MEMORY MATCH MADNESS:

A CONSOLE-BASED MEMORY GAME IN C

SUBMITTED BY:

ROLL NO: (CT-25064)

ROL NO: (CT-25069)

SUBMITTED TO:

Mr. MUHAMMAD ABDULLAH
COURSE CODE: CT-175
COURSE TITLE: PROGRAMMING FUNDAMENTALS

SEMESTER:

1ST SEMESTER FALL 2025

DATE OF SUBMISSSION:

12-11-2025

DEPARTMENT:

INSTITUTE NAME:

INDEX:

:

ABSTRACT:

. The game challenges the player’s memory and concentration

---

1. INTRODUCTION:

---

2. OBJECTIVES:

---

3. BACKGROUND:

ogramming — foundational to modern software development.

---

4. SYSTEM REQUIREMENTS:

---

5. CODE

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Function to clear the console screen (works on Windows & Linux)
void clearScreen() {
#ifdef _WIN32
   system("cls");
#else
   system("clear");
#endif
}

int main() {
   int board[6][6], shown[6][6];
   int size, totalScore = 0;
   int i, j;

   // Title
   clearScreen();
   printf("=======================================\n");
   printf("       MEMORY MATCH MADNESS\n");
   printf("=======================================\n");
   printf("Match all pairs to win!\n");
   printf("Scoring: +10 for a match, -2 for a wrong guess\n\n");

   // Choose difficulty
   printf("Choose difficulty:\n");
   printf("1) Easy (2x2)\n2) Medium (4x4)\n3) Hard (6x6)\n> ");
   int level;
   scanf("%d", &level);
   while (getchar() != '\n');
```

```
if (level == 1)
size = 2;
else if (level == 2)
size = 4;
else size = 6;

// Start from chosen level to Hard
for (int lvl = level; lvl <= 3; lvl++) {
   if (lvl == 1)
      size = 2;
   else if (lvl == 2)
      size = 4;
   else
      size = 6;

   // Initialize shown array
   for (i = 0; i < size; i++)
      for (j = 0; j < size; j++)
         shown[i][j] = 0;

   int total = size * size;
   int pairs = total / 2;
   int nums[36];
   int k = 0;

   // Fill pairs
   for (i = 1; i <= pairs; i++) {
      nums[k++] = i;
      nums[k++] = i;
   }

   // Shuffle numbers
```

```
        srand(time(NULL));
        for (i = total - 1; i > 0; i--) {
            int r = rand() % (i + 1);
            int temp = nums[i];
            nums[i] = nums[r];
            nums[r] = temp;
        }

        // Fill the board
        k = 0;
        for (i = 0; i < size; i++)
            for (j = 0; j < size; j++)
                board[i][j] = nums[k++];

        // Game loop
        int found = 0, score = 0, tries = 0;
        while (found < pairs) {
            clearScreen();
            printf("Level %dx%d | Total Score: %d | Attempts:
%d\n",size, size, totalScore + score, tries);
            // Print board
            printf("\n    ");
            for (j = 0; j < size; j++)
                  printf("%3d", j + 1);
            printf("\n   +");
            for (j = 0; j < size; j++)
                  printf("---");
            printf("\n");

            for (i = 0; i < size; i++) {
                printf("%2d |", i + 1);
                for (j = 0; j < size; j++) {
                    if (shown[i][j])
```

```
                printf("%3d", board[i][j]);
            else
                printf("%3s", "*");
        }
        printf("\n");
    }

    int r1, c1, r2, c2;

    // First card
    while (1) {
        printf("\nSelect first card:\n Row (1-%d): ", size);
        scanf("%d", &r1);
        printf(" Col (1-%d): ", size);
        scanf("%d", &c1);
        while (getchar() != '\n');
        r1--; c1--;
        if (r1 < 0 || r1 >= size || c1 < 0 || c1 >= size) {
            printf("Invalid position!\n");
            continue;
        }
        if (shown[r1][c1]) {
            printf("Already revealed!\n");
            continue;
        }
        break;
    }

    shown[r1][c1] = 1;
    clearScreen();

    // Show after first pick
    printf("Level %dx%d | Total Score: %d | Attempts: %d\n",
```

```
      size, size, totalScore + score, tries);
printf("\n   ");
for (j = 0; j < size; j++)
      printf("%3d", j + 1);
printf("\n  +");
for (j = 0; j < size; j++)
      printf("---");
printf("\n");
for (i = 0; i < size; i++) {
   printf("%2d |", i + 1);
   for (j = 0; j < size; j++) {
      if (shown[i][j])
                  printf("%3d", board[i][j]);
      else
                  printf("%3s", "*");
   }
   printf("\n");
}

// Second card
while (1) {
   printf("\nSelect second card:\n Row (1-%d): ", size);
   scanf("%d", &r2);
   printf(" Col (1-%d): ", size);
   scanf("%d", &c2);
   while (getchar() != '\n');
   r2--; c2--;
   if (r2 < 0 || r2 >= size || c2 < 0 || c2 >= size) {
      printf("Invalid position!\n");
      continue;
   }
   if (r1 == r2 && c1 == c2) {
      printf("Same card! Try again.\n");
```

```
            continue;
        }
        if (shown[r2][c2]) {
            printf("Already revealed!\n");
            continue;
        }
        break;
    }

    shown[r2][c2] = 1;
    tries++;
    clearScreen();

    // Display board again
    printf("Level %dx%d | Total Score: %d | Attempts: %d\n",
          size, size, totalScore + score, tries);
    printf("\n    ");
    for (j = 0; j < size; j++)
            printf("%3d", j + 1);
    printf("\n   +");
    for (j = 0; j < size; j++)
            printf("---");
    printf("\n");
    for (i = 0; i < size; i++) {
        printf("%2d |", i + 1);
        for (j = 0; j < size; j++) {
            if (shown[i][j])
                        printf("%3d", board[i][j]);
            else
                        printf("%3s", "*");
        }
        printf("\n");
    }
```

```
        // Check match
        if (board[r1][c1] == board[r2][c2]) {
            printf("\nMATCH! +10 points.\n");
            score += 10;
            found++;
        } else {
            printf("\nNot a match! -2 points.\n");
            score -= 2;
            if (score < 0)
                    score = 0;
            shown[r1][c1] = 0;
            shown[r2][c2] = 0;
        }

        printf("\nPress Enter to continue...");
        getchar();
        clearScreen(); // <-- clears after every try
    }

    printf("\nLEVEL COMPLETE! Score gained: %d\n", score);
    totalScore += score;
    printf("Press Enter to continue...");
    getchar();
}

clearScreen();
printf("=======================================\n");
printf("    GAME COMPLETE! FINAL SCORE\n");
printf("=======================================\n");
printf("Total Score: %d\n", totalScore);
printf("Thanks for playing!\n");
```

```
    return 0;
}
```

SYSTEM DESIGN:

– handles user input and validation.

– manages board setup, shuffling, and matching.

– outputs the grid and score information.

– coordinates levels, score, and time tracking.

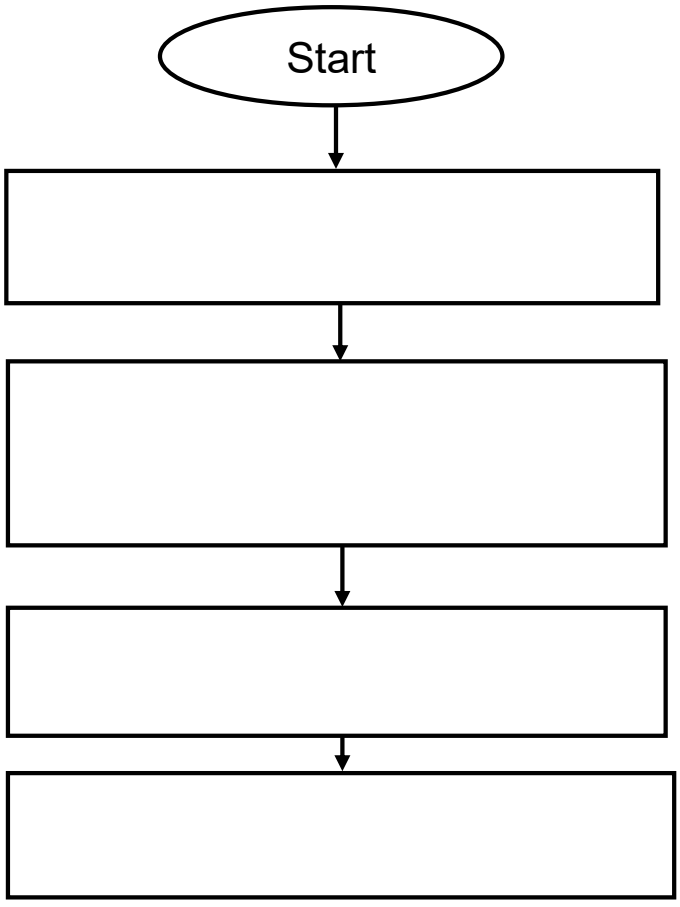
---

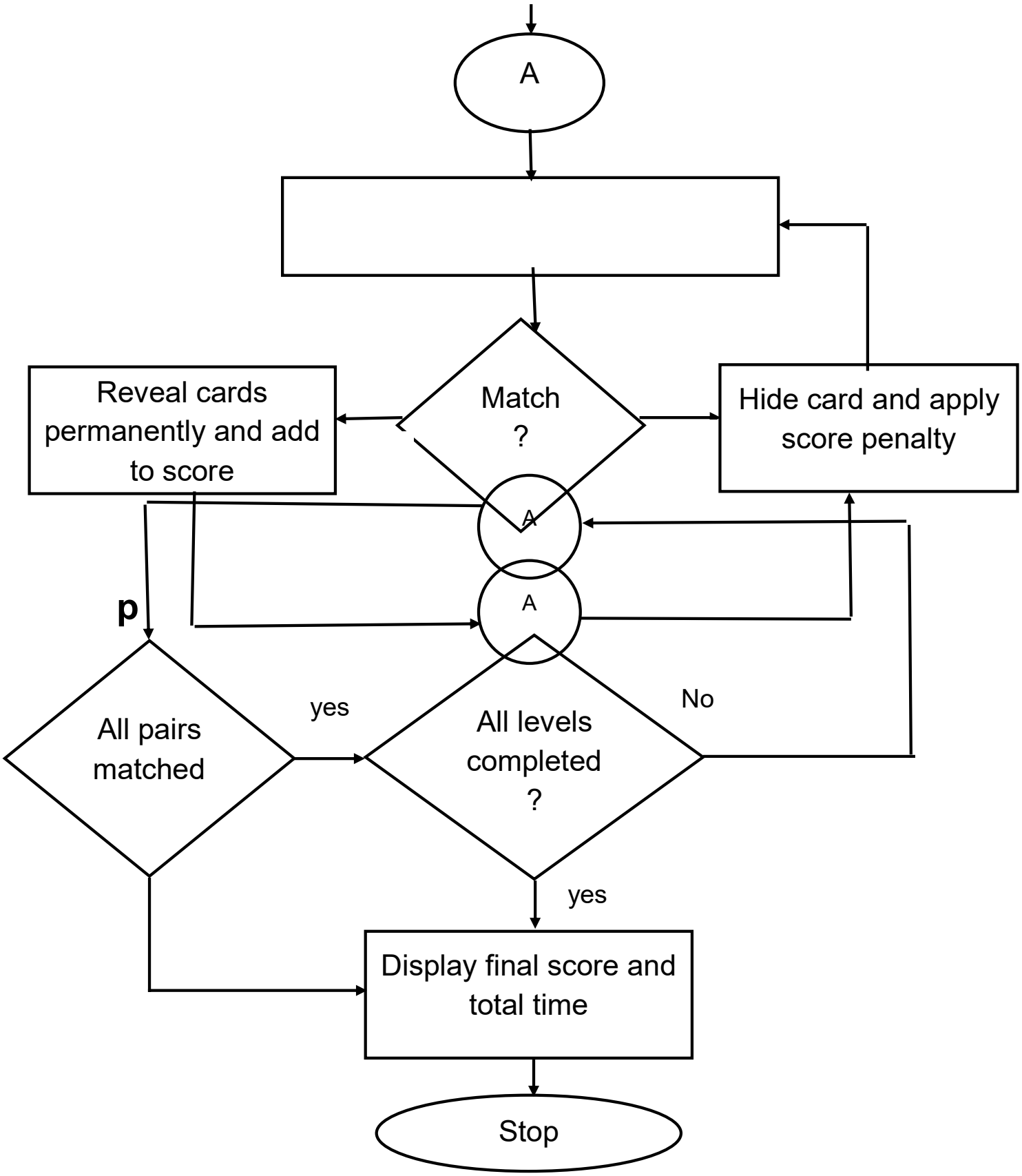
tart.

1
0
1
1
12. Stop.
.

.

Flowchart:




A
Match ?
Reveal cards permanently and add to score
Hide card and apply score penalty
A
A
p
All pairs matched
yes
All levels completed ?
No
yes
Display final score and total time
Stop

6.IMPLEMENTATION DETAILS:

HEADERS AND THEIR FUNCTIONS:

The stdio.h header file in C is responsible for handling input and output operations in a program. Its functions include:

STANDARD LIBRARY:

The stdlib.h header file in C stands for standard library and it is responsible and provide functions for general purpose tasks like

- Memory management (malloc(),calloc(),free()).
- Program control (exit(), abort()).
- Random numbers (rand(), srand()).

FUNCTIONS:

isisuseudeto

The time.h header file in C is used for date and time operations. It provides the function to get the current time intervals, and format

FUNCTIONS:

is used

TESTING AND OUTPUT:

TEST CASES:

Test ID	Test Description	Input / Action	Expected Output	Pass Criteria
TC01	Launch game	Run the program memory match	Title screen and difficulty menu appear	Game starts without crash
TC02	Select Easy difficulty	Choose option 1	Game starts with 2×2 grid	Easy level board displayed
TC03	Select Medium difficulty	Choose option 2	Game starts with 4×4 grid	Medium board (4×4) displayed

It uses preprocessor directives (`#ifdef _WIN32`, `#else` and `#endif`) to determine the operating system at compile time and execute the appropriate system command (`cls` for Windows) to refresh the display for a cleaner user interface.

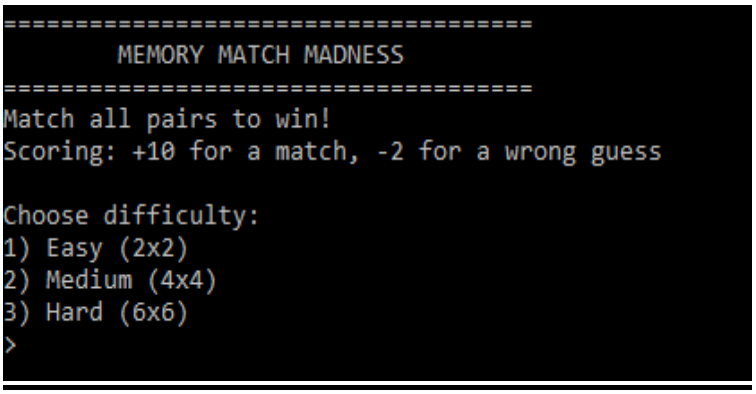
TC04	Select Hard difficulty	Choose option 3	Game starts with 6×6 grid	Hard board (6×6) displayed
TC05	Invalid difficulty input	Enter a or 5	Error message shown: “Invalid input” or “Please enter 1, 2, or 3.”	Input validated, reprompt displayed
TC06	Select unrevealed valid card	Enter coordinates in range (e.g., Row:1, Col:1)	Card value revealed temporarily	Correct tile shown

TC07	Select same card twice	Choose same coordinate twice	Message “You selected the same card twice. Turn cancelled.”	Turn cancelled properly
TC08	Select already revealed card	Pick a tile that’s already revealed	Message “That card is already revealed.”	Turn cancelled, no crash
TC09	Matching pair	Select two tiles with same number	Message “It’s a MATCH! +10 points.”	Both remain revealed, +10 score
TC10	Non-matching pair	Select two different tiles	Message “Not a match. -2 points.”	Both hidden again, -2 score
TC11	Negative score check	Miss multiple pairs	Score never goes below zero	Score >= 0 always

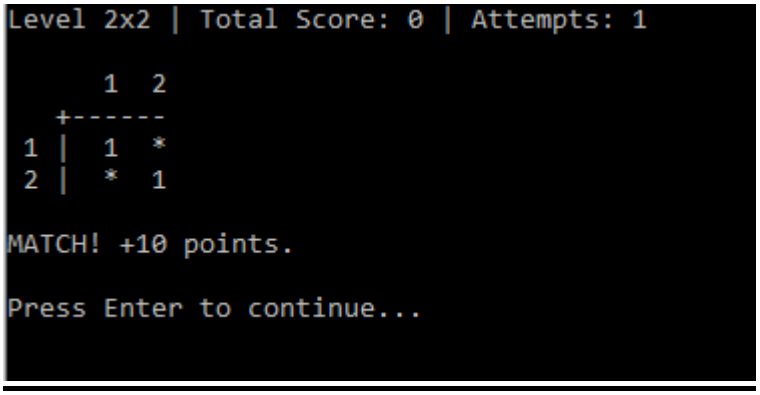
TC12	Level completion	Match all pairs	“LEVEL COMPLETE” summary displayed	Moves to next level
TC13	Multi-level progression	Start from Easy and complete up to Hard	Three level summaries, final screen shown	Correct progression order
TC14	Timing check	Let timer run for a while	Time in seconds increases properly	Timer updates
TC15	Clear screen check	Observe console during turns	Screen clears between moves	Clean transitions
TC 16	Exit after game end	Finish hard level	“CONGRATULATIONS” Message dispalys	Program exit

SAMPLE OUTPUT SCREENSHOTS:

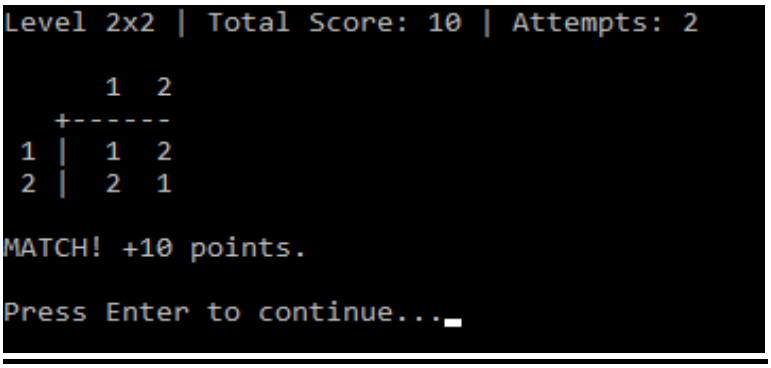
Title screen



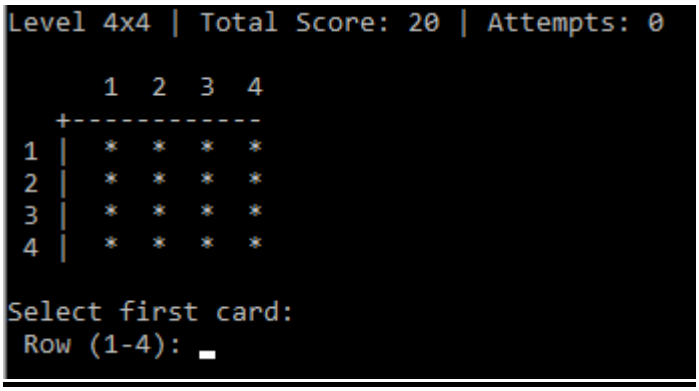
Easy (level):



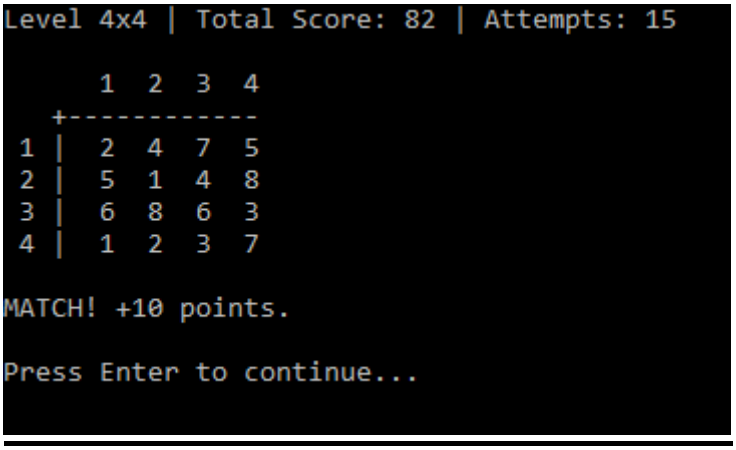
Easy level completed:



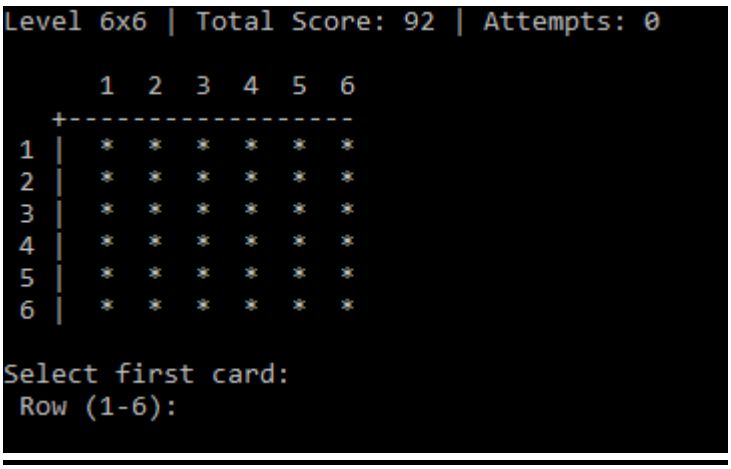
Medium (level):



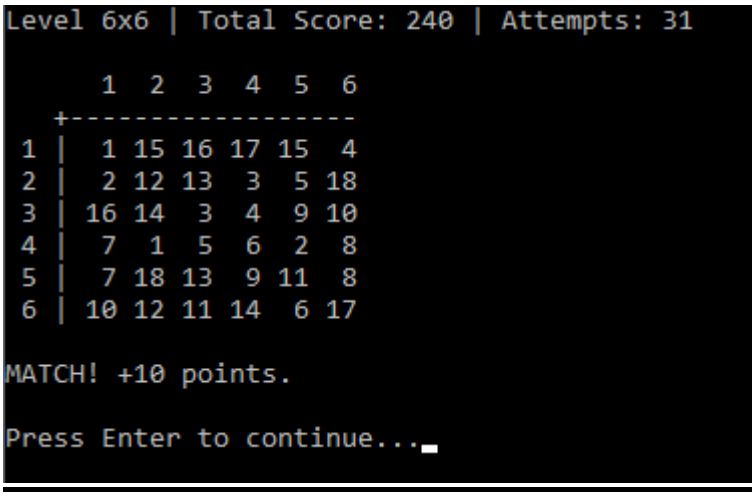
Medium level completed:

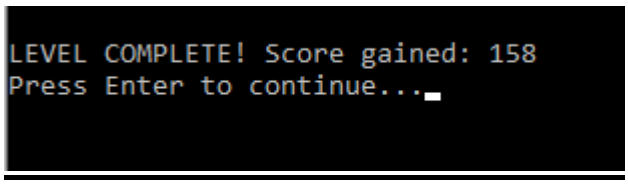


Hard (level):

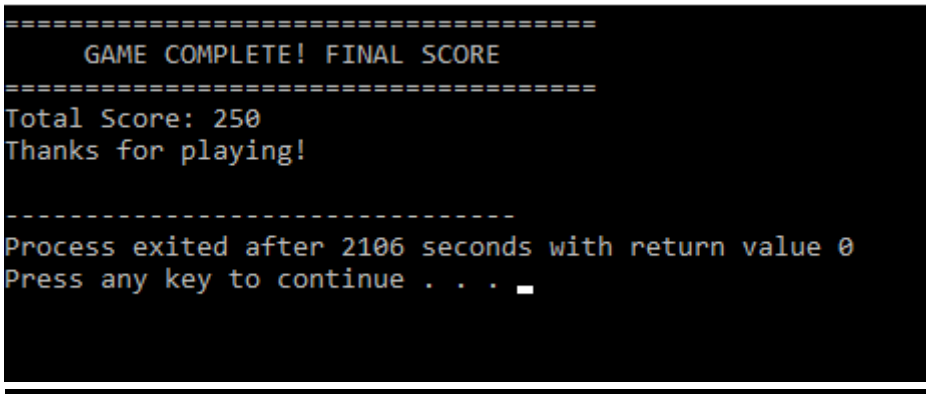


Hard level completed:


Level 6x6 | Total Score: 240 | Attempts: 31
1 2 3 4 5 6
1 | 1 15 16 17 15 4
2 | 2 12 13 3 5 18
3 | 16 14 3 4 9 10
4 | 7 1 5 6 2 8
5 | 7 18 13 9 11 8
6 | 10 12 11 14 6 17
MATCH! +10 points.
Press Enter to continue...


LEVEL COMPLETE! Score gained: 158
Press Enter to continue...

End of game:


GAME COMPLETE! FINAL SCORE
Total Score: 250
Thanks for playing!
Process exited after 2106 seconds with return value 0
Press any key to continue . . .

7. RESULTS AND DISCUSSION:

---

9. LIMITATIONS AND FUTURE SCOPE:

---

10. CONCLUSION: