CSCE 121 Team Project Specification – “15-Puzzle”

Dr. Daugherity

Due: 11:59 P.M. Thursday, November 30, 2017

The project is to design and write a C++17/FLTK computer game program with a graphical user interface. The object of the game is to move numbered tiles into consecutive order in as few moves as possible.

The program must:

1. Start with an attractive splash screen showing (at least) the name of the game, the team number and name, and the team members' names. Feel free to add team members' pictures, play music, show a game being played (see “Extra Items” below), etc. Have a START button which then explains how to play the game.
2. Ask for a difficulty level from 1 to 4 (which you may name, *e.g.*, beginner-intermediate-advanced-expert or page-squire-knight-king); this determines the number of moves necessary to finish the game (10/20/40/80 moves, respectively). Display both the game (see below) and the top 5 scores for that difficulty level (read in from the disk; the top scores file starts out empty).
3. Ask for the player's initials and display them with a blank score below the top 5 scores.
4. Show how many moves remain for that difficulty level and also how many tiles are not in the correct final position. Instruct the player to click on a tile to move it to the blank square. Move the selected tile and update the number of moves remaining. Remember to redraw the screen!
5. Have a HINT button to recommend a move based on the “Manhattan distance” formula. Here is the formula: Depending on the position of the blank, there are 2, 3, or 4 possible moves. For each of the possible moves, pretend the move has been made and add up the number of rows and columns each of the 15 tiles would still have to be moved to reach its final position, then recommend the move with the lowest sum. (Obviously a sum of 0 means that move finishes the game with the highest possible score.)
6. When there are no remaining moves, the game is over and the player’s score is the number of tiles in their correct final positions times the number of moves. For example, if there are 11 tiles in their correct positions at the end of a 10-move game, the score is 110. Add the player’s score and initials to the scores list, sort the list of 6 scores, and write the top 5 out to disk with initials. Then the next time the game is played, that file will be read in and displayed in step 2 above. Ask the player if they want to play another game or quit.

Here are the games for each level:

10 moves

|  |  |  |  |
| --- | --- | --- | --- |
|  | 12 | 11 | 13 |
| 15 | 14 | 10 | 9 |
| 3 | 7 | 6 | 2 |
| 4 | 8 | 5 | 1 |

20 moves

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 5 | 1 | 4 |
| 10 | 7 | 3 | 8 |
|  | 13 | 11 | 12 |
| 6 | 9 | 14 | 15 |

40 moves

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | 5 | 11 | 4 |
| 10 | 13 | 2 | 1 |
| 9 | 15 | 7 | 3 |
| 14 | 12 | 8 |  |

80 moves

|  |  |  |  |
| --- | --- | --- | --- |
|  | 12 | 11 | 13 |
| 15 | 14 | 10 | 9 |
| 3 | 7 | 6 | 2 |
| 4 | 8 | 5 | 1 |

The final position for each game is

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 |  |

EXTRA ITEMS

* On the splash screen show a game being played. This could be a simple loop which pauses one second between pictures and five seconds at the end.
* Make up 3 more games for each level (making sure it has the correct minimum number of moves) and then when a level is chosen, randomly select one of the 4 games for that level.
* Make the background of each tile green or red depending on whether it is in the correct final position or not. Make the numbers white so they will be readable.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This is a team project, with three or four students on a team. The instructor will assign the teams. (Note: If there are any problems with your team assignment, please talk to your TA promptly.) Choose a team leader who will coordinate the project.

Divide up the code, with each student on a team of 3 doing two of the six items on page 1. If you have a fourth team member, that person must do one of the extra items.

A team of 3 will receive 5 points extra credit for doing one "Extra Item" or 10 points for doing two. A team of 4 will receive 5 points extra credit for doing a second "Extra Item" or 10 points for doing all three “Extra Items.”

You must use at least two C++11 or C++14 or C++17 features, such as auto and range-based for. (See the reference pages on <http://cplusplus.com/> to see if a feature is C++11 or C++14 or C++17.) You must write at least two classes of your own, with separate header and implementation files. Follow good style, and limit each function to no more than 24 lines (one terminal window). Each team member is expected to have a rough idea of how all the code works, and should be able to explain in detail how their own part of the code works.

Choose a clever name for your team (but keep it clean ).

Be creative in deciding how to meet these specifications in an attractive and user-friendly way, but get the basic functionality working before you try to make it too fancy, or you may run out of time!

All user input and output must be through the GUI, not the console window. However, you may use the console window for printing debugging messages for the developers (your team).

Your program should compile and run without change on the Visual C++ 2017 environment in the lab or build.tamu.edu with g++ -std=c++17 and X windows.

Your program must be submitted both to CSNET and also on a CD or DVD. The project report (described below) should be submitted on paper to your TA, along with your CD or DVD. You only need to submit one report and CD or DVD and CSNET file per team. Write a report according to the outline below. All team members will receive the same project grade, unless some team member does not do his/her part (see report outline below).

**Important! You must demonstrate your project to your TA or it will not be graded!**

Note about teamwork: Immediately exchange contact information with your teammates and schedule times to meet and work on the project. If your teammates are in the same lab section, you have the two hours per week during lab, but in any case you need to schedule meeting times outside of lab. Note that the lab is actually available for the full 75-minute period on the Tuesday/Thursday class schedule, so (for example) if your lab is 9:35-10:25 you can stay until 10:50. As meeting scheduling can be difficult, use this lab time wisely! Attendance will be taken during lab, so that complaints of "We could never find a time to meet" will not be taken seriously.

REPORT OUTLINE

The project report must be printed on a laser printer. The report should include the following sections:

1. Team information (team name, members' names, who did what, did each member

do a fair share of the work)

2. Statement of the problem, significance, etc.

3. Restrictions and limitations

4. Explanation of your approach (analysis to choose a strategy for programming the project, how you coded it, etc.)

5. Sample run (screen shots)

6. Results and analysis

7. Conclusions - What did you show? What did you learn?

8. Future research (how your program could be improved or extended)

9. Instructions on how to run your program

10. Listing of the program, which must have adequate comments

11. Bibliography - references used, if any