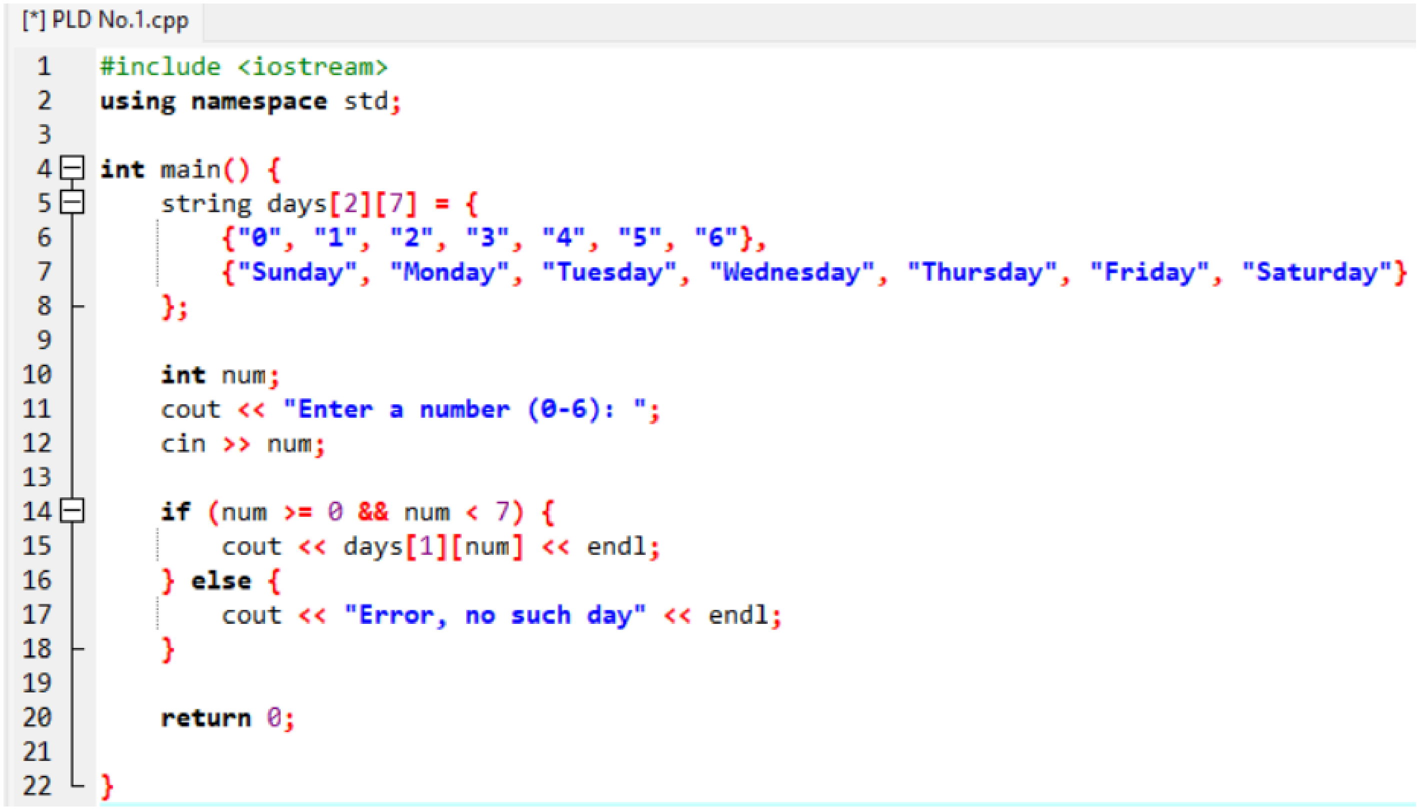
Activity No. 4.3	
Sorting and Searching Arrays	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:** September 16, 2025
Section: CPE11S1	**Date Submitted:** September 17, 2025
Name(s): Mendoza, Nathaniel B.	**Instructor: Engr. Jimlord M. Quejado**
6. Output	
7. Supplementary Activity	

Code:

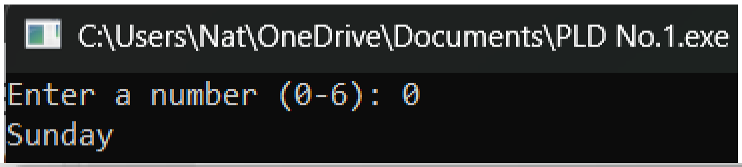


[*] PLD No.1.cpp

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

int main() {
    string days[2][7] = {
        {"0", "1", "2", "3", "4", "5", "6"},
        {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"}
    };

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

    if (num >= 0 && num < 7) {
        cout << days[1][num] << endl;
    } else {
        cout << "Error, no such day" << endl;
    }

    return 0;
}
```

Code Outputs:

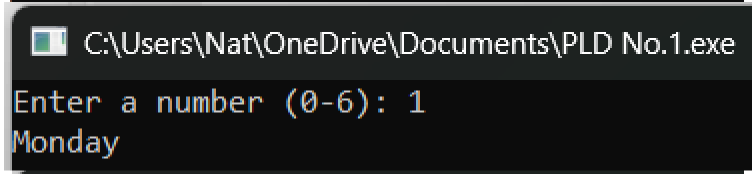
#1 Sunday



C:\Users\Nat\OneDrive\Documents\PLD No.1.exe

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

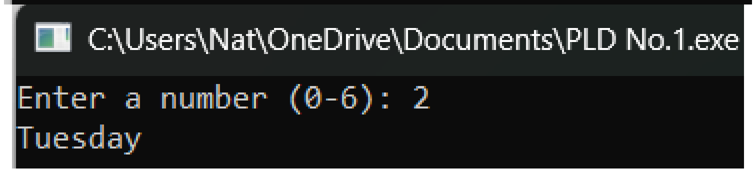
#2 Monday



C:\Users\Nat\OneDrive\Documents\PLD No.1.exe

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

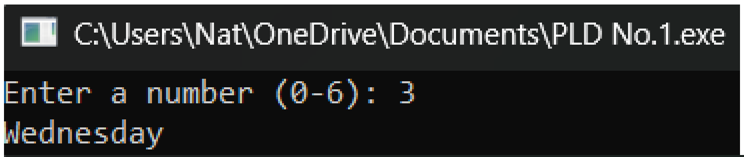
#3 Tuesday



C:\Users\Nat\OneDrive\Documents\PLD No.1.exe

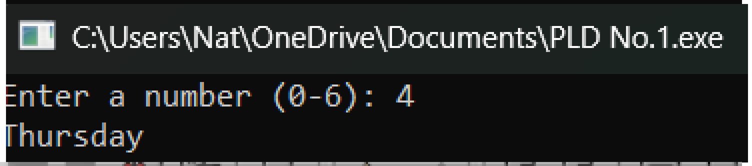
```
Enter a number (0-6): 2
Tuesday
```

#4 Wednesday



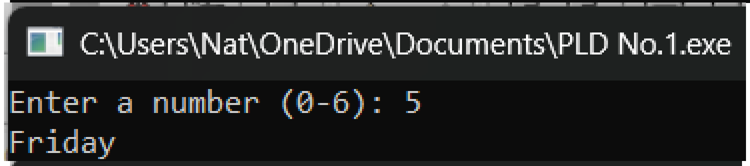
```
Enter a number (0-6): 3
Wednesday
```

#5 Thursday



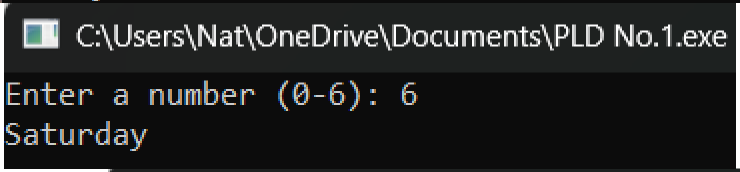
```
Enter a number (0-6): 4
Thursday
```

#6 Friday



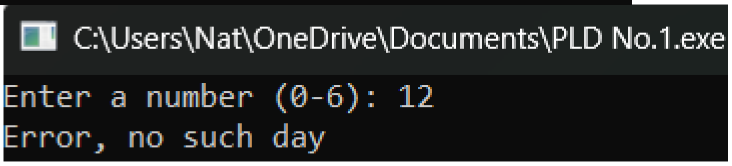
```
Enter a number (0-6): 5
Friday
```

#7 Saturday



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

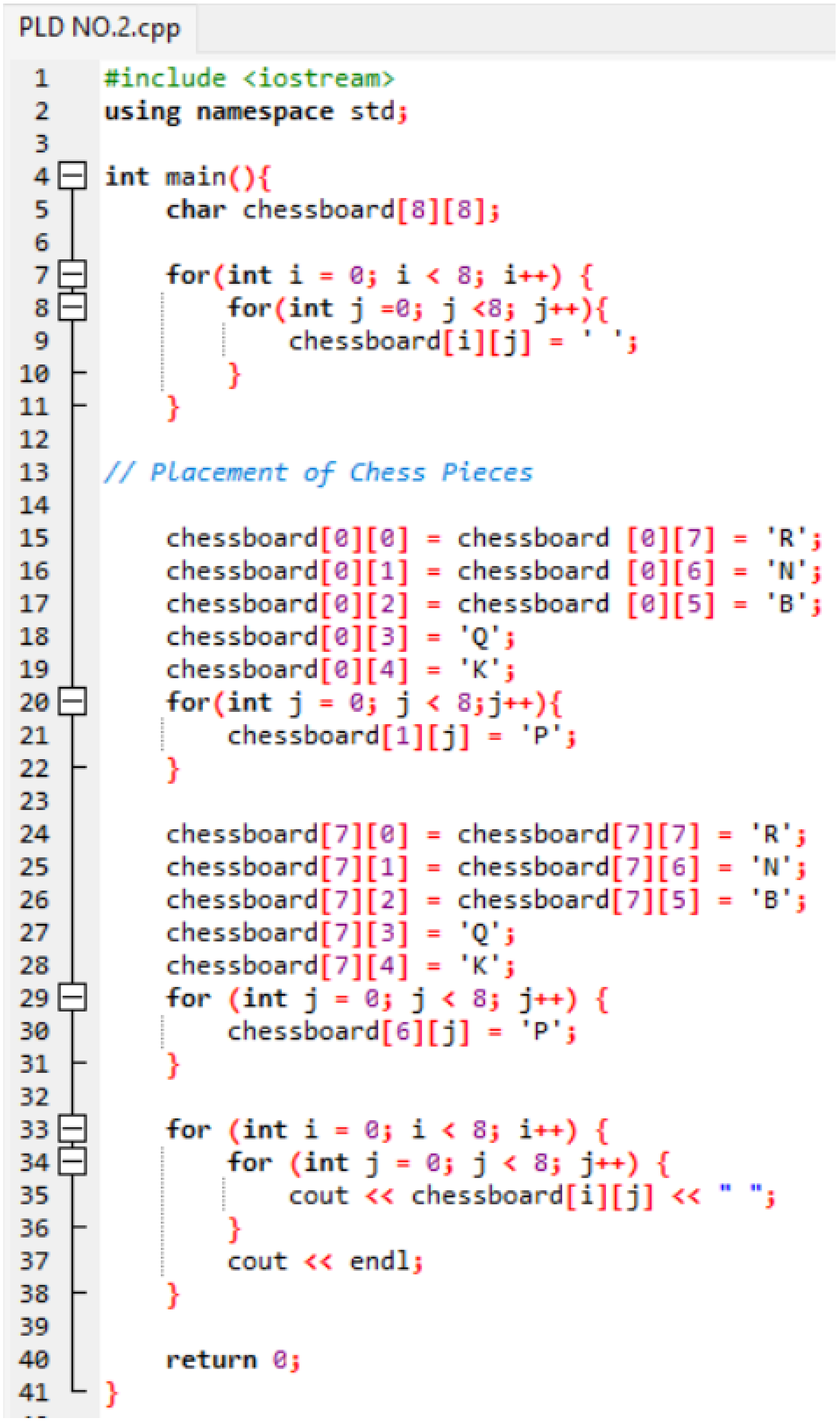
# 8 Input > 6 (Error, no such day)



```
Enter a number (0-6): 12
Error, no such day
```

Code Analysis:
I started the code using #include <iostream> known as Input/Output stream which allows me to use cout and cin for displaying and receiving input for my code. Next is that I implemented using namespace std; so that I do not need to repeatedly type std for my cout and cin everytime I use it. Then for the next line, int main() is the starting point of my program where everything inside {} will run when I execute my code. After that I programmed a 2 dimensional array which is the string days[2][7] that contains the numbers “from 0” and the name of the days from “Sunday to Saturday”. Next is the variable int num; so that I will use it to be stored within my input, so that by using cin>>num; it will allow the user or me as the coder to type a number I want. So in order for my program to check whether the input is invalid, I used logical operators to be specific “&&” called logical and in which by definition it returns true if both statements are true to implement a condition if (num >= 0 && num <= 6). This condition means that if the user enters a number from 0 to 6, the program will print the corresponding day using cout << days[num];. However, if the condition is not satisfied (for example the input is less than 0 or greater than 6), the program will go to the else part and print "Error: no such day". Therefore, this program only responds from 0-6 in response to the day of the week while it handles invalid input in which it received an input that is less than 0 or greater than 6 where it prints Error, no such day.

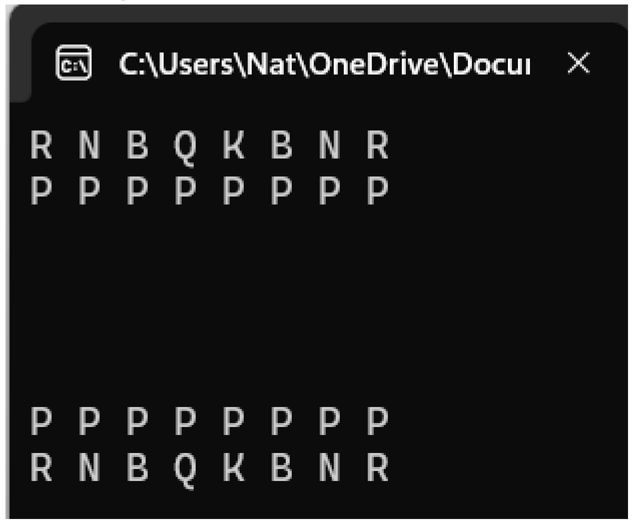
Code:



PLD NO.2.cpp

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

int main(){
    char chessboard[8][8];

    for(int i = 0; i < 8; i++) {
        for(int j =0; j <8; j++){
            chessboard[i][j] = ' ';
        }
    }

// Placement of Chess Pieces

    chessboard[0][0] = chessboard [0][7] = 'R';
    chessboard[0][1] = chessboard [0][6] = 'N';
    chessboard[0][2] = chessboard [0][5] = 'B';
    chessboard[0][3] = 'Q';
    chessboard[0][4] = 'K';
    for(int j = 0; j < 8;j++){
        chessboard[1][j] = 'P';
    }

    chessboard[7][0] = chessboard[7][7] = 'R';
    chessboard[7][1] = chessboard[7][6] = 'N';
    chessboard[7][2] = chessboard[7][5] = 'B';
    chessboard[7][3] = 'Q';
    chessboard[7][4] = 'K';
    for (int j = 0; j < 8; j++) {
        chessboard[6][j] = 'P';
    }

    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            cout << chessboard[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Code Output:



```
C:\Users\Nat\OneDrive\Docu

R N B Q K B N R
P P P P P P P P




P P P P P P P P
R N B Q K B N R
```

Code Analysis:
I started the code using #include <iostream> which is known as the Input/Output stream, allowing me to use cout for displaying outputs and cin if I ever want to receive inputs. Next, I implemented using namespace std; so that I don't need to repeatedly type *std::cout* or *std::cin* every time I want to use them. Then, I wrote int main() which is the entry point of the program. Everything inside the curly braces {} will be executed once the program starts running. Inside the main function, I first created a two-dimensional array called char chessboard[8][8]; which represents the chessboard with 8 rows and 8 columns so meaning its 8*8 a total of 64 slots or spaces which corresponds to the number of slots a chessboard has. That element in this array (chessboard) stores a single character that symbolizes either a chess piece (R as Rook, N as Knight, B as Bishop, Q as Queen, K as King, and P as Pawn) or a blank space when no piece is present. After that, I used a nested for loop to initialize the entire board with spaces ' '. In which this will ensure that before placing any pieces, all squares on the chessboard are empty. Next, I placed the black pieces at the top of the board (rows 0 and 1). The first row is arranged as: Rooks (R) on the corners, Knights (N) beside them, Bishops (B) next, followed by the Queen (Q) on column 3 and the King (K) on column 4. Then, in the second row, I filled all 8 columns with Pawns (P). After finishing the black side, I did the same for the white pieces at the bottom of the board (rows 7 and 6). The arrangement is identical: Rooks, Knights, Bishops, Queen, King, and then a full row of Pawns above them. So basically to summarize that way, I just filled which corresponding slots a specific piece should be placed upon as you base on my code. Finally, I used another nested for loop to print the contents of the board. Each element of the array is displayed with a space after it, making the board look like an actual chessboard layout. After each row is finished, cout << endl; is used to move to the next line. Therefore, this program has successfully provided the chessboard initial setup / layout.

8. Conclusion

Today, I learned how to analyze C++ programs step by step by identifying its structure, purpose, and logic. As we are required to analyze it from start to finish, I find it beneficial to me as a Computer Engineering student because I found it challenging in a way that not just by coding I am also learning how each of the functions works in the way that I am breaking down each code and analyzing it one by one. In the first program (days of the week),I encountered many errors than my performance on the 2nd problem as I am still trying to understand how arrays, user input, and conditional statements work together to validate input and display the correct output in the end I find it more challenging and a less enjoyment as I found encountering many errors as you try to fix it one by one and still receiving an error can be stressful. In the second program (chessboard setup), I applied the concept of two-dimensional arrays and nested loops to represent and print a more complex structure, the chessboard, with its initial piece arrangement, and to conclude my way of implementing this, I just arranged and initialized or code a specific piece to their corresponding areas so that it will be more easy to me as I read the problem I saw that it only requires me to only print it with the use of two-dimensional arrays and using a loop to shorten my code not just from printing. From this, I can conclude that I am becoming more comfortable not only in writing programs but also in explaining how each part functions and contributes to the overall result.