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

/**
    Initializes the game board
    @param info The information about the player and instance
*/
void boardInitialization(struct PlayerInfo *info)
{
    FILE *inFile = fopen("state.txt", "r");
    char c;
    char buffer[BUFFER_SIZE];
    char *initialState, *nextState;
    int rows = 0, cols = 0, temp = 0, index = 0;

    /*
        Read in the dimensions of the game grid from the input file.
        If the number of dimensions isn't formatted properly, terminate the
        program to prevent errors.
    */
    if (fscanf(inFile, "%d %d\n", &rows, &cols) != 2) {
        fprintf(stderr, "Error: Expected Dimensions to be specified on Line 1");
        exit(0);
    }

    /*
        Allocate two contiguous blocks of memory to store the game grid as well
        as the next state.
        I am using characters as opposed to integers so as to save space.
        A character only uses a single byte as opposed to 4 bytes.
    */
    initialState = calloc(rows*cols, sizeof(char));
    nextState = calloc(rows*cols, sizeof(char));

    /*While there is a character to be read from the file, parse it*/
    while ((c = fgetc(inFile)) != EOF) {
        /*If there are more cells than the allocated amount, terminate the program*/
        if (index > (rows * cols)) {
            fprintf(stderr, "Error: Invalid amount of cells");
            exit(1);
        }

        switch(c)
        {
            case ALIVE_CELL:
            case DEAD_CELL:
                temp = c;
                break;
            case '\n':
            case ' ':
                continue;
            default:
                fprintf(stderr, "Error: invalid character %c encountered. Terminating
                execution.", c);
                exit(1);
        }

        /*The next entry is located at the base address + the offset (index)*/
        *(initialState + index++) = temp;
    }

    /*Compute and display the user specified number of generations*/
    displayGenerations(initialState, nextState, info->numGenerations, rows, cols, info,
    TRUE);

    /*
        After all the generations have been displayed, free the allocated memory so as
        to not waste resources and
```

```
67              leak memory
68          */
69          free(initialState);
70          free(nextState);
71      }
```