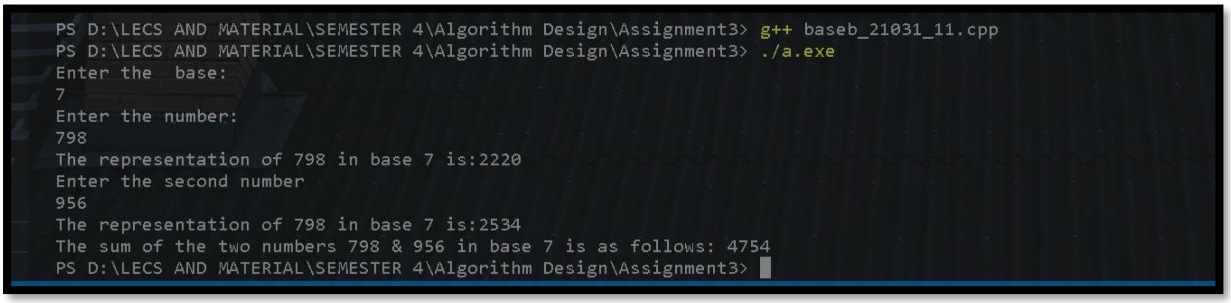
INDIAN INSTITUTE OF TECHNOLOGY GOA

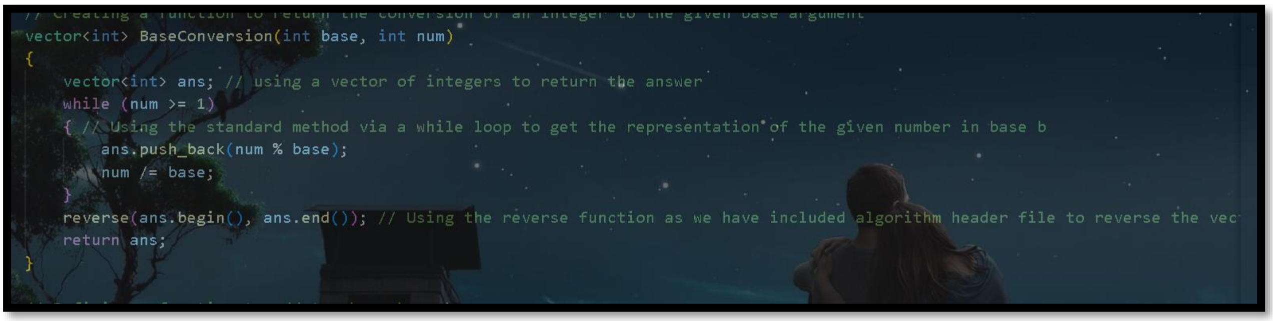
Name: Aditya Rajesh Bawangade

Roll Number: 2103111



```
PS D:\LECS AND MATERIAL\SEMESTER 4\Algorithm Design\Assignment3> g++ baseb_21031_11.cpp
PS D:\LECS AND MATERIAL\SEMESTER 4\Algorithm Design\Assignment3> ./a.exe
Enter the  base:
7
Enter the number:
798
The representation of 798 in base 7 is:2220
Enter the second number
956
The representation of 798 in base 7 is:2534
The sum of the two numbers 798 & 956 in base 7 is as follows: 4754
PS D:\LECS AND MATERIAL\SEMESTER 4\Algorithm Design\Assignment3>
```

The output obtained is as given above.

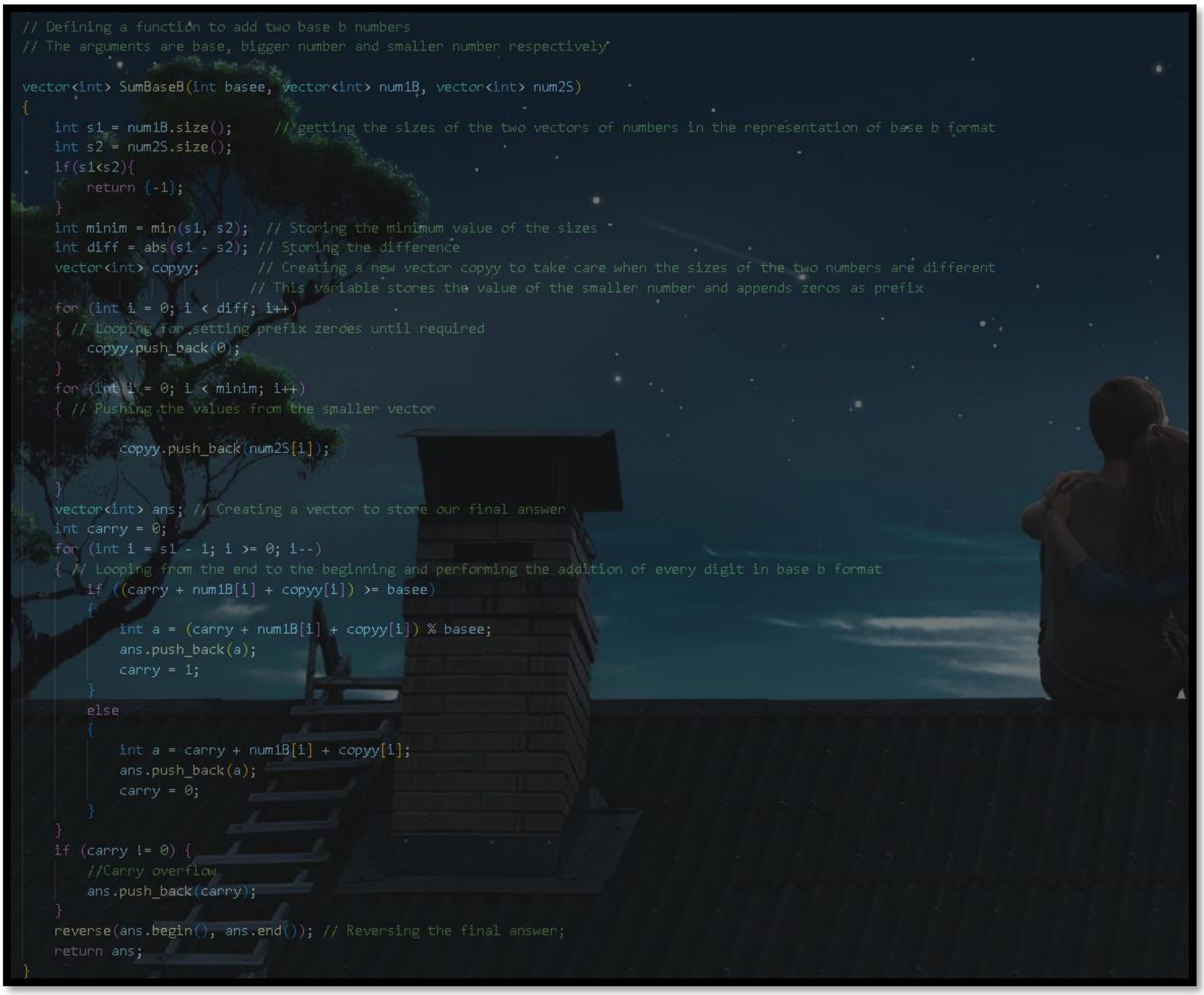


```
// Creating a function to return the conversion of an integer to the given base argument
vector<int> BaseConversion(int base, int num)
{
    vector<int> ans; // using a vector of integers to return the answer
    while (num >= 1)
    { // Using the standard method via a while loop to get the representation of the given number in base b
        ans.push_back(num % base);
        num /= base;
    }
    reverse(ans.begin(), ans.end()); // Using the reverse function as we have included algorithm header file to reverse the vec
    return ans;
}
```

Having a closer look at the code for converting the number “num” to the base “b” representation; we define a vector “ans” and using a while loop, keep on dividing the number “num” by the base “b” and “push_back” the remainder using the modulo operator to the “ans” vector. Thus, at every iteration, the value of “num” is decremented by a factor of “(1/b)”. Thus, log(num) steps are required in the given while loop.

Now, in the second last line, as we are using vector for storing the number in base “b” representation; we won’t have the liberty to use reverse indexing and avoid reversing the vector manually; thus, in order to reverse the vector, we use the reverse function which comes with the algorithm header file as a part of the C++ STL. This, algorithm thus requires O(N) time, where “N” is the size of the integer “num” (number of digits in N) in the given representation (decimal).

Thus, the overall time complexity will be O(log(num)).



```cpp
// Defining a function to add two base b numbers
// The arguments are base, bigger number and smaller number respectively

vector<int> SumBaseB(int basee, vector<int> num1B, vector<int> num2S)
{
    int s1 = num1B.size();     // getting the sizes of the two vectors of numbers in the representation of base b format
    int s2 = num2S.size();
    if(s1<s2){
        return {-1};
    }
    int minim = min(s1, s2);  // Storing the minimum value of the sizes
    int diff = abs(s1 - s2); // Storing the difference
    vector<int> copyy;       // Creating a new vector copyy to take care when the sizes of the two numbers are different
                            // This variable stores the value of the smaller number and appends zeros as prefix
    for (int i = 0; i < diff; i++)
    { // Looping for setting prefix zeroes until required
        copyy.push_back(0);
    }
    for (int i = 0; i < minim; i++)
    { // Pushing the values from the smaller vector

            copyy.push_back(num2S[i]);

    }
    vector<int> ans; // Creating a vector to store our final answer
    int carry = 0;
    for (int i = s1 - 1; i >= 0; i--)
    { // Looping from the end to the beginning and performing the addition of every digit in base b format
        if ((carry + num1B[i] + copyy[i]) >= basee)
        {
            int a = (carry + num1B[i] + copyy[i]) % basee;
            ans.push_back(a);
            carry = 1;
        }
        else
        {
            int a = carry + num1B[i] + copyy[i];
            ans.push_back(a);
            carry = 0;
        }
    }
    if (carry != 0) {
        //Carry overflow
        ans.push_back(carry);
    }
    reverse(ans.begin(), ans.end()); // Reversing the final answer;
    return ans;
}
```

Having a look at this algorithm, we can see that the addition is being carried out for each index. Rest all the operations are constant time operations or the operations which require time less than or equal to O(N).Thus, the overall time complexity shall be O(N), where N is the size of the largest vector of integers i.e., the vector num1B.