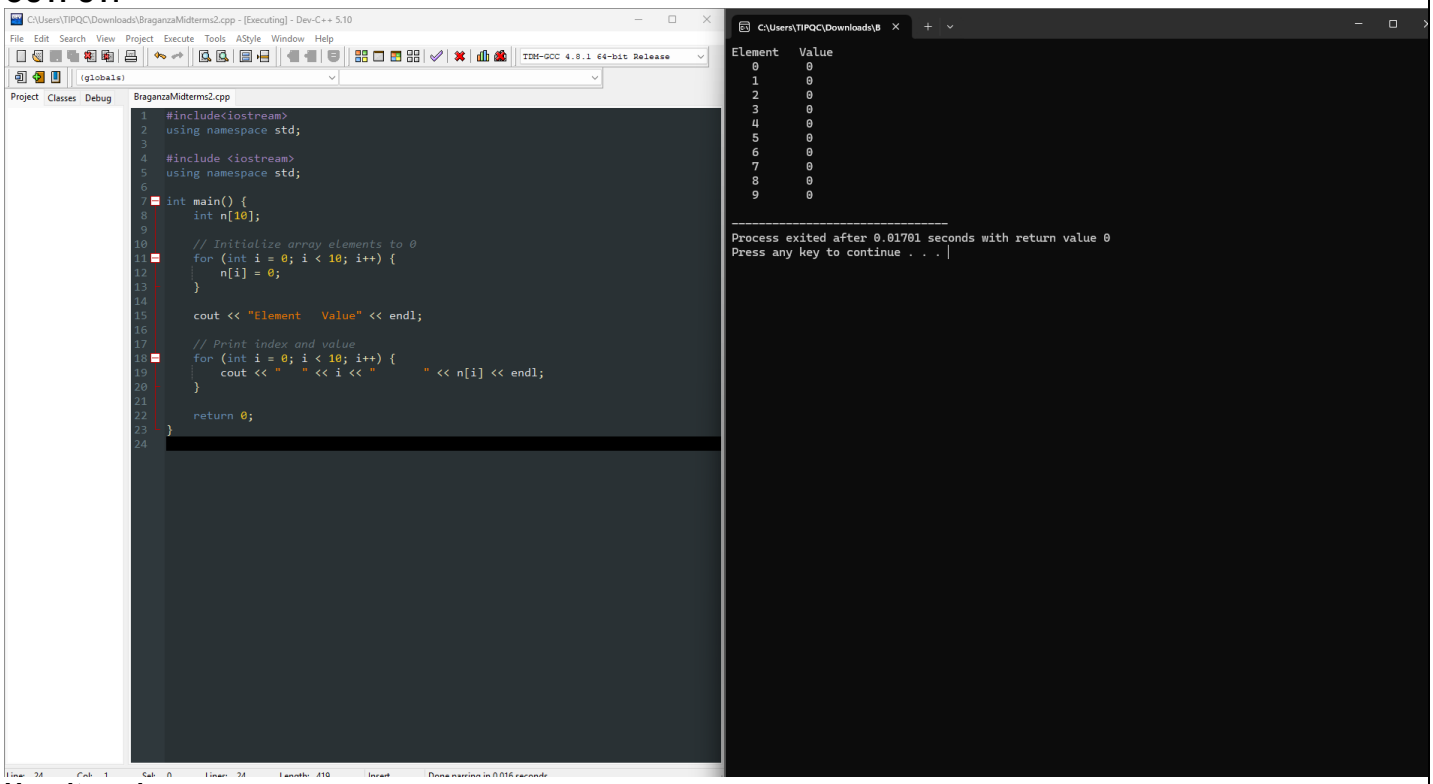
Hands-on Activity 4.2: Arrays	
Arrays	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: 13/09/2025**
Section: CPE11S1	**Date Submitted: 13/09/2025**
Name(s): Ralph Angelov F. Braganza	**Instructor: Engr. Jimlord M. Quejado**
Output	

EXAMPLE 1: initializing an array.

```
#include<iostream>
using namespace std;
#include <iostream>
using namespace std;
int main() {
    int n[10];
    // Initialize array elements to 0
    for (int i = 0; i < 10; i++) {
        n[i] = 0;
    }
    cout << "Element   Value" << endl;
    // Print index and value
    for (int i = 0; i < 10; i++) {
        cout << "   " << i << "       " << n[i] << endl;
    }
    return 0;
}
```

OUTPUT:



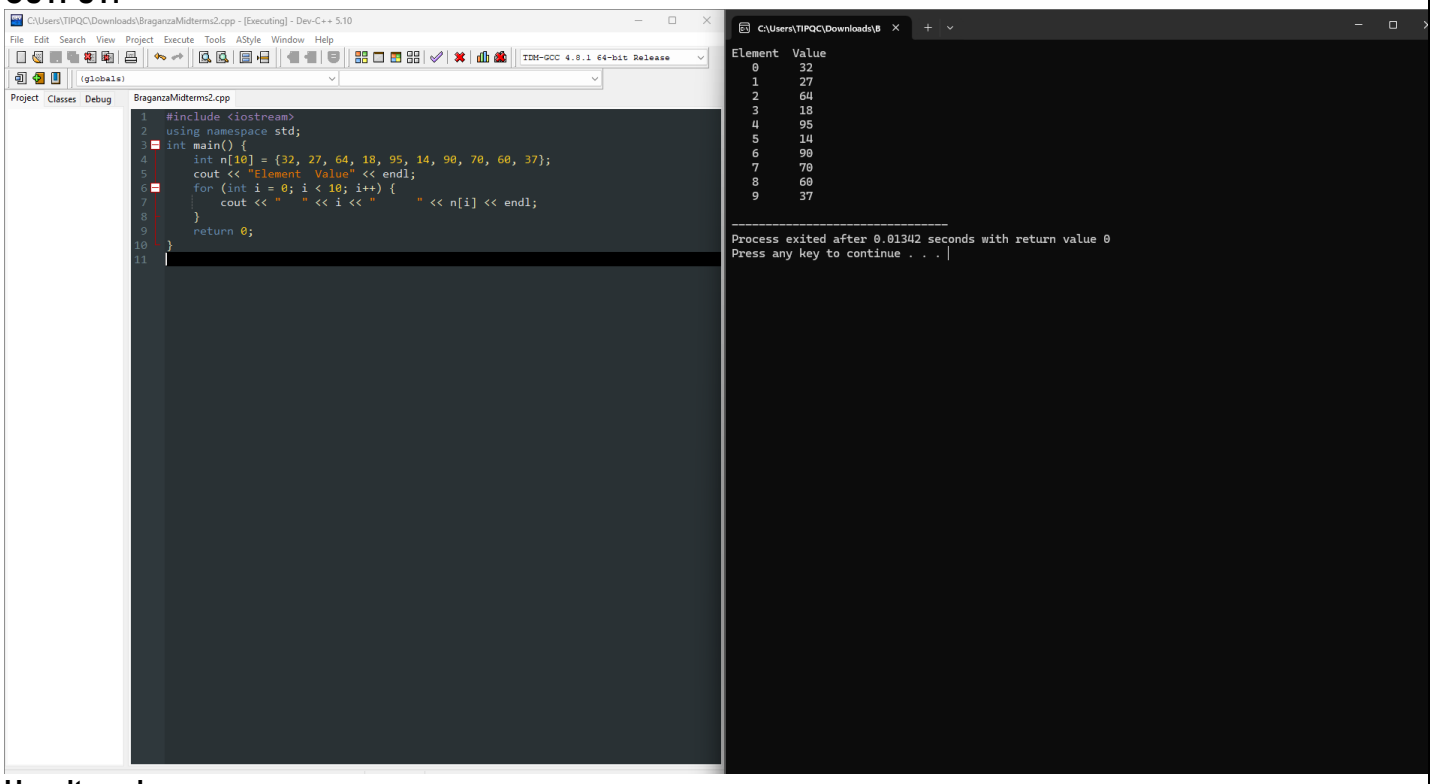
How it works:

We have the initialized array which is the int n[10] with “10” being the 10 elements for the array and “int” declaring that everything inside the array is an integer. The loop will run from i = 0 all the way to i = 9 so what this will do is in each iteration (or in other words repetition of a computational procedure) sets the current array element to 0 then i++ increments it by 1 so it will go to the next element of the array to make it 0. The reason why it goes up to only 9 is because the array always starts at 0 rather than 1 that is why it will keep on running until i < 10. The "Element Value" is just for labeling the values but after that we have the same identical loop except it doesn’t have the the n[i] = 0 that makes the value 0 so it just prints out the array index and i++ to print out the next index. The rest of the code is just spacing but “endl” ensures that each index-value pair is printed on a new line, which is why the result looks vertical.

EXAMPLE 2: initializing an array with a declaration.

```
#include <iostream>
using namespace std;
int main() {
    int n[10] = {32, 27, 64, 18, 95, 14, 90, 70, 60, 37};
    cout << "Element  Value" << endl;
    for (int i = 0; i < 10; i++) {
        cout << "   " << i << "      " << n[i] << endl;
    }
    return 0;
}
```

OUTPUT:



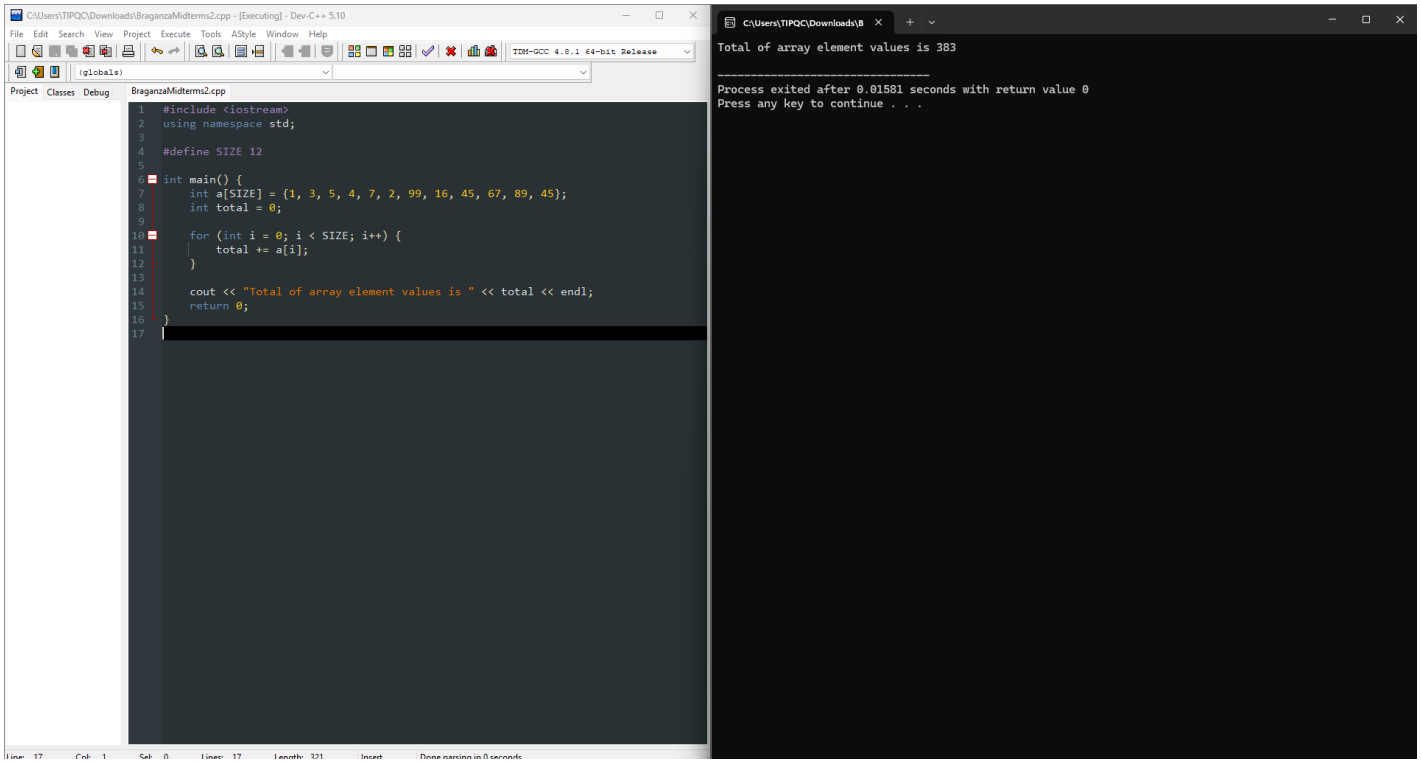
How it works:

This time we have an initialized array with declarations, this creates an array of 10 integers called n and each element of the array is preloaded with values (32, 27, 64, 18, 95, 14, 90, 70, 60, 37). Again, “Element Value” is just to print out the labeling so the output looks organized. We have the loop that starts with i = 0 and it runs until i < 10 and in each iteration it will print the current index (i) then it will print the stored value of that index (n[1]). Then the rest is spacing and “endl” ensures that every new pair is printed onto a new line as mentioned before.

EXAMPLE 3: computing sum of elements of the array.

```
#include <iostream>
using namespace std;
#define SIZE 12
int main() {
    int a[SIZE] = {1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45};
    int total = 0;
    for (int i = 0; i < SIZE; i++) {
        total += a[i];
    }
    cout << "Total of array element values is " << total << endl;
    return 0;
}
```

OUTPUT:



How it works:

With this code we have the normal required function and an optional one which is the “using namespace std” but in this code we have #define SIZE 12, this is a macro and essentially what this does is set the word “SIZE” (wherever the word is in the code) to the value of 12 so int a[SIZE] really means int a[12]. We have our array with “a” being the name and [SIZE] being our 12 elements with our preloaded values (1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45). int total = 0 will be used to keep track of the sum of all the values in the array elements. Then we have our loop that will start with i = 0 and runs until i < SIZE (i < 12), i++ will increment so it goes to the next element. On each iteration total += a[i] (total = total + a[i] I’m just rewriting this) will keep adding the value of the current array to the total until i < SIZE. After the loop is done it will print out the total value of the array.

Supplementary Activity

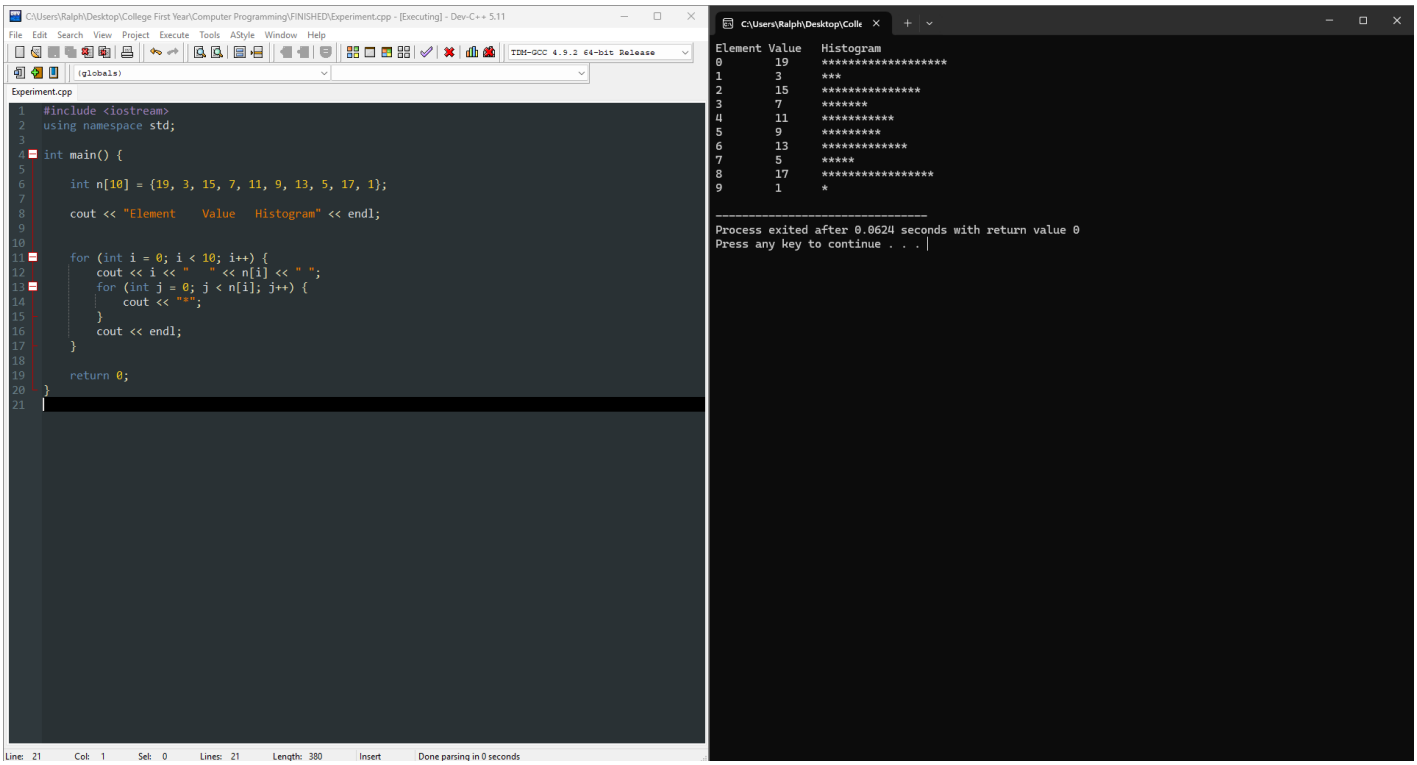
1. Given the size of an array which is 10, and the elements such as 19, 3, 15, 7, 11, 9, 13, 5, 17 and 1, create a program that will display the following output:

```
Element     Value     Histogram
    0          19      ******************
    1           3      ***
    2          15      ***************
    3           7      *******
    4          11      ***********
    5           9      *********
    6          13      *************
    7           5      *****
    8          17      *****************
    9           1      *
```

CODE:

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

int main() {

   int n[10] = {19, 3, 15, 7, 11, 9, 13, 5, 17, 1};

   cout << "Element       Value    Histogram" << endl;


   for (int i = 0; i < 10; i++) {
      cout << i << "           " << n[i] << "      ";
      for (int j = 0; j < n[i]; j++) {
         cout << "*";
      }
      cout << endl;
   }

   return 0;
}
```

RESULT:



```
Element Value   Histogram
0       19      ********************
1       3       ***
2       15      ***************
3       7       *******
4       11      ***********
5       9       *********
6       13      *************
7       5       *****
8       17      *****************
9       1       *

--------------------------------
Process exited after 0.0624 seconds with return value 0
Press any key to continue . . .
```

How it works:

We have our initialized and declared array which is int n[10] with 10 elements and our preloaded values (19, 3, 15, 7, 11, 9, 13, 5, 17, 1). "Element Value Histogram" is just for labeling, the outer loop with i = 0 and it will run until i < 10 and in each iteration, it will print the current index (i) then it will print the stored value of that index (n[1]). The inner loop will print be responsible for the histogram, how this works is that it initializes j = 0 and runs while j < n[i]. Since n[i] is the stored value of the current element, this determines how many "*" will be printed. Each iteration prints one star, and j++ increments the counter until it reaches n[i] (let's say n[i] is 20 so it will stop until it prints 20 "*"). At that point, the inner loop stops, and the outer loop moves on to the next array element.

2. Given the following data, create a program that summarizes the number of each type. Use array responses for the 40 element array of student's responses. Such as
int responses[RESPONSE_SIZE] = { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10}

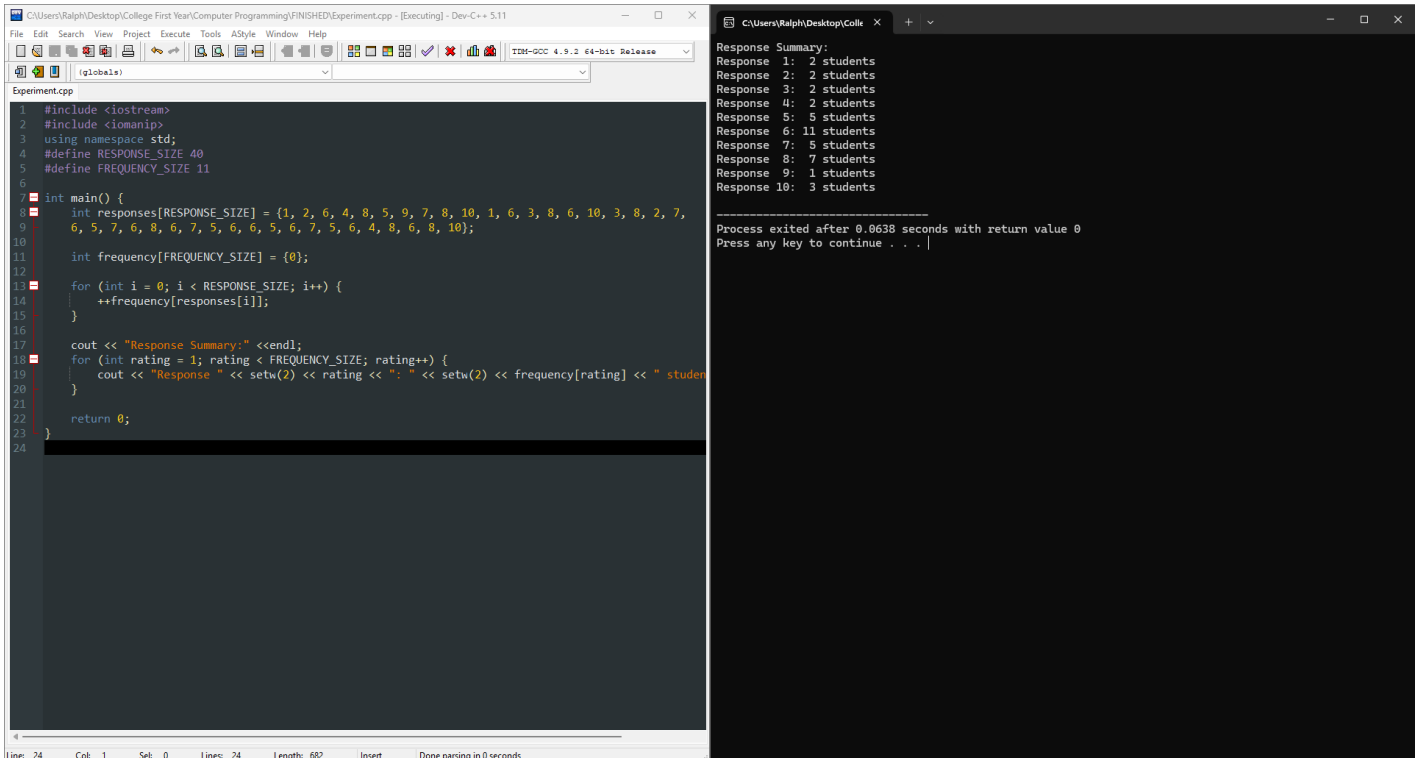
The output should be similar to the one below:

```
Response Summary:
Response  1:  2 students
Response  2:  2 students
Response  3:  2 students
Response  4:  2 students
Response  5:  5 students
Response  6: 12 students
Response  7:  4 students
Response  8:  7 students
Response  9:  1 students
Response 10:  3 students
```

CODE:

```
#include <iostream>
#include <iomanip>
using namespace std;
#define RESPONSE_SIZE 40
#define FREQUENCY_SIZE 11

int main() {
   int responses[RESPONSE_SIZE] = {1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 1, 6, 3, 8, 6, 10, 3, 8, 2, 7,
           6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 5, 6, 7, 5, 6, 4, 8, 6, 8, 10};

   int frequency[FREQUENCY_SIZE] = {0};

   for (int i = 0; i < RESPONSE_SIZE; i++) {
      ++frequency[responses[i]];
   }

   cout << "Response Summary:" <<endl;
   for (int rating = 1; rating < FREQUENCY_SIZE; rating++) {
      cout << "Response " << setw(2) << rating << ": " << setw(2) << frequency[rating] << " students" << endl;
   }

   return 0;
}
```

RESULT:



How it works:

In this code I added #include <iomanip> so I could use setw(), this would fix the spacing/alignment between Reponses and students. After learning about #define I've implemented that function into my code, I added #define RESPONSE_SIZE 40 (this will be the total number of student responses) and #define FREQUENCY_SIZE 11 since

responses range from 1 to 10, we need 11 slots and index 0 will be unused because nobody can answer 0 as their answer. We have the array of 40 student responses that represents the student's choice between 1 and 10, int frequency[FREQUENCY_SIZE] = {0} this will create an array to store counts of each responses (how answered 1 , 2 ,3, 4, and so on till 10) and it is initialized to 0 so all the elements start at 0. Now for outer loop, the loop goes through every element in the response array and since the RESPONSE_SIZE = 40 then the loop runs 40 times for each student response. The responses[i] is the actual student's answer at index I and frequency[responses[i]] uses the student's answer as the index into the frequency array (frequency[FREQUENCY_SIZE] = {0}). Example if we have response[i] = 6 then we can access frequency [6] and if response[i] = 6 happens again then ++frequency[responses[i]] it will increment by 1

Here is what I mean
First response is 1 then we have our frequency[1] it goes from 0 to 1.
Second response is 2 then we have our frequency[2] it goes from 0 to 1.
Third response is 6 then we have our frequency[6] it goes from 0 to 1.
And if we encounter a student who also answered 6 then frequency[6] goes 1 to 2

After that we print out "Response Summary" then endl so it goes to the next line of code. The inner loop, this is where we will printing out our ratings, so int rating starts at 1 and runs until < FREQUENCY_SIZE (which is 11 because we wont be using 0) so that means it loops through 1 to 10 (this will only print out 1-10 not show many students answered this specific value from 1 to 10 we will get to that later). It will print out "Response" first then for proper spacing we used setw(2) so the numbers line up nicely. It will show the rating righter after it then a ": " we add another setw(2) for spacing purposes then print out how many students answered this specific value from the 1-10 rating. frequency[rating] if the value of rating = 1 it will show how many people answered "1", and after it prints the number of students who answered that value it will print out "student" then endl to go the next line.

Conclusion

This was a very tiring hands-on activity, and I will be honest, it fried my brain in so many places trying to understand how the code works and then explaining why it works right after that. I am thankful though, it gave me a better understanding of how the code works and how each line operates. Without this activity, I would probably still be really confused about how arrays work, get a lot of error codes, and generally just suffer in the future even more than now. I've learned new things like #define and setw() for spacing. Actually, it took me so long trying to figure out how to do the spacing because I was so determined to figure it out by myself without searching online for a solution. I gave up eventually and ended up trying to figure out how setw() worked and then implemented it in my code. What took me the longest was the student response summarization. I had no idea what I was going to do; it took me so much searching. I even tried to do if-else statements for the code. I mean, they do work, but it makes my code look janky. By the grace of God, I managed to make another array to store the frequency of how many times students chose that response from rating 1-10, and the code worked. The next part was me trying to explain this as clearly as possible, which took me a whole hour. Other than that, this activity has taken my whole evening (5 hours), but again, I'm thankful that I've expanded my knowledge to figure this entire activity out.

Assessment Rubric

Rubric for SO 7 (13)

Criteria	Ratings						Pts
SO 7 PI 1 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently and applies knowledge learned into practice	5 pts Good | Educational interests and pursuits exist and flourish outside classroom requirements,knowledge and/or experiences are pursued independently	4 pts Satisfactory | Look beyond classroom requirements, showing interest in pursuing knowledge independently	3 pts Unsatisfactory | Begins to look beyond classroom requirements, showing interest in pursuing knowledge independently	2 pts Poor | Relies on classroom instruction only	1 pts Very Poor | No initiative or interest in acquiring new knowledge	6 pts
SO 7 PI 2 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Completes an assigned task independently and practices continuous improvement	5 pts Good | Completes an assigned task without supervision or guidance	4 pts Satisfactory | Requires minimal guidance to complete an assigned task	3 pts Unsatisfactory | Requires detailed or step-by-step instructions to complete a task	2 pts Poor | Shows little interest to complete a task independently	1 pts Very Poor | No interest to complete a task independently	6 pts
SO 7 PI 3 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Synthesizes and integrates information from a variety of sources; formulates a clear and precise perspective; draws appropriate conclusions	5 pts Good | Evaluate information from a variety of sources; formulates a clear and precise perspective.	4 pts Satisfactory | Analyze information from a variety of sources; formulates a clear and precise perspective.	3 pts Unsatisfactory | Apply the gathered information to formulate the problem	2 pts Poor | Gather and summarized the information from a variety of sources but failed to formulate the problem	1 pts Very Poor | Gather information from a variety of sources	6 pts
SO 7 PI 4 IILO4 Utilize lifelong learning skills in pursuit of personal development and excellence in professional practice. threshold: 4.8 pts	6 pts Excellent | Ideas are combined in original and creative ways in line with the new and emerging technology trends to solve a problem or address an issue.	5 pts Good | Ideas are creative and adapt the new knowledge to solve a problem or address an issue	4 pts Satisfactory | Ideas are creative in solving a problem, or address an issue	3 pts Unsatisfactory | Shows some creative ways to solve the problem	2 pts Poor | Shows initiative and attempt to develop creative ideas to solve the problem	1 pts Very Poor | Ideas are copied or restated from the sources consulted	6 pts
						Total Points: 24	