Design Notes

This system utilises a doubly linked list of squares which poses an action construct. Normal squares are not defined to save space. The source file gameBoard.c contains all of the methods required to manipulate a game board from a supplied text sequence as per the specification (SINGLE space separated values).

The main.c file calls the functionality from the other source files in order to interpret and display the game board correctly.

The program should be supplied with the source file, this is then interpreted line by line until the end line is detected. At this point the data structure that was created (a game board) is sent to the ‘print_game_board’ function. The ‘print_game_board’ function then outputs the game board to the standard output stream, after this the program releases its memory and exits.

A additional file linkedListHelper.C holds the functions for manipulating the square structures linked list.

The inputRedHandlers file handles the receiving of the input to the program, these functions are then used by the interpreter present in the ‘gameboard.c’ file. Breaking the reading functionality into its own functions reduces the code duplication in the application and allows for the program to grow more easily.

Key Structures:

The square structure represents a single square in the game board, generally plain squares (squares that are only a number with no action) are not modeled in the systems data structures. Only squares that have actions (snake or ladder) are listed in the game boards structure squares linked list. Here the system can be extended to include attributes such as square colour.

```
typedef struct square{
  int no;
  struct square *next_square;
  struct square *prev_square;
  struct action *action;

} square;
```

The board structure represents a game board, holds the size of each dimension of the board and the pointers to the head and tail of the linked list of action squares that are in the board.

```
typedef struct board{
   int xMax;
   int yMax;
   int nMax;
   square *action_Squares_Head;
   square *action_Squares_Tail;

} board;
```

Action structure models the actions that a square can exhibit, such as the destination of the action and the type as a enum.

```
typedef struct action{
   int destination_Square_No;
   enum action_Types action_type;
} action;
```

Actions available on a square, it can be a snake or a ladder or the final square in the board.

```
typedef enum action_Types { SNAKE , LADDER, ENDOFBOARD} action_Types;
```

Data structure used during the printing of the board to the screen to track the progress between the lines.

```
typedef struct board_Print_Data{
   int index;
   int line_Length;
   square *ptr_To_Square;
   board *game_board;

} board_Print_Data;
```

The source files are commented with explanations as to the function of all functions, refer to these for detail.