:

- Lab06
- Lab07
- Lab08
- Lab09
- Lab10

I also use google to conduct some research in order to solve some doubts.