Hands On 4.3

Sorting and Searching Arrays

Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: 09/16/25**
Section: CPE11S1	**Date Submitted: 09/18/25**
Name(s): James Daniel M. Verano	**Instructor: Engr. Jimlord M. Quejado**

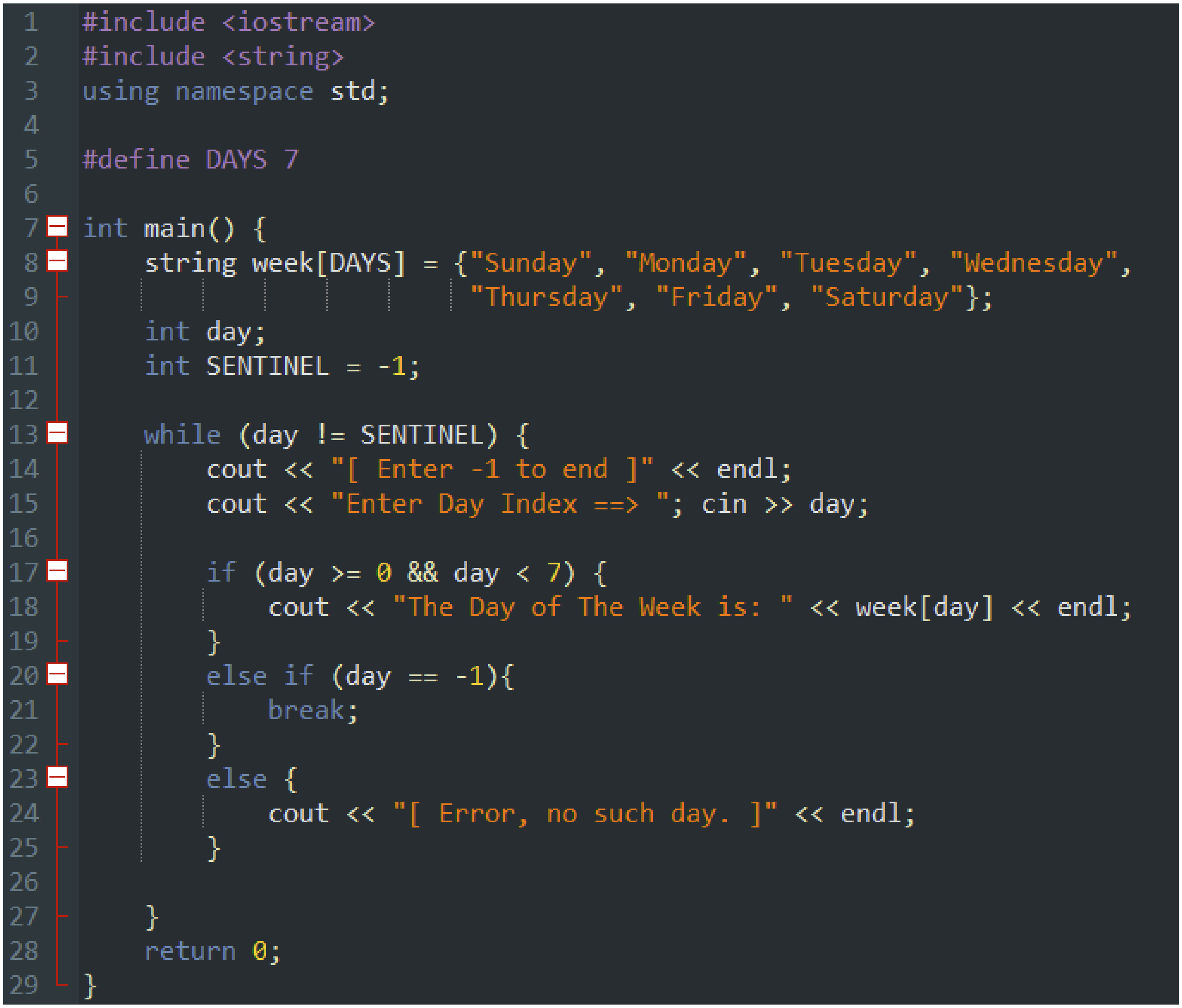
6. Output

- Screenshot of Code(Readable):
- Output of Code(label and compile ALL possible outputs):
- Short Analysis(min 5 sentences)

7. Supplementary Activity

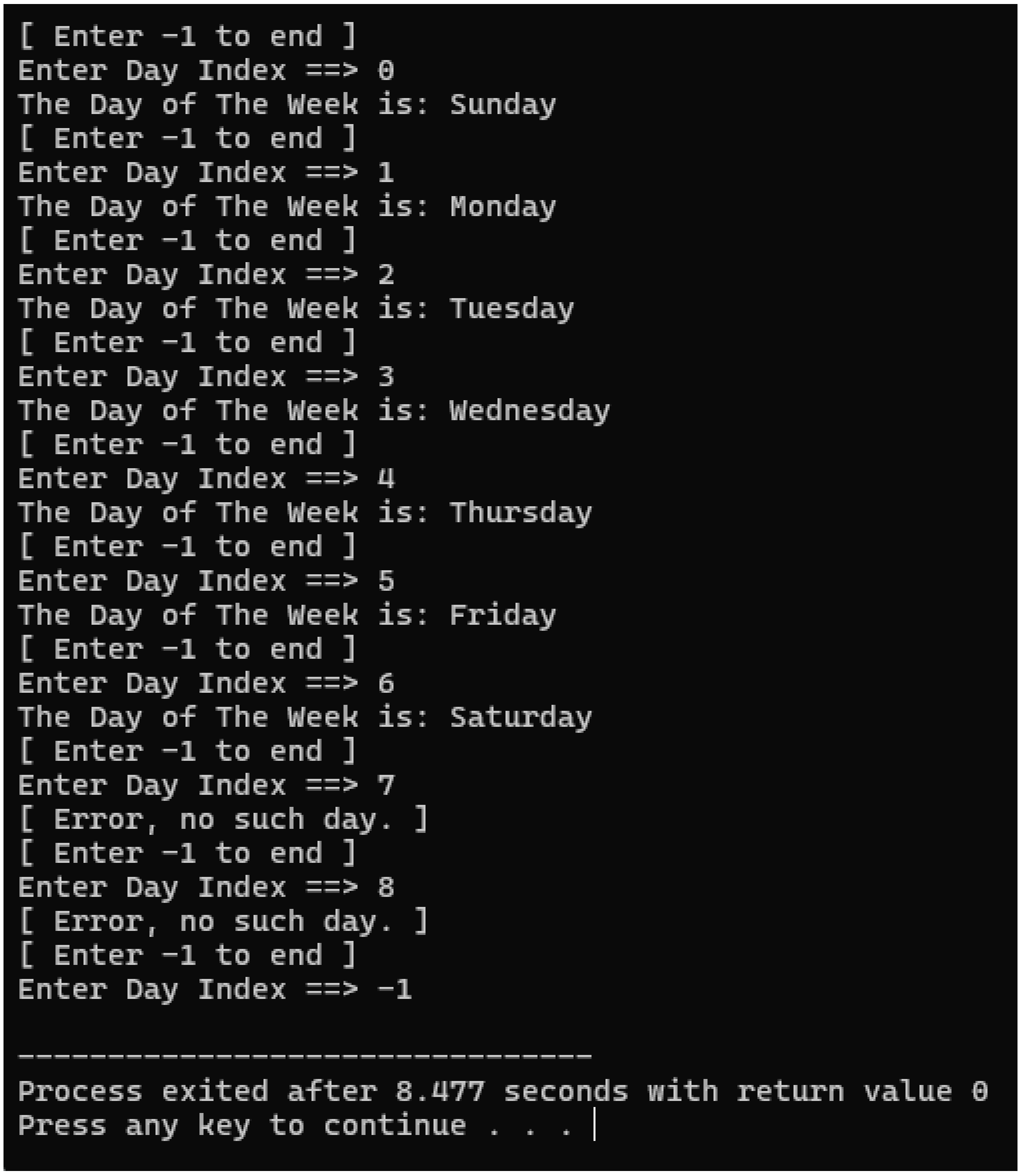
Example 1:

Screenshot of the Code:



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

#define DAYS 7

int main() {
    string week[DAYS] = {"Sunday", "Monday", "Tuesday", "Wednesday",
                         "Thursday", "Friday", "Saturday"};
    int day;
    int SENTINEL = -1;

    while (day != SENTINEL) {
        cout << "[ Enter -1 to end ]" << endl;
        cout << "Enter Day Index ==> "; cin >> day;

        if (day >= 0 && day < 7) {
            cout << "The Day of The Week is: " << week[day] << endl;
        }
        else if (day == -1){
            break;
        }
        else {
            cout << "[ Error, no such day. ]" << endl;
        }

    }
    return 0;
}
```

Outputs of the Code:



```
[ Enter -1 to end ]
Enter Day Index ==> 0
The Day of The Week is: Sunday
[ Enter -1 to end ]
Enter Day Index ==> 1
The Day of The Week is: Monday
[ Enter -1 to end ]
Enter Day Index ==> 2
The Day of The Week is: Tuesday
[ Enter -1 to end ]
Enter Day Index ==> 3
The Day of The Week is: Wednesday
[ Enter -1 to end ]
Enter Day Index ==> 4
The Day of The Week is: Thursday
[ Enter -1 to end ]
Enter Day Index ==> 5
The Day of The Week is: Friday
[ Enter -1 to end ]
Enter Day Index ==> 6
The Day of The Week is: Saturday
[ Enter -1 to end ]
Enter Day Index ==> 7
[ Error, no such day. ]
[ Enter -1 to end ]
Enter Day Index ==> 8
[ Error, no such day. ]
[ Enter -1 to end ]
Enter Day Index ==> -1

--------------------------------
Process exited after 8.477 seconds with return value 0
Press any key to continue . . .
```

From an input index 0 to 8 and SENTINEL input to end the program.

Analysis:

In this program, I used a while and if-else condition statements. The if-else condition statement is to read and identify what is the inputted value by the recipient, and this is also where they will identify WHICH type of index of array element will they print as it will read their desired input.

In the very start of the program, we declared the variable *string week[DAYS]* to have a directory on where we will be linking our index value and the inputs we desire to output.
for visuals:
[i] = 0 = Sunday
[i] = 1 = Monday
[i] = 2 = Tuesday
. . . . .

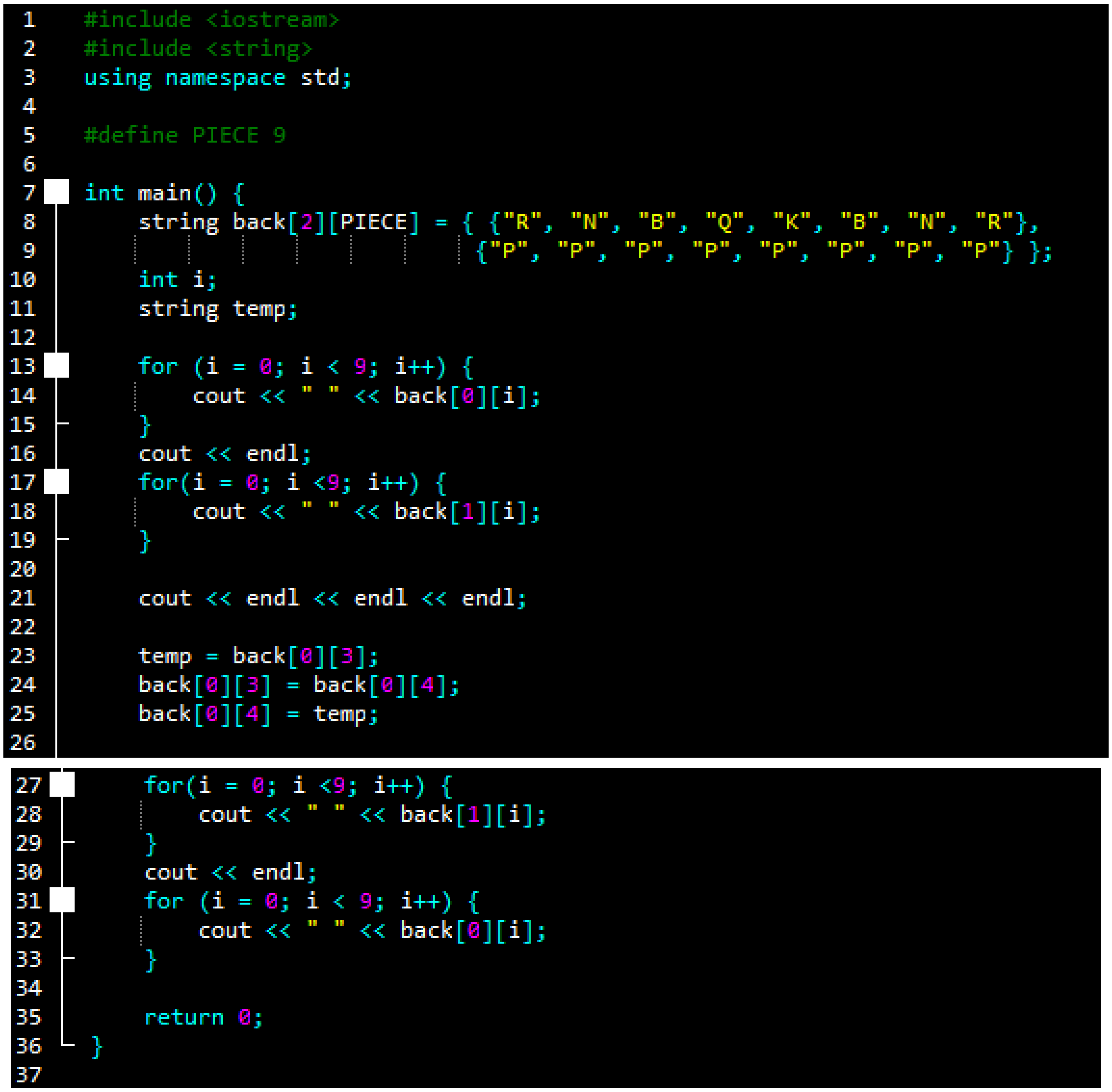
BUT, if the value is non-existent, it will output an error prompt of “Error, no such day.” as it is programmed in the else statement. Our limit is only 6, since it is only 7 days in a week, counting 0 - 6, is already enough for the index directory for the whole week.

The while loop is for, WHEN the input is not equal to the SENTINEL value, which is defined as -1, the program will continue to run. Inside the while function, there are if-else statements where each statement has conditions that need to be met, and that is where we read and identify our inputted value, and where the function will be compared to see which condition has been met to output their desired conditional function. Let’s say, the recipient, inputted 1, It’s not -1 so the program will continue to run, it’s in the range of the first if-else statement, where the inputted value is >= 7, thus it will run the conditioned code, which is printing the variable *week[day]*. The variable *week[day]* is the same as, *week[i]*. We know that the day is the inputted value of the recipient, so it is declared as, *week[1]* where 1 can be interpreted as an index value.

Now, IF the recipient inputted a value that isn’t in the range of the array elements, it will proceed in the ELSE statement where it will print out the error prompt.

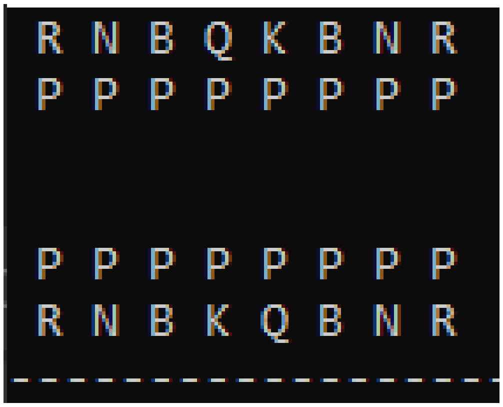
Example 2:

Screenshot of the Code:



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

#define PIECE 9

int main() {
    string back[2][PIECE] = { {"R", "N", "B", "Q", "K", "B", "N", "R"},
                             {"P", "P", "P", "P", "P", "P", "P", "P"} };
    int i;
    string temp;

    for (i = 0; i < 9; i++) {
        cout << " " << back[0][i];
    }
    cout << endl;
    for(i = 0; i <9; i++) {
        cout << " " << back[1][i];
    }

    cout << endl << endl << endl;

    temp = back[0][3];
    back[0][3] = back[0][4];
    back[0][4] = temp;

    for(i = 0; i <9; i++) {
        cout << " " << back[1][i];
    }
    cout << endl;
    for (i = 0; i < 9; i++) {
        cout << " " << back[0][i];
    }

    return 0;
}

```

Output of the Code:



```
 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 K Q B N R
-----------------
```

Analysis:

In this program, we have created a chess board. In order to do that, I created a multi array that consists of both chess piece rows, the outer pieces, which are the pawns, and the inner pieces, which are the rook, bishop, etc.

In the beginning, we declared our variables, with *string back[2][PIECE]*. In that string array, we have 2 element arrays that are counted up to 8 since it is an 8 by 8 chess board. We will be using it in the process of printing the pieces in their order.

Secondly, we have a declaration for "s*tring temp*", as we know, in the common chess piece position of the both players, the Queen and the King switches places.

Now, in the usage of the first for loop function, I used the common looping function wherein if *[i]* is less than 9. Alternatively, we can use the term *PIECE* in that term as it is also defined equal to 9. Now, in the first loop function, is the printing of the inner chess pieces, which are the rook, bishop, knight, etc. Now, the question is, how can we get the inner piece of the specific set of array element values? As we know, we declared *back[2]* to identify that it declared 2 sets of arrays. We can define it as, *back[0]* and *back[1]*. Just the same with the index, 0 means the first element value = the first set of ARRAY elements, and 1 means the second set of array element value. I first declared the *back[0]* inside the for loop function to print out the inner piece, and in the second for loop function, I used the *back[1]* to now proceed on printing the outer pieces, which are known as the pawns.

In the middle, we can see that we have a switching process, where I switched the array value element 3 and 4, which is the KING and the QUEEN. This is to accurately measure the correct position of each piece, for both players, thus I used the switching of array values, so that we don't need to declare another set of array for the second player.

And I just copy and pasted the code, but the first for loop in the second part of the chess board, is the pawn, and the second for loop is for the printing of inner chess pieces.

8. Conclusion

With the use of the for loop function with the multi array declaration, It created a whole new different area where we no longer need to declare multiple types of array values if we need to print out or use a specific number of array as an element. With the 2 programs that I have created, I have tried to create a thorough procedure and a thorough explanation of the purpose of each line code, and the process of their function, such as the if-else statement, switch case, for loop function, and while function. The supplementary activity has been the basis for implementing the different processes of each function where each is dependent on one another in order for the program to work. I think, for me, I did quite well, yet, lacking. Given that there could have been an easier method in creating the program to make it more concise and clear, thus, my areas for improvement is to further gain a deeper knowledge about the topics related to arrays, sorting, searching and switching of arrays, and many more conditional function statements.