```c
#include "decl.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/*Stores the current number of generations*/
static int currentGeneration = 1;

/**
 * Outputs the current state of the grid.
 * @param grid The grid to print.
 * @param rows The number of rows in the grid.
 * @param cols The number of columns in the grid.
 */
static void display(char *grid, int rows, int cols)
{
    char temp;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            temp = *(grid + i * rows + j);
            printf("\t%c ", temp);
        }
        printf("\n");
    }
    printf("\n");
}

/**
 * Displays n generations to the player
 * @param state The current state of the game
 * @param nextState The next state of the game
 * @param gen The number of generations to produce
 * @param rows The number of rows in the grid
 * @param cols The number of columns in the grid
 * @param info The PlayerInfo struct to update as needed.
 * @param displayInitialState Determines if the initial state of the board should be displayed
 */
void displayGenerations(char *state, char *nextState, int gen, int rows, int cols, struct PlayerInfo *info, bool displayInitialState)
{
    char buf[BUFFER_SIZE];
    int additional = 0;
    bool isValid = FALSE;

    /*Display the initial state*/
    if (displayInitialState) {
        printf("Initial State: \n");
        display(state, rows, cols);
    }

    for (int i = 0; i < gen; i++) {
        /* Prevent redundant computation if all the cells are inactive by terminating */
        if (!generations(state, nextState, rows, cols)) {
            printf("After generation %d, all cells in the board are inactive.\n", i);
            exit(1);
        }
        /* The next state becomes the current state after each iteration*/
        memcpy(state, nextState, sizeof(char) * rows * cols);
        printf("Generation %d: \n", currentGeneration++);

        /* Display the current state after updating */
        display(state, rows, cols);
    }

    printf("Would you like to see more? Enter 'yes' or 'no' >> ");
    scanf("%s", buf);
    isValid = (strcmp(buf, "yes") == 0 || strcmp(buf, "no") == 0);

```

```
68      /*Prompt the user for a valid response if necessary*/
69      while (!isValid) {
70          printf("Please enter 'yes' or 'no' >> ");
71          scanf("%s", buf);
72          isValid = (strcmp(buf, "yes") == 0 || strcmp(buf, "no") == 0);
73      }
74
75      if (strcmp(buf, "yes") == 0) {
76          printf("Enter the number of generations >> ");
77          scanf("%s", buf);
78          isValid = sscanf(buf, "%d", &additional);
79
80          /*Prompt the user for a valid response if necessary*/
81          while (!isValid) {
82              printf("Please enter a valid Integer >> ");
83              scanf("%s", buf);
84              isValid = sscanf(buf, "%d", &additional);
85          }
86
87          /*Update the number of generations in the struct*/
88          info->numGenerations += additional;
89
90          /*Recurse for the additional generations*/
91          displayGenerations(state, nextState, additional, rows, cols, info, FALSE);
92      } else {
93          return;
94      }
95  }
```