**BY- AYUSH MAHANT**

**2019353**

**CSSS BRANCH**

**GROUP-9**

**BOOTH’s ALGORITHM-**

**DOCUMENTATION**

Booth's Algorithm description-

Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation in an efficient way, i.e., less number of additions/subtractions required. It operates on the fact that strings of 0's in the multiplier need no addition, but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^(k+1 ) to 2^m.

As in all multiplication schemes, the booth algorithm requires examination of the multiplier bits and shifting of the partial product. Before the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to the following rules:

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
2. The multiplicand is added to the partial product upon encountering the first 0 (provided there was a previous '1') in a string of 0's in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

(<https://www.geeksforgeeks.org/computer-organization-booths-algorithm/>)

LANGUAGE USED-

I have written my code in JAVA. JAVA is currently being taught this semester, and I am more comfortable in solving systems in this language. Also, JAVA is a faster language.

EXPLANATION OF THE CODE-

In the main method, I have input the two numbers using Scanner class. The two numbers are converted to their binary forms using the function convertBinary().

If both the numbers are greater than or equal to 0 or are positive, then, to make the strings containing binary forms of the numbers to length 33, I have added '0' to them. I print the binary forms of the numbers.

If the first number is less than 0 or is negative and the second number is positive, I take the two's complement of the negative number, and store it as the first string. To convert to two's complement, I have used the function twocomplement(). If both the numbers are less than 0 or are negative, I have taken two's complement of both the numbers.

I finally have the two strings of the binary form of both numbers. To calculate the multiplication values, I used the function calculate(). The function returns the final string of the binary form of the result.

The function convert() converts back the binary number to decimal form, which is the multiplication result of the two numbers.

The following are the functions used and their explanations.

1. String convertBinary(int num) - This function converts the decimal number to its binary equivalent and forms a string and returns it.

2. String twocomplement(String stp1)- This function converts the binary number in its two complements and returns it. The purpose first finds the first '1' from the least bit and then converts all the '0' to '1' and '1 to '0'before the first one from the right and leaves the digits after the first '1' as it is.

3. String calculate(String zaq, String zaq1, String zaq2) - This function does the main task of multiplication. In the booth's algorithm, first, the accumulator is initialized to '0' \* number of bits used. We have the second number and another single bit 'qo' initialized to '0'. The function combines the three and stores it in the string variable fin.

If the first bit of the second number(from the right) and 'qo' are equal, then we need to do the right shift on the string fin. After the right shift, storing back the values to the accumulator, second string, and 'qo' is a must.

If the first bit of the number(from the right) is '1' and 'qo' is '0' then, the function adds the two's complement of the first number to the accumulator. After that , the right shift takes place. If the first bit of the number(from the right) is '0' and 'qo' is '1' then, the function adds the first number to the accumulator. After that, the right shift takes place. All this happens till the while loop stops. The while loop stops when the value of n, i.e., the number of bits given to number equal to 0. After it finishes the final value of string jun, which stores the right-shifted string is returned.

4. String addition(String binary,String binary2) -

This function is used to add the contents of the accumulator to the first number which is sent by the function calculate() and returns it.

5. String rightshift(String lg) -

This function is used to return the right-shifted string to function calculate(). The function first stores the string sent up to the last bit(the last bit not included) and adds the first bit from the left at the front of the new string.

6. int convert(String erp) - This function returns the decimal equivalent of the binary number.

ASSUMPTIONS AND CONSTRAINTS-

1. The code assumes that the number is of a maximum of 32 bits.
2. The range of numbers would be (-2,147,483,647 to +2,147,483,647)
3. If the first number is positive and the second number is negative, then I have reverted them. (the first number is now the second and vice-versa)
4. The user must enter only and only the decimal numbers.
5. The decimal numbers should not contain and ',' and. ''.
6. I have used the scanner class for inputs.
7. No decimal points included.
8. The input and output will be shown on the terminal itself.

OUTPUT FORMAT-

First, printing the binary format of the numbers entered,then the accumulator,second number and the operation performed at each step.

Final answer is the binary format of the number and the resultant number after multiplication.

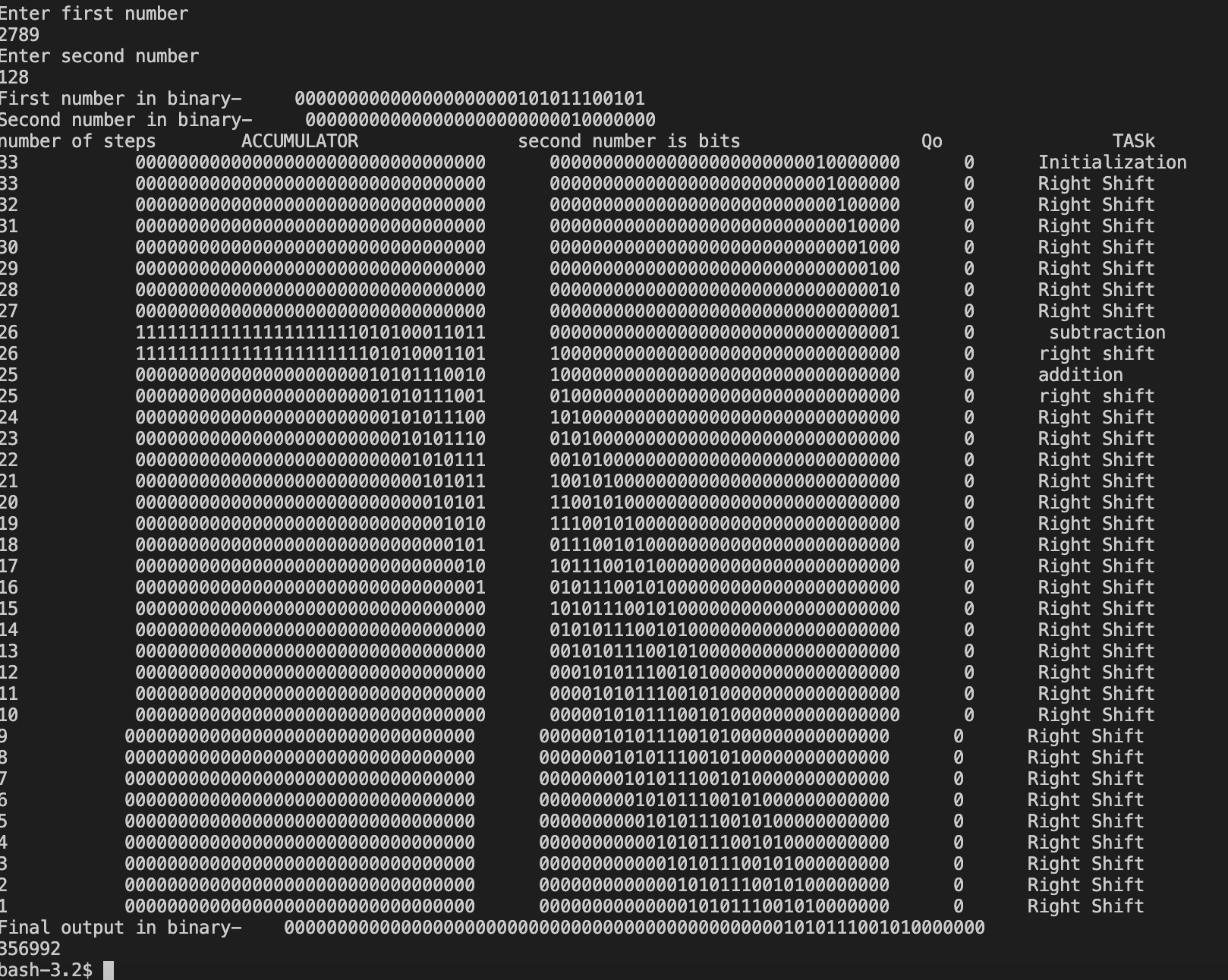
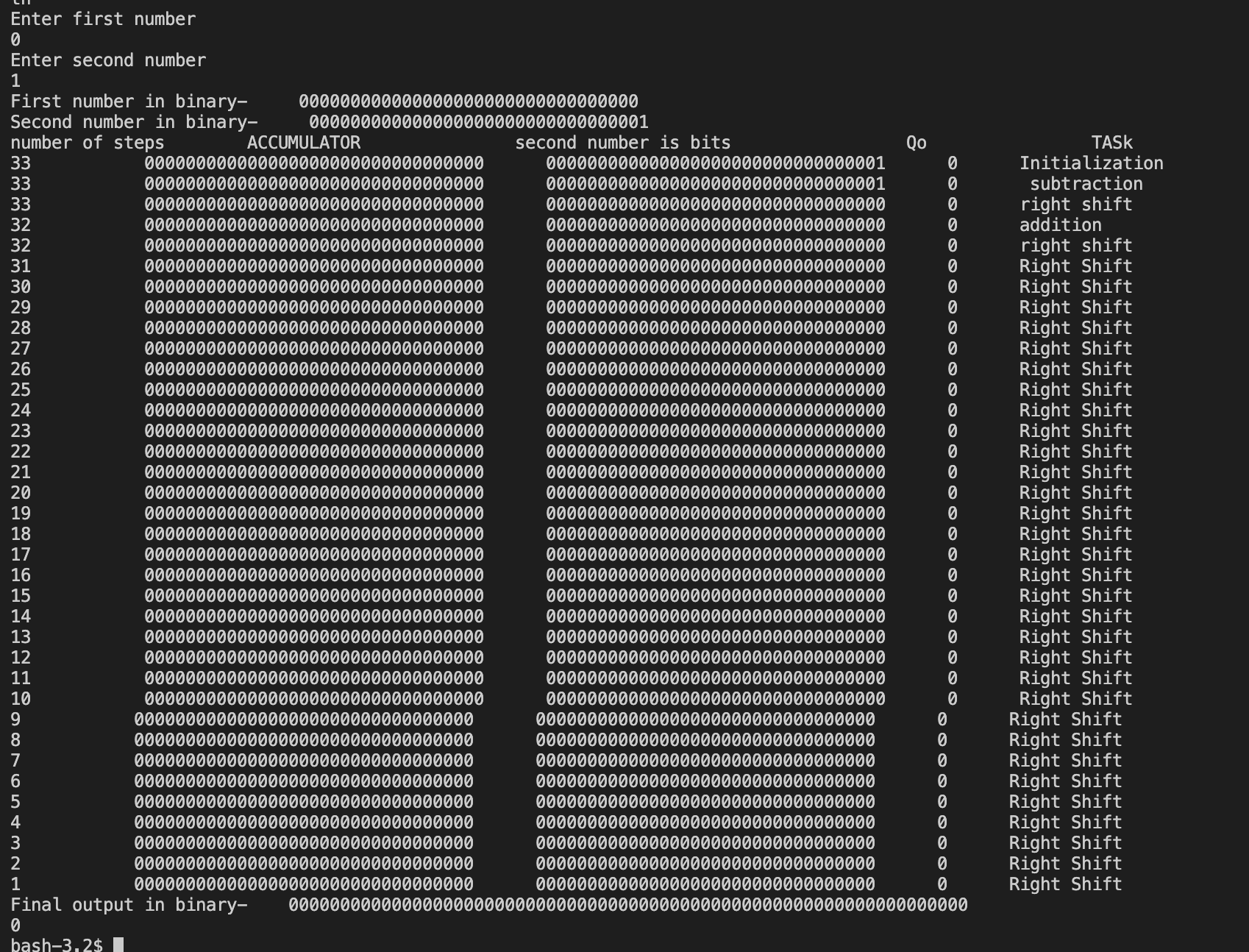
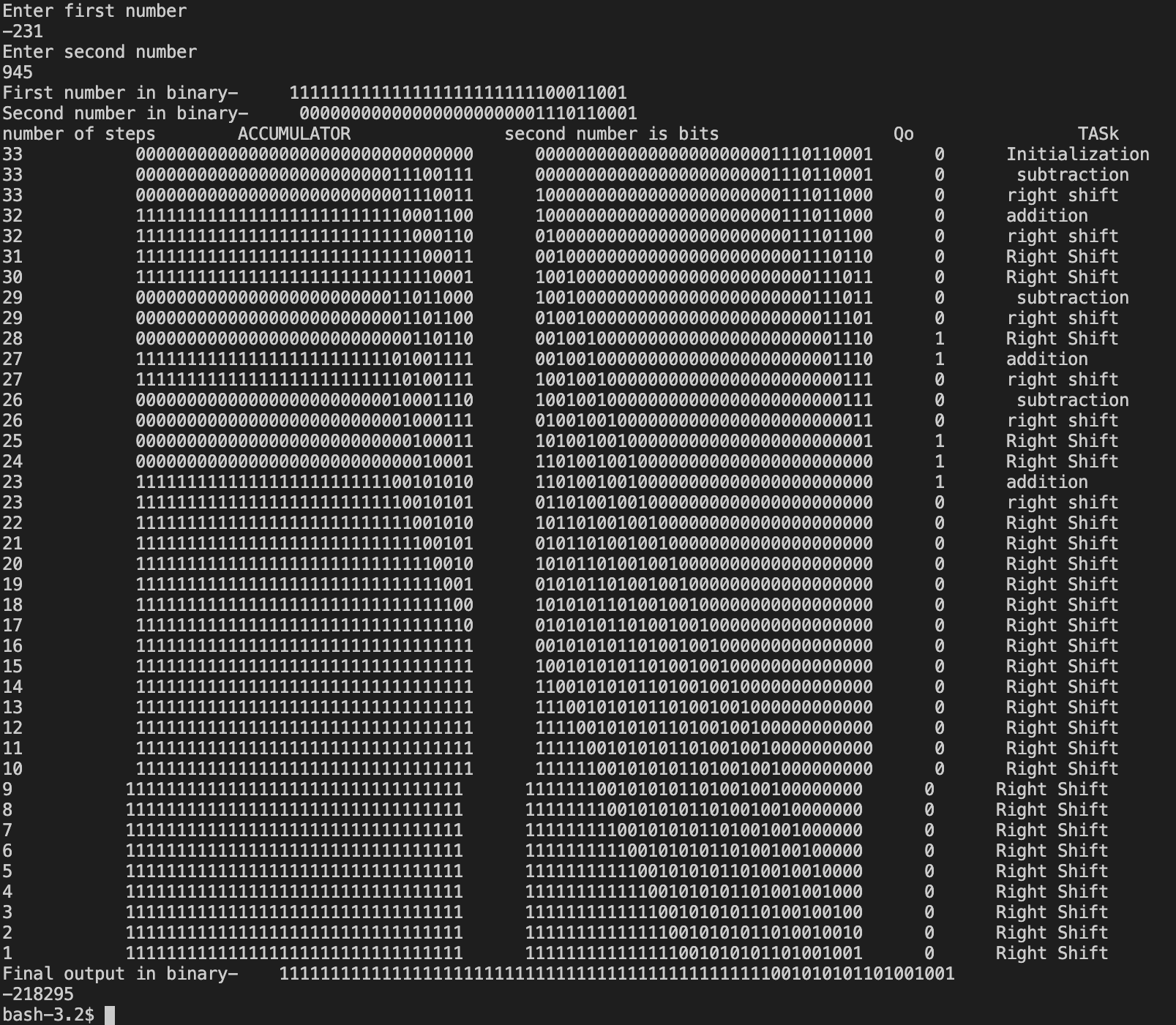
SCREENSHOTS OF OUTPUTS-

1.First number negative and second positive.

2.Both numbers are negative.

3.When one number is 0.

4.When both numbers are positive.



THANKYOU