Hands-on Activity 4.3:	
Sorting and Searching Arrays	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:** 9/17/2025
Section: CPE11S1	**Date Submitted:** 9/18/2025
Name(s): Ramirez, Angel Mae C.	**Instructor:** Engr. Jimlord M. Quejado

6. Output

7. Supplementary Activity

1. Write a program that asks for a number from the user and prints which day of the week that number corresponds to. The days are indexed from 0 (Sunday) to 6 (Saturday). Before the program gets a value from the array, it must first check if the given day is greater than or equal to zero and less than 7. If not, it should print the message: "Error, no such day." Your version of the program must print the same result as the expected output.

CODE:

Untitled1.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    string days[7] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
    int num;

    cout << "Enter a number (0-6): ";
    cin >> num;

    if (num >= 0 && num < 7) {
        cout << days[num] << endl;
    } else {
        cout << "Error, no such day." << endl;
    }

    return 0;
}
```

OUTPUT:

```
Enter a number (0-6): 7
Error, no such day.

--------------------------------
Process exited after 3.155 seconds with return value 0
Press any key to continue . . .
```

```
Enter a number (0-6): 1
Monday

--------------------------------
Process exited after 2.822 seconds with return value 0
Press any key to continue . . .
```

```
Enter a number (0-6): 2
Tuesday

--------------------------------
Process exited after 2.532 seconds with return value 0
Press any key to continue . . .
```

```
Enter a number (0-6): 3
Wednesday

--------------------------------
Process exited after 1.912 seconds with return value 0
Press any key to continue . . .
```

```
Enter a number (0-6): 4
Thursday

--------------------------------
Process exited after 2.766 seconds with return value 0
Press any key to continue . . .
```

```
Enter a number (0-6): 5
Friday

--------------------------------
Process exited after 4.03 seconds with return value 0
Press any key to continue . . .
```

```
Enter a number (0-6): 6
Saturday

--------------------------------
Process exited after 4.073 seconds with return value 0
Press any key to continue . . .
```

SHORT ANALYSIS:

In this program, I first created a string array called days that stores the names of the week in order, i started from index 0 as "Sunday" and ending with index 6 as "Saturday." After that i declared an integer variable num to store the number that the user will enter. After it, I ask the user to input a number between 0 and 6, and the program saves that input into num. Next thing i do is I used an if-else statement to check if the number is within the valid range. If the number is greater than or equal to 0 and less than 7, the program prints the corresponding day from the array using days[num]. If the number is outside this range, the program prints the message "Error, no such day." This makes sure the user only gets correct results and avoids invalid array access. Overall even though it looks like a simple easy code, i still struggled to make it because of syntax errors, but im definitely more familiar with it now through this.

2. Write a program that creates a chessboard, sets all the pieces on it and then displays the contents of the board. Create a two-dimensional array, fill it with data and print a letter when a piece is on the field and a space when there is no piece. Store one letter for one piece. For now, we don't need any information about the color of the pieces. The starting positions (with letters which symbolize each piece) for all pieces are: The rooks (R) are placed on the outside corners, right and left edge (white on the 1st and black on the 8th line). The knights (N) are placed immediately inside of the rooks. The bishops (B) are placed immediately inside of the knights. The queen (Q) is placed on the central square of the same color as that of the player: white queen on the white square and black queen on the black square. Both stand on the d rank: white queen on the d1 field and black queen on the d8 field. The king (K) takes the vacant spot next to the queen. The pawns (P - not the official symbol, but you need to print something) are placed one square in front of all of the other pieces. Your version of the program must print the same result as the expected output.

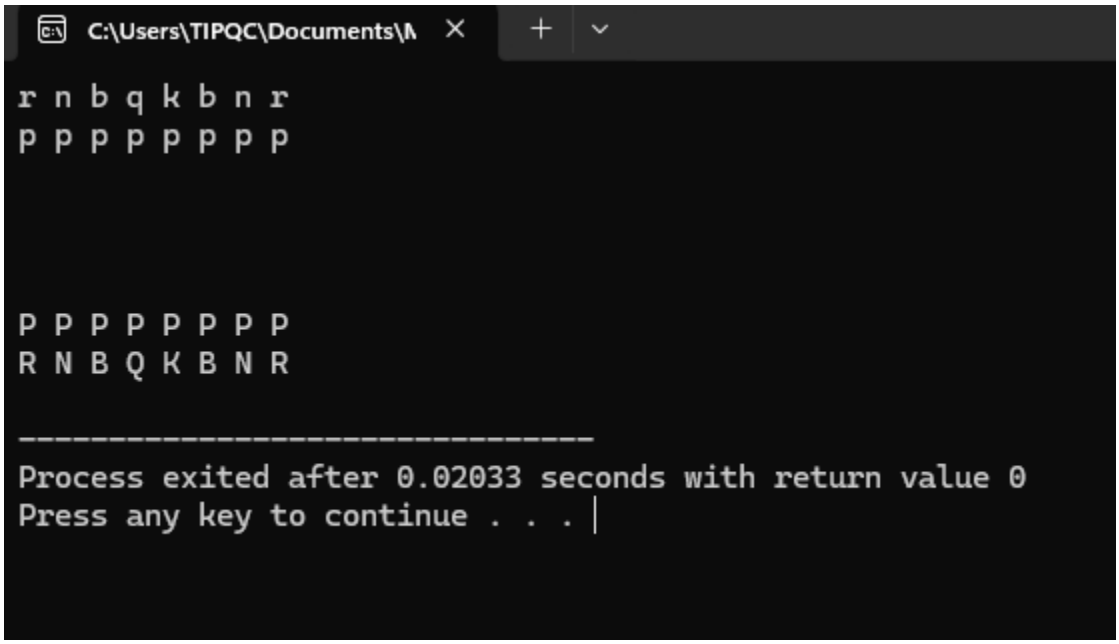
CODE:

Untitled1.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    const int SIZE = 8;
    char board[SIZE][SIZE];


    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            board[i][j] = ' ';
        }
    }

    board[0][0] = board[0][7] = 'r';
    board[0][1] = board[0][6] = 'n';
    board[0][2] = board[0][5] = 'b';
    board[0][3] = 'q';
    board[0][4] = 'k';
    for (int j = 0; j < SIZE; j++) {
        board[1][j] = 'p';
    }

    board[7][0] = board[7][7] = 'R';
```

```
25      board[7][1] = board[7][6] = 'N';
26      board[7][2] = board[7][5] = 'B';
27      board[7][3] = 'Q';
28      board[7][4] = 'K';
29      for (int j = 0; j < SIZE; j++) {
30          board[6][j] = 'P';
31      }
32
33      for (int i = 0; i < SIZE; i++) {
34          for (int j = 0; j < SIZE; j++) {
35              cout << board[i][j] << ' ';
36          }
37          cout << endl;
38      }
39
40      return 0;
41  }
42
```

OUTPUT:



SHORT ANALYSIS:

As a chess player i was enthusiastic to do this task, so after many errors this is how i manage to accomplish it. The first thing I did is, I started by making a constant and put a name size and set it to 8 because a chessboard has 8 rows and 8 columns. I created a 2d array and name it board to represent the chessboard and i used nested loops to fill it with spaces, showing that all squares are empty at first. After that, I placed the black pieces on the top two rows using lowercase letters like r for rook and p for pawn. I followed the correct order of the chess pieces by putting rooks on the corners, then knights, bishops, queen, and king in the middle. Then, I placed the white pieces on the bottom two rows, using capital letters to tell them apart from the black ones. I also used another loop to fill the second-to-last row with white pawns. Finally, I printed out the board using nested loops so the pieces would appear in the correct position, just like in a real chess game.

8. Conclusion

By doing these two programs, I understood a lot about arrays and loops in C++ and how essential it is to learn the logic step by step. One of the key takeaways from these was how to use arrays to store and retrieve data efficiently, such as a string array for days of the week or a 2D array to represent the chessboard. I also understood how important it is to incorporate condition checking, such as in the program for days of the week, when I employed an if-else statement to prevent invalid entry. Another aspect I learned was the need to stick to the proper structure of the program, from variable declaration, initialization, and then employing loops to process and print the data. When I coded the chessboard, I had to be extremely careful about the order of the pieces and the way the board is printed, thereby enhancing my knowledge of nested loops and array indexing.

Both programs needed meticulous planning and debugging procedure. Initially, I had syntax errors, semicolons missing, or wrong loop conditions, but through continued testing and correcting those errors, I gained more confidence in the C++ syntax. I also had to study the logic of each step, particularly in the chess program, where putting the right pieces in the right place took a few attempts. I discovered that dividing the task into smaller segments makes it more manageable, and checking the output on a regular basis catches errors early on. Despite it being frustrating at times, particularly when the program didn't function as I intended, I understood that it is okay to make mistakes and that mistakes are what make me a better programmer.

This exercise not only helped me feel more at ease with coding C++, but it also allowed me to feel more comfortable attempting more difficult things in the future. I'm satisfied with what I was able to do, and I'm excited to learn more and build my skills gradually.