



# HOMEWORK 1

SUBJECT NAME: Object-Oriented Programming and Machine Learning

SUBJECT CODE	: ECE-590
STUDENT NAME	: Meenakshi Sridharan Sundaram
STUDENT ID	: A20592581
STUDENT EMAIL	: msridharansundaram@hawk.iit.edu
PROFESSOR NAME	: Dr. Won-Jae Yi
DATE	: 25 <sup>TH</sup> SEPTEMBER 2024

## I. AIM:

To create a report that includes all answers to the questions below including codes, actual outputs vs desired outputs.

## II. PLATFORM USED:

The codes were modified and run in VScode after adding, installing, and configuring the necessary C++ environment and by saving the file names with an extension of **.cpp**. The same code could also be run in **TurboC** but **VScode** is recommended due to its integration with a Virtual Machine

## III. BODY:

### 1. Question:

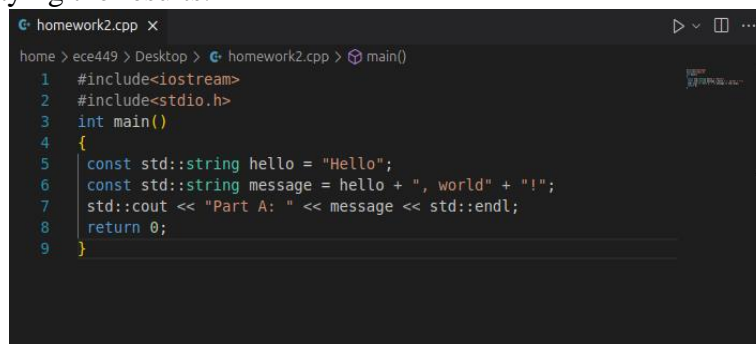
Compile and run the program below A and B. Write a C++ style output function (printing on the screen) to show the data in a message. Are these definitions valid? Why or why not? Explain your answer along with the screenshot of your output and your code. You can utilize VScode from your VM.

**A.**     `const std::string hello = "Hello";`  
`const std::string message = hello + ", world" + "!"`;

**B.**     `const std::string exclaim = "!"`;  
`const std::string message = "Hello" + ", world" + exclaim`;


### 1.1. Answer :

**A.**     The definition given in part A is valid with the proper header files, and main, return functions shown as a snippet in Figure 1.1. When it was compiled in VScode it showed no error and yielded the output given in Figure 1.2. It holds valid and yields the correct output as a constant string named hello is created in which the object “Hello” is stored. In the other line, another constant string named message is created in which the addition of the object of the string named hello(in this case “hello”) and “world” which is a string literal is added using the string operand + to create an object “hello world” and the entire object is added to an “!” which produces a string object “Hello world!” stored in the const message which is furthermore printed for displaying the results.



```
homework2.cpp x
home > ece449 > Desktop > homework2.cpp > main()
1  #include<iostream>
2  #include<stdio.h>
3  int main()
4  {
5      const std::string hello = "Hello";
6      const std::string message = hello + ", world" + "!";
7      std::cout << "Part A: " << message << std::endl;
8      return 0;
9  }
```

*Figure 1.1: Code of Part A*



```
Part A: Hello, world!
[1] + Done
"/usr/bin/gdb" --interpreter=mi --
tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-p0g4vyod.tg3" 1>"/tmp/
Microsoft-MIEngine-Out-adwjnmb.0sb"
ece449@ece449:~/my449$
```

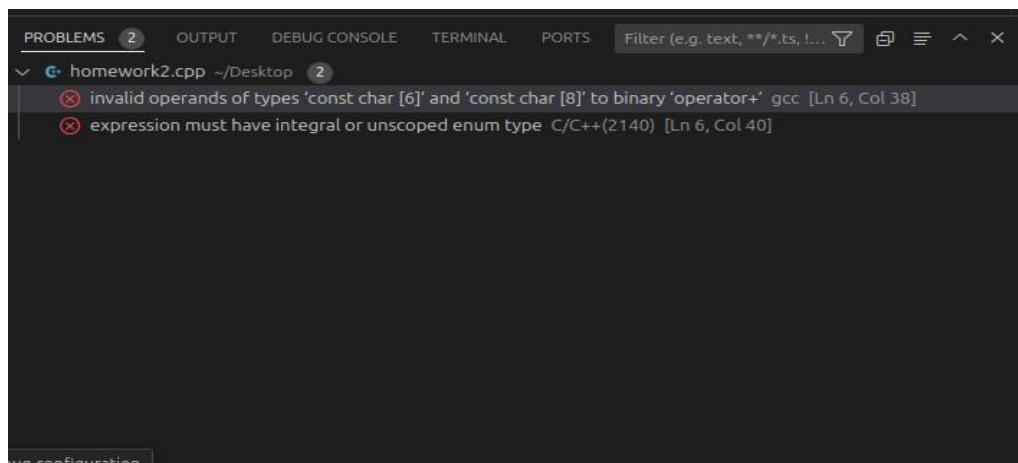
*Figure 1.2: Output of Part A*

The program given in part B was run in VSCode and the output yielded an error of type mismatch using the “+” operand. The compilation error is due to adding two strings literal “Hello” and “world,” which is impossible in C++. The respective code and its output are mentioned in Figure 1.3 and Figure 1.4.



```
home > ece449 > Desktop > homework2.cpp > main()
1 #include<iostream>
2 #include<stdio.h>
3 int main()
4 {
5     const std::string exclaim = "!";
6     const std::string message = "Hello" + ", world" + exclaim;
7     std::cout << "Part B " << message << std::endl;
8     return 0;
9 }
```

*Figure 1.3: code of part 2*



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, **/*.ts, ...)
homework2.cpp ~/Desktop 2
invalid operands of types 'const char [6]' and 'const char [8]' to binary 'operator+' gcc [Ln 6, Col 38]
expression must have integral or unscoped enum type C/C++(2140) [Ln 6, Col 40]
```

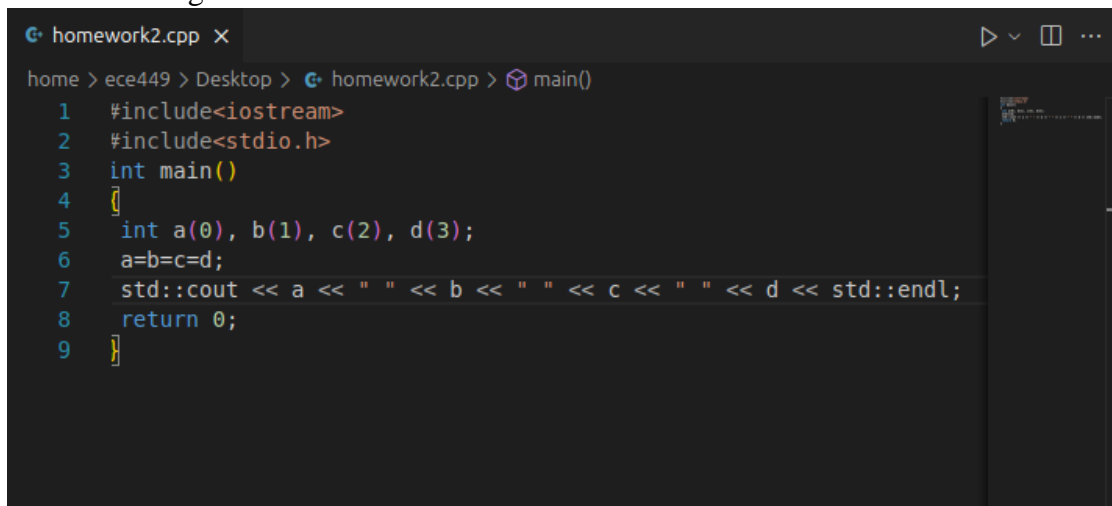
*Figure 1.4: Compilation error showed in part B*

2. **QUESTION:** The assignment operator = works with two operands L and R in the form L=R. For the following code to generate an output of 3 3 3 3, what should be the associativity of = and what should be the result and the side effects of L=R? Confirm your answer by executing the below program in C++, provide the screenshot of your code and result to back up your answer.

```
int a(0), b(1), c(2), d(3);
a=b=c=d;
std::cout << a << " " << b << " " << c << " " << d << std::endl;
```

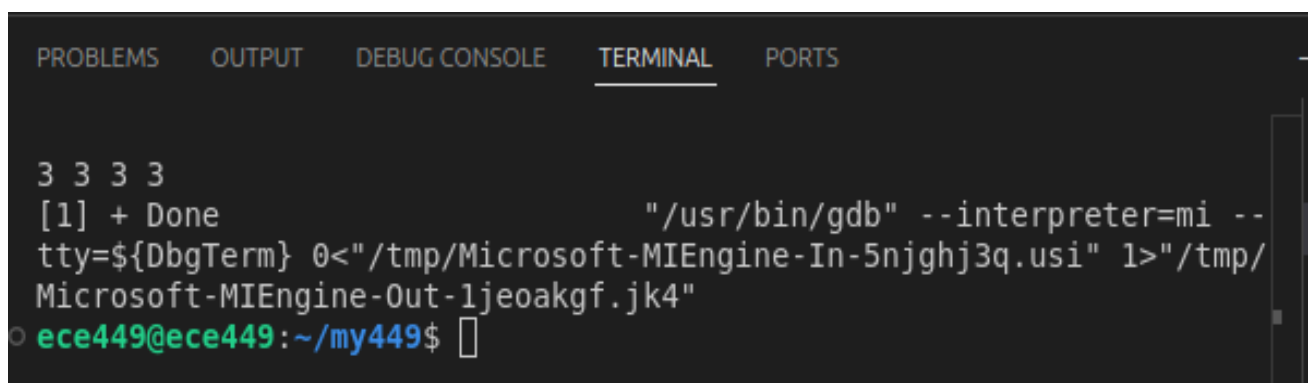
- 2.1. **ANSWER:** In the given code snippet, the operand “=” is used to assign values to the given variables. Since **d** has the value of **3**, the associativity of “=” is from **right to left**. The value of the variable **d** which is **3** is passed on to **c**, the current value of the variable **c** which is **3** in this instance is passed on to **b** and the chain carries on to yield the final output of **3 3 3 3** since the **cout** statement gives out the output.

The code was run in the C++ platform as shown in Figure 2.1 and the output it yielded is shown in Figure 2.2.



```
homework2.cpp X
home > ece449 > Desktop > homework2.cpp > main()
1 #include<iostream>
2 #include<stdio.h>
3 int main()
4 {
5     int a(0), b(1), c(2), d(3);
6     a=b=c=d;
7     std::cout << a << " " << b << " " << c << " " << d << std::endl;
8     return 0;
9 }
```

Figure 2.1: correct code



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
3 3 3 3
[1] + Done
"/usr/bin/gdb" --interpreter=mi --
tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-5njghj3q.usi" 1>"/tmp/
Microsoft-MIEngine-Out-1jeoakgf.jk4"
ece449@ece449:~/my449$
```

Figure 2.2: output of the program

3. **QUESTION :** The following program attempts to copy from `u` into `v`. Explain why this is an incorrect method.

```
std::vector u(10, 100);
```

```
std::vector v; std::copy(u.begin(), u.end(), v.begin());
```

Correct the above program. There are at least two possible ways to correct the program but you are only required to implement one.

- 3.1. **ANSWER:** The above-mentioned code snippet was run in VScode and the results are shown in Figure 3.1 where the environment looked crashed. This happens because vector `u` has the memory space to contain 10 values which are all assigned 100 but vector `v` has no memory assigned to it. Hence, when the copy command is run there's confusion as to where the copying is starting from since vector `v` has no beginning or end. This could be corrected if vector `v` is assigned the same memory as vector `u` and by beginning the code with proper header files as shown in Figure 3.2 and its output is verified in Figure 3.3.

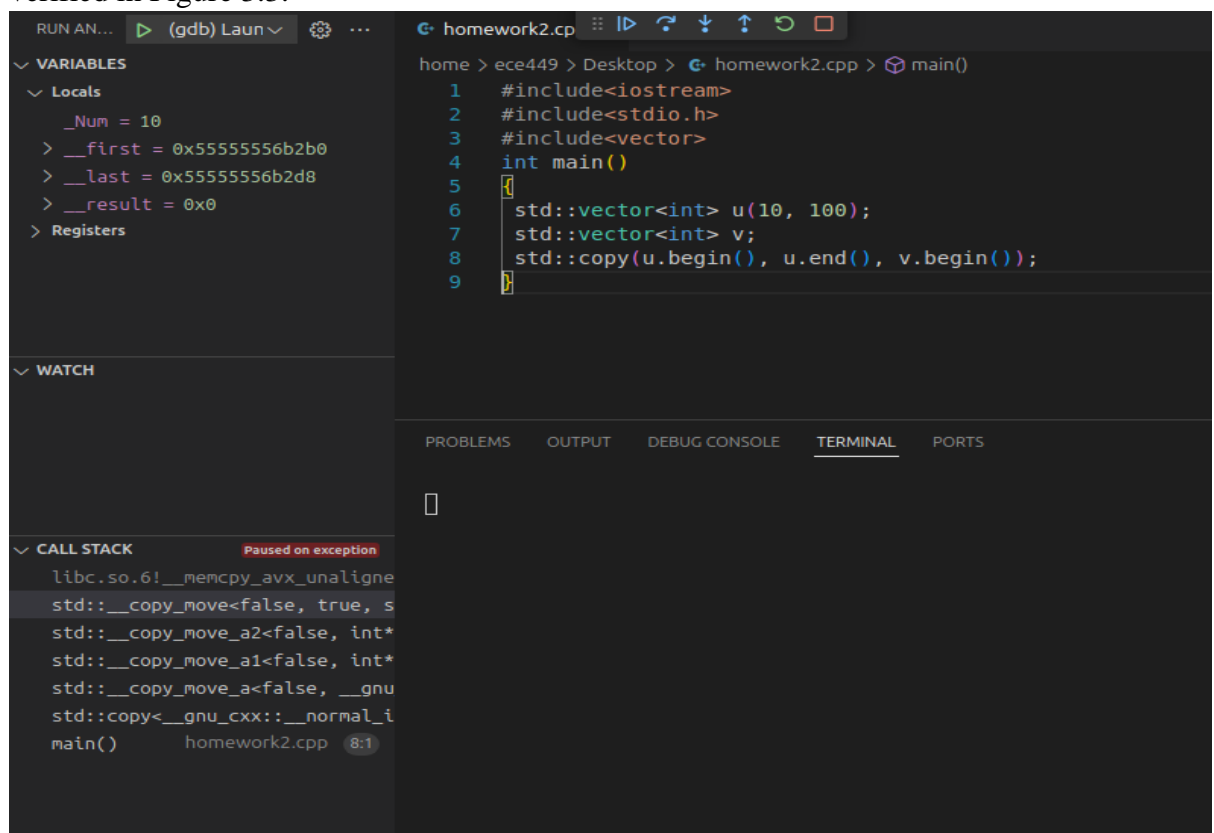
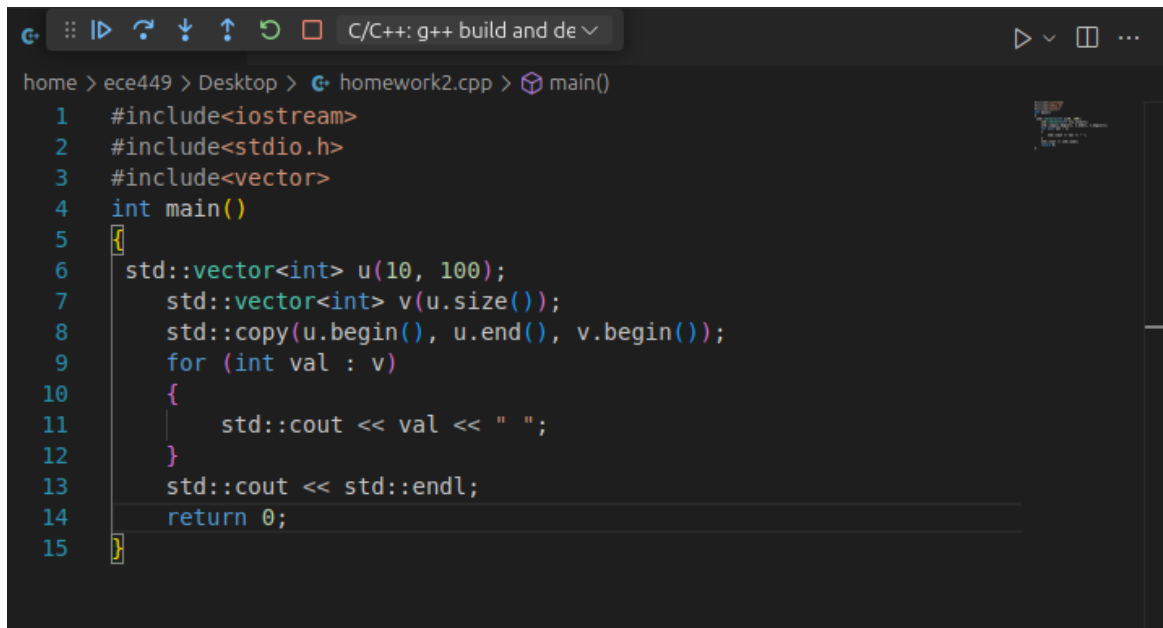


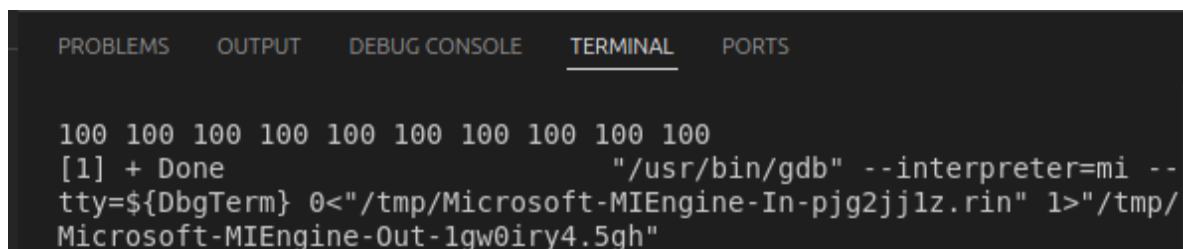
Figure 3.1: Runtime error which is manually paused in the call stack due to lack of memory.

The code was corrected using and run after including the proper header files and the output is printed to show if the elements of `u` have been copied into the elements of `v(u.size())`; to have the same memory space as `u` and the command `std::cout<<val<<" "`; helps us to print the contents of `v` after converting the vector to `int` type and running it in a for loop



```
home > ece449 > Desktop > homework2.cpp > main()
1  #include<iostream>
2  #include<stdio.h>
3  #include<vector>
4  int main()
5  {
6      std::vector<int> u(10, 100);
7      std::vector<int> v(u.size());
8      std::copy(u.begin(), u.end(), v.begin());
9      for (int val : v)
10     {
11         std::cout << val << " ";
12     }
13     std::cout << std::endl;
14     return 0;
15 }
```

Figure 3.2: Corrected code



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

100 100 100 100 100 100 100 100 100 100
[1] + Done
"/usr/bin/gdb" --interpreter=mi --
tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-pjg2jjlz.rin" 1>"/tmp/
Microsoft-MIEngine-Out-lgw0iry4.5gh"
```

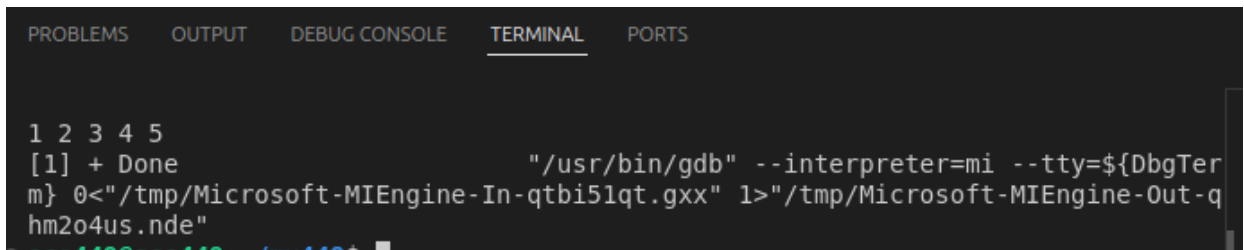
Figure 3.3: obtained output

4. QUESTION: Suppose you have a `std::vector temp = { 1, 2, 3, 4, 5 }` Write a short C++ program using an iterator to print each element in this vector. Include a screenshot of your code and the output.
- 6.1 ANSWER: Iterators are used in C++ to provide a unified accessibility to the elements of different types of containers such as arrays, vectors and lists. These elements can be accessed, utilised, and manipulated using an iterator. In this case, we use an iterator to print each vector element named temp. Figure 4.1 shows the code being executed in VScode and its respective output is shown in Figure 4.2



```
homework2.cpp x
home > ece449 > Desktop > homework2.cpp > main()
1  #include<iostream>
2  #include<stdio.h>
3  #include<vector>
4  int main()
5  {
6      std::vector<int> temp = {1, 2, 3, 4, 5};
7      for (std::vector<int>::iterator it = temp.begin(); it != temp.end(); ++it)
8      {
9          std::cout << *it << " ";
10     }
11     std::cout << std::endl;
12     return 0;
13 }
```

Figure 4.1: code to print out vector elements using iterator



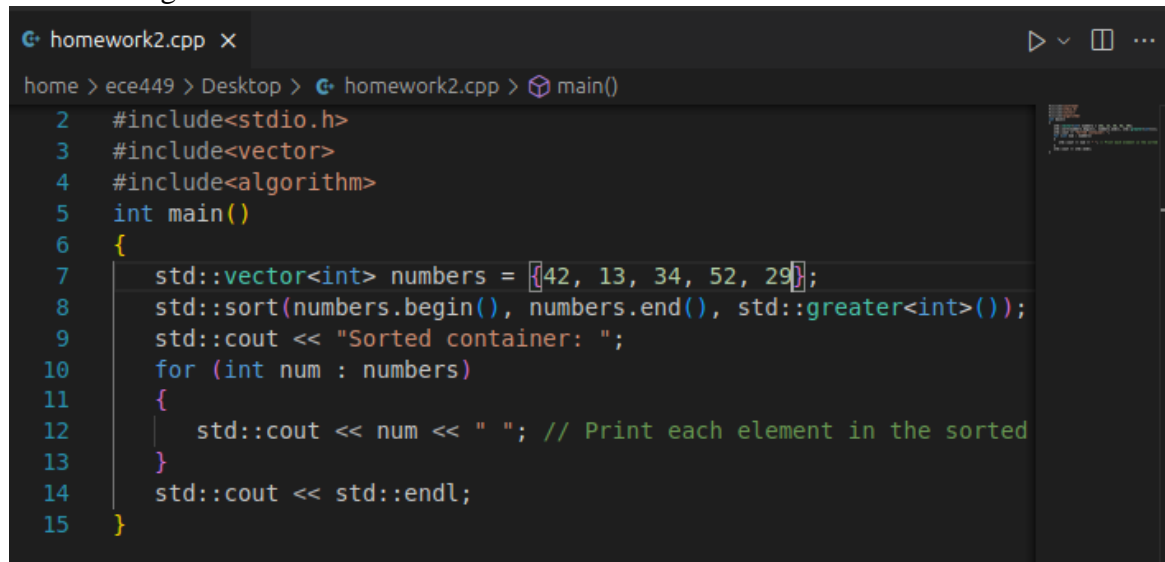
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

1 2 3 4 5
[1] + Done                               "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm}
m} 0<"/tmp/Microsoft-MIEngine-In-qtbi51qt.gxx" 1>"/tmp/Microsoft-MIEngine-Out-q
hm2o4us.nde"

ece449@ece449:~/my449$
```

Figure 4.2: code output

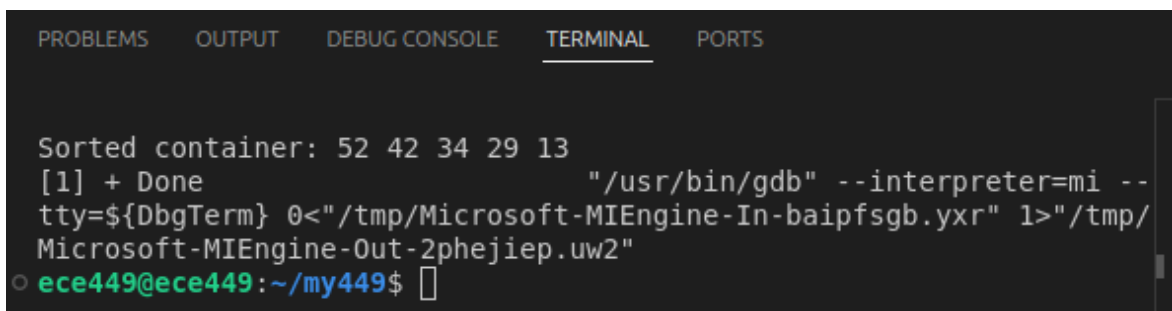
- 5 QUESTION: Suppose integers are containers with int elements. Implement a function to sort integers from the largest to the smallest. Include a sample container to prove your program is working. Provide your code and screenshot of the output. (Hint: use `std::sort`).
- 5.1 ANSWER: Containers in C++ consist of a collection of elements in an organised way. These elements can be accessed and manipulated using iterators and member functions. The code snippet is given in Figure 5.1 and its output shows numbers sorted in descending order in Figure 5.2.



```
homework2.cpp X
home > ece449 > Desktop > homework2.cpp > main()

2 #include<stdio.h>
3 #include<vector>
4 #include<algorithm>
5 int main()
6 {
7     std::vector<int> numbers = {42, 13, 34, 52, 29};
8     std::sort(numbers.begin(), numbers.end(), std::greater<int>());
9     std::cout << "Sorted container: ";
10    for (int num : numbers)
11    {
12        std::cout << num << " "; // Print each element in the sorted
13    }
14    std::cout << std::endl;
15 }
```

Figure 5.1: code for sorting out elements in descending order



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Sorted container: 52 42 34 29 13
[1] + Done                               "/usr/bin/gdb" --interpreter=mi --
tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-baipfsgb.yxr" 1>"/tmp/M
Microsoft-MIEngine-Out-2phejiej.uw2"
ece449@ece449:~/my449$
```

Figure 5.2: output showing numbers sorted out in descending order

6. QUESTION: Compile and execute hw1\_q6.cpp (from Canvas).
- A. Write line-by-line comments to this program.
- B. Discuss the output of this program. Run it a couple of times. Try with different values of `max_size`. Take screenshots of your outcomes. Explain why vector/list is always faster than the other.

**6.1 ANSWER:** The answers to part A and part B of the questions are discussed below

**A.** The comments are used in C++ for a better intuitive understanding of the given program by the reader. They are typed inside “//.” Figure 6.1 will show a snippet of the code where line-by-line comments are given for finding a random number in a list and a vector and finding the time duration it took to find them. Below is the line of commands

1. `//including iostream for input and output operations`
2. `//including algorithm for using sort function`
3. `//including list for using elements in a container`
4. `//including vector to use std::vector container`
5. `//including chrono for time related functionality`
6. `//the main function indicating the beginning of the program`
7. `{`
8. `//an empty vector consists of integers whose name is integers_vectors`
9. `//an empty list consists of integers whose name is integer_list`
10. `empty line`
11. `//setting the maximum size of elements`
12. `Empty line`
13. `//printing out that the values are to be inserted into the vector and the list`
14. `//for loop to initialize inserting values from 0 to max size-1`
15. `{`
16. `//inserting value, i at the back of the vector`
17. `//inserting value of I at the back of the list`
18. `}`
19.
20. `//to generate a random number between 1 and max_size`
21. `empty line`
22. `//printing out the statement search for the number in the list and printing it out`
23. `empty line`
24. `//line to print out the search in vector`
25. `//recording the time when a search is initiated when the end of the vector is reached`
26. `//waiting to find the number`
27. `//recording the time when the searching stops when the end of the vector is reached`
28. `Empty line`
29. `//line to print out the search in lists`
30. `//recording the time when searching is initiated in the starting of the list`
31. `//waiting to find the random number in the list`
32. `//recording the time when searching stops when the end of the list is reached`
33. `Empty line`
34. `//calculating the time took in the vector to find the number`
35. `//calculating the time took to find the number in the list`
36. `Empty line`
37. `//time taken to find the number in vector in ms`
38. `//time taken to find the number in the list in ms`
39. `Empty line`
40. `If returned 0, the program shows that it has executed successfully`
41. `//closing of the main function`



```

1 #include <iostream> //including iostream for input and output operations
2 #include <algorithm> //including algorithm for using sort functions
3 #include <list> // including list for using elements in a container
4 #include <vector> // including vector to use std::vector container
5 #include <chrono> //including chrono for time related functionality
6 int main() // The main function indicating the beginning of the program
7 {
8     std::vector<int> integers_vector; //an empty vector consists of integers whose name is integers_vector
9     std::list<int> integers_list; //an empty list consists of integers whose name is integers_list
10
11     size_t max_size = 100000000; //setting the maximum size of elements to insert to be 100000000
12
13     std::cout << "inserting values into vector and list..." << std::endl; //printing out that the values are to be inserted in the vector and the list
14     for(size_t i = 0; i < max_size; i++) //for loop to initialize inserting values from 0 to max_size - 1
15     {
16         integers_vector.push_back(i); //inserting value i at the back of the vector
17         integers_list.push_back(i); //inserting values of i at the back of the list
18     }
19
20     size_t random_number = rand() % max_size + 1; //to generate a random number between 1 and max_size
21
22     std::cout << "random number to find in vector and list is: " << random_number << std::endl; //printing out the statement search for the number in the list and vector and printing it out
23
24     std::cout << "searching in vector..." << std::endl; //line to print out the search in vector
25     auto start_vector = std::chrono::high_resolution_clock::now(); //recording the time when search is initiated in the beginning of the vector
26     std::find(integers_vector.begin(), integers_vector.end(), random_number); //waiting to find the number
27     auto end_vector = std::chrono::high_resolution_clock::now(); //recording the time when search is terminated when the end of the vector is reached
28
29     std::cout << "searching in list..." << std::endl; //line to print out the search in lists
30     auto start_list = std::chrono::high_resolution_clock::now(); //recording the time when searching initiated in the starting of the list
31     std::find(integers_list.begin(), integers_list.end(), random_number); //waiting to find the random number in the list
32     auto end_list = std::chrono::high_resolution_clock::now(); //recording the time when searching stops when the end of the list is reached
33
34     std::chrono::duration<double, std::milli> vector_time = end_vector - start_vector; //calculating the time took in the vector to find the number
35     std::chrono::duration<double, std::milli> list_time = end_list - start_list; //calculating the time time took to find the number in the list
36
37     std::cout << "Time took searching " << random_number << " in vector: " << vector_time.count() << "ms" << std::endl; //time taken to find the number in vector in ms
38     std::cout << "Time took searching " << random_number << " in list: " << list_time.count() << "ms" << std::endl; //time taken to find the number in the list in ms
39
40     return 0; //if returned 0, the program shows that it has executed successfully
41 } //closing of the main function

```

**Figure 6.1: program code with in-line comments**

```

inserting values into vector and list...
random number to find in vector and list is: 4289384
searching in vector...
searching in list...
Time took searching 4289384 in vector: 31.605ms
Time took searching 4289384 in list: 78.3025ms
[1] + Done

```

**Figure 6.2: output of the code for max=100000000**

- B. The output mentioned in Figure 6.2 shows how a random number(4289385) was chosen and the time took to search in vector and the list is calculated and printed. Furthermore, the maximum number was changed to 99675 and its output is shown in figure 6.3. later, the max value was set to 82828228 and 1098 and its output is given in figure 6.4 and figure 6.5. The time it took to find in vector is observed to be less than the time took to find the number in the list irrespective of the maximum number. It is so because a vector has an array-like structure and when one element is accessed the other elements are kept in the cache thus providing quicker accessibility while in a list the structure is like that of a linked list and accessing and finding its element becomes a bit complicated as the pointer to the element has to be referenced and dereferenced over and over again.

```

PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL  PORTS

inserting values into vector and list...
random number to find in vector and list is: 72209
searching in vector...
searching in list...
Time took searching 72209 in vector: 0.474966ms
Time took searching 72209 in list: 1.34766ms
[1] + Done

```

**Figure 6.3: Output for maximum val=99675**

```
inserting values into vector and list...
random number to find in vector and list is: 64896596
searching in vector...
searching in list...
Time took searching 64896596 in vector: 438.927ms
Time took searching 64896596 in list: 1153.97ms
[1] + Done                                     "/usr/bin/qdb" --interpreter=mi
```

*Figure 6.4: Output for maximum val=82828228*

```
inserting values into vector and list...
random number to find in vector and list is: 884
searching in vector...
searching in list...
Time took searching 884 in vector: 0.0059ms
Time took searching 884 in list: 0.015577ms
[1] + Done                                     "/usr/bin/qdb" --
```

*Figure 6.5: output for maximum val=1098*