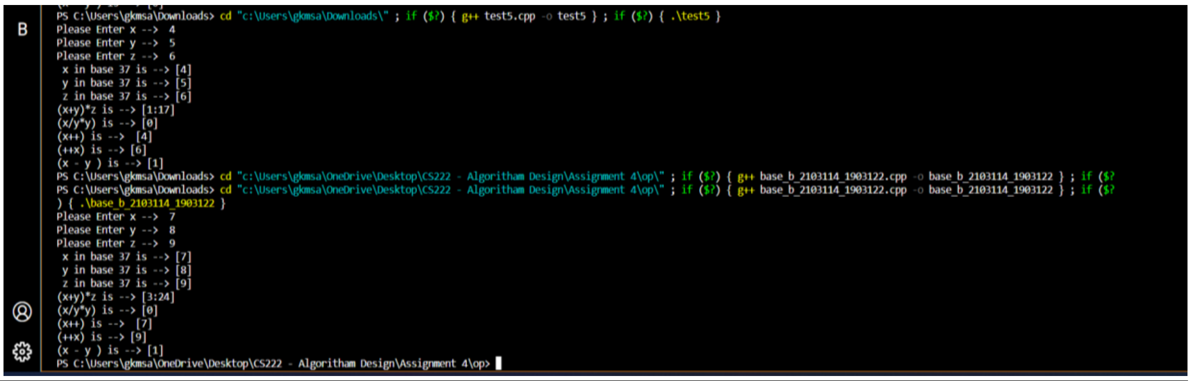
CS 222
Assignment 3

Name – Gautam Kumar Mahar and Kanwar Raj Singh
Roll No. – 210311 and 1903122
Branch – Computer Science Engineering

Question 1)

- Output -



```
PS C:\Users\gkmsa\Downloads> cd "c:\Users\gkmsa\Downloads\" ; if ($?) { g++ test5.cpp -o test5 } ; if ($?) { .\test5 }
Please Enter x -->  4
Please Enter y -->  5
Please Enter z -->  6
 x in base 37 is --> [4]
 y in base 37 is --> [5]
 z in base 37 is --> [6]
(x+y)*z is --> [1:17]
(x/y*y) is --> [0]
(x++) is -->  [4]
(++x) is --> [6]
(x - y ) is --> [1]
PS C:\Users\gkmsa\Downloads> cd "c:\Users\gkmsa\OneDrive\Desktop\CS222 - Algoritham Design\Assignment 4\op\" ; if ($?) { g++ base_b_2103114_1903122.cpp -o base_b_2103114_1903122 } ; if ($?
PS C:\Users\gkmsa\Downloads> cd "c:\Users\gkmsa\OneDrive\Desktop\CS222 - Algoritham Design\Assignment 4\op\" ; if ($?) { g++ base_b_2103114_1903122.cpp -o base_b_2103114_1903122 } ; if ($?
) { .\base_b_2103114_1903122 }
Please Enter x -->  7
Please Enter y -->  8
Please Enter z -->  9
 x in base 37 is --> [7]
 y in base 37 is --> [8]
 z in base 37 is --> [9]
(x+y)*z is --> [3:24]
(x/y*y) is --> [0]
(x++) is -->  [7]
(++x) is --> [9]
(x - y ) is --> [1]
PS C:\Users\gkmsa\OneDrive\Desktop\CS222 - Algoritham Design\Assignment 4\op>
```

The temporal complexity of the function Object() { [native code] } function, where n is the input base, is O(log b(n)). This is because the base is split by b in each iteration of the for loop, which runs until base = 0. iteration, therefore log b is the number of iterations (n). The total time complexity is O(log b(n)) since calling the reverse function takes so long.

O(max(m, n), where m is the number of digits in the first operand and n is the number of digits in the second operand, is the temporal complexity of the operator+() function. This is due to the fact that with each iteration of the for loop, either m or n decreases by 1 until they both equal 0. The entire time complexity is O(max(m, n)) since the reverse function call requires O(max(m, n)) time.

This operator-() function has a temporal complexity of O(max(m,n)), where m is the size of the data for the first base b object and n is the size of the data for the second base b object. This is due to the fact that the code subtracts by iterating through the data of both base b objects. Until one of the data arrays has been fully processed, the loop is repeated. The loop's size depends on the maximum value of m and n. As a result, the temporal complexity is O(max(m,n)).

O(n), where n is the size of the data for the base b objects being compared, is the temporal complexity of the operator () function. This is so that the code can identify which base b object has smaller data by repeatedly looping through both data. The loop keeps going until either all the data pieces have been compared and it is decided that the objects are equal, or until a difference is found and the loop is broken. The loop will be run n times, resulting in an O time complexity in the worst-case scenario where the data amount is n. (n).

The operator() function has an O(n) time complexity because the for loop performs the subtraction operation n times. This is because each iteration of the for loop necessitates a base-b object subtraction technique that takes a linear amount of time in relation to the size of the objects. Since the loop continues until the divisor exceeds the dividend, the number of loop iterations is proportional to the size of the dividend and the divisor. O is hence the total time complexity (n)

The operator++() function has an O(n) time complexity, where n is the number of digits in the number that the data vector represents. This is so that each iteration of the for loop, which has a maximum number of n iterations, accomplishes a fixed amount of work. The temporal complexity of the calls to reverse is also O(n).

The number of digits in the number that the data vector represents, denoted by n, determines the operator++() function's O(n) time complexity. This is done to ensure that each iteration of the for loop, which can execute n times in total, completes a set amount of work. The calls to reverse's temporal complexity is also O (n).

The temporal complexity of this operator++() method is O(n), where n is the number of digits in the number that the data vector represents. This is because the function invokes the operator++() function, for also has an O(n) time complexity, after making a duplicate of the object, ans, which requires O(n) time (n).

The operator=() function has a time complexity of O(n), where n is the number of digits in the number represented by the equal.data vector. This is due to the function resizing the data vector to the size of the equal.data vector and then performing an O(n) time loop that copies each element from equal.data to data.

The temporal complexity of the operator=() function is O(n), where n is the number of digits in the number that the equal.data vector represents. Due to the function's scaling of the data vector to equal.data's size followed by an O(n) time loop that copies each element from equal.data to data, this occurs.

- **Thanks -**