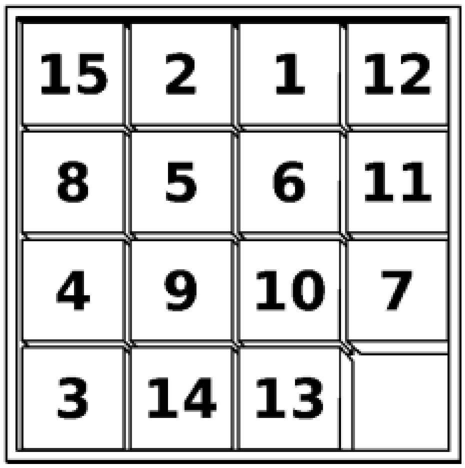
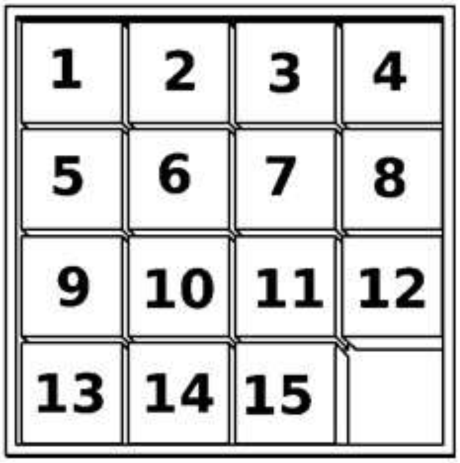
PROJECT DESCRIPTION

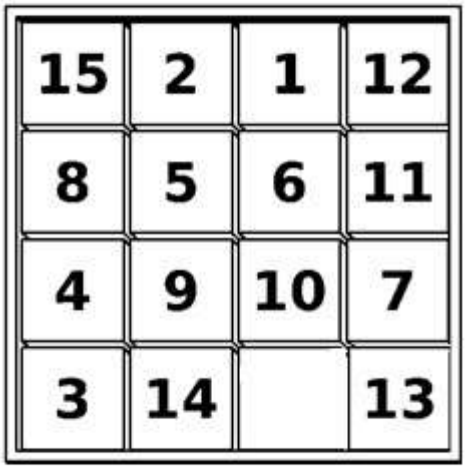
This project is about developing a sliding puzzle game. The sliding puzzle is a game composed of a number of pieces dispersed randomly along a board and one empty square. You can slide pieces by using the empty square. The objective is to rearrange these pieces to their original form where the distribution of pieces is on ascending order.

Figure 1-a shows an example of sliding puzzle game with 15 pieces numbered randomly from 1 to 15 which lay in a square frame (4 rows x 4 columns). The goal is to rearrange these numbers in ascending order by sliding pieces using the free square as shown in figure 1-b.

The score of the game is defined as the number of movements to solve the puzzle game.

Sliding puzzle board with 4 rows and 4 columns.	
 15 2 1 12 8 5 6 11 4 9 10 7 3 14 13 Figure 1-a: Numbers are mixed randomly	 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 Figure 1-b: Solved puzzle game (all numbers are in order)

Basically, the game allows a player to enter a number that represents a tile next to the blank square, and then moves this tile to the blank square. For example, considering figure 1-a, if a player enters the value 13, then the board will be changed as shown in figure 2.



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

Figure 2 – Puzzle board after moving tile 13

As shown in figure 1-b, to succeed, the player must move the tiles until the numbers are arranged in ascending order from left to right and then from top to bottom with the blank square in the lower right corner. The program should calculate and display the score of the player that is his/her number of moves in the game.

Detailed Requirements

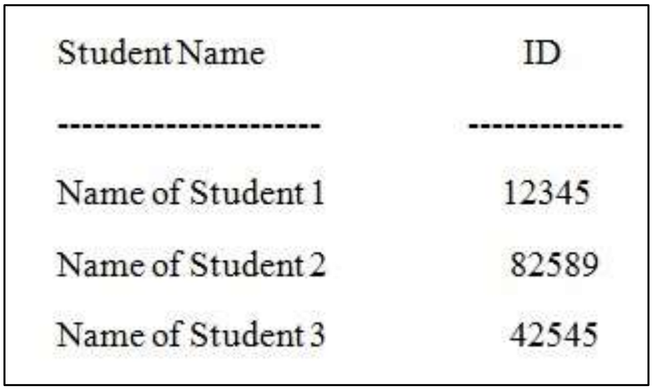
Create a Console based C program to achieve the below requirements:

1. Provide a **Welcome screen** containing the title of the program, which should appear in the middle of the screen, followed by the main menu.

2. After the welcome screen, the program should provide the user with the main menu as well as the **submenus** so that the user can navigate through the menu options based on his/her selection.

3. The main menu and its sub-menus should have the following options:

   A. Team members: Call a function to display students information in a group following the below format (It should appear in the middle of the screen).



```
Student Name                    ID
---------------------          -------------
Name of Student 1               12345
Name of Student 2                82589
Name of Student 3                42545
```

B. Game instructions: Call a function to display a brief description of how to go about playing the sliding puzzle game.

C. Play game: Call a function to display the following menu to choose the difficulty level of the game. The game has three different levels of difficulty: beginner, intermediate, and advanced. Once a level is chosen, the game will start.

   1. Beginner level
   2. Intermediate level
   3. Advanced level
   4. Back to main menu

D. Exit: this option exits the program.

4. The size of puzzle game board is 3×3 for beginner level, 4×4 for intermediate, and 5×5 for advanced.

5. The numbers associated with the tiles depend on the game's level.

   - Beginner level: Numbers should be between 1 and 8.
   - Intermediate level: Numbers should be between 1 and 15. Advanced level: Numbers should be between 1 and 24.

6. Create three two-dimensional arrays named:

    I. **BoardPuzzle1** of size 3x3 for beginner level.
    II. **BoardPuzzle2** of size 4x4 for intermediate level. III.
    BoardPuzzle3 of size 5x5 for advanced level.

7. Initialize the arrays with random mixed numbers using the function **rand ()**. Note that none of the arrays can contain repeated numbers.

    - 1 to 8 for beginner level.
    - 1 to 15 for intermediate level.
    - 1 to 24 for advanced level.
        - **<u>Optional choice applicable to advanced level only:</u>** The 5x5 array can be hard coded as shown in figure 3.

1	**2**	**3**	**4**	**5**
11	**6**	**7**	**8**	**9**
16	**12**	**13**	**14**	**10**
0	**17**	**18**	**19**	**15**
21	**22**	**23**	**24**	**20**

Figure 3 – Initial board game for advanced level

8. The blank square (highlighted in grey) should be initialized with 0.

<table>
<tr><th>Beginner level</th><th>Intermediate level.</th><th>Advanced level.</th></tr>
<tr>
<td>
<table>
<tr><td>7</td><td>1</td><td>5</td></tr>
<tr><td>4</td><td>0</td><td>3</td></tr>
<tr><td>8</td><td>6</td><td>2</td></tr>
</table>
</td>
<td>
<table>
<tr><td>8</td><td>12</td><td>6</td><td>7</td></tr>
<tr><td>11</td><td>13</td><td>15</td><td>14</td></tr>
<tr><td>0</td><td>2</td><td>5</td><td>3</td></tr>
<tr><td>1</td><td>9</td><td>10</td><td>4</td></tr>
</table>
</td>
<td>
<table>
<tr><td>19</td><td>20</td><td>4</td><td>22</td><td>12</td></tr>
<tr><td>18</td><td>21</td><td>15</td><td>5</td><td>13</td></tr>
<tr><td>6</td><td>16</td><td>14</td><td>24</td><td>23</td></tr>
<tr><td>17</td><td>0</td><td>11</td><td>2</td><td>3</td></tr>
<tr><td>7</td><td>8</td><td>1</td><td>9</td><td>10</td></tr>
</table>
</td>
</tr>
</table>

Figure 4 – Example of initialized arrays

9. Once the player starts playing the game, the program must clear the screen and display the game board.
   The program should then ask the player to enter a number next to the blank square to make the next move.
   It should also check whether the number is located next to the blank element (that is zero). In this case, the program should replace the zero with the entered number and vice-versa. Otherwise, it should display an “error message”. These steps must be repeated until all numbers are sorted.

10. The game should not allow the player to do any invalid moves. The player is only allowed to swap the blank square with any of the surrounding numbers.

11. After each valid move the program should check whether the level is completed. If yes, the program should display the number of moves (the score) made to win the game along with a “Congratulations” message. After this step, the level’s submenu should be displayed again. Though, if the level is incomplete, the program should keep asking the player for more moves/numbers.

12. After each valid move, the program should clear the screen and show the updated array.

13. The player has to enter **-1** if s/he wishes to exit a game level and return to the submenu.