PRINT FIRST NON-REPEATING CHARACTER IN THE STRING

Given a String where characters are repeating. Write code to print the first character that is repeating later in the string. e.g,

Input String: ritambhara

Output: i

Input String: social ritambhara

Output: s

Solution:

There are multiple questions that works on hashing and counting of characters in a string. See previous problems “Print character repeating maximum numberr of times in a string“, “Find first non-repeating character in the string“, “Find all repeating characters in a string“..

The solution for this problem is on similar lines..

Algorithm:

1. Keep a count array which will store the number of times a character is repeating in the array.
2. Initially all the counts will be zero.
3. Traverse the string, for each character in the string, increment the corresponding count.
4. Traverse the string again, and return the first character for which count is 1.

Code:

```
1       /**
2        * Returns the first non-repeating character.
3        */
4       char printNonRepeating(char* str)
5       {
6           // Boundary check
7           if(str == NULL || str[0] == '\0')
8           {
9               printf("EMPTY STRING");
10              return NULL;
11          }
12
13          // Count Array. All elements are zeros
```

```
14        int cntArr[256] = {0};
15
16        // Populating the Count Array
17        for(int i = 0; str[i] != '\0';  i++)
18            cntArr[str[i]]++;
19
20        // Getting the index of Maximum count in the Array
21        int maxCntIdx = 0;
22        for(int i=0; str[i] != '\0';  i++)
23            if(cntArr[str[i]] == 1)
24                return str[i];
25
26        // If no non-repeating character
27        return NULL;
28    }
```

Time Complexity: O(n)

Extra Space: O(1).. in fact is it O(256) in this case because we are using the count array. But since that in independent of the size of input array, hence O(1).

Method-2: Brute force (without hashing)

If you have strict memory requirements and you cannot afford to have the count array. Then the brute force method is to check the entire string for each character and see if that character is present in the string or not.

Code:

```
1       char printNonRepeating(char* str)
2       {
3           // Boundary check
4           if(str == NULL || str[0] == '\0')
5           {
6               printf("EMPTY STRING");
7               return;
8           }
9
10          for(int i = 0; str[i] != '\0';  i++)
11          {
12              bool found = false;
13              for(int j=i+1; str[j] != '\0'; j++)
14              {
15                  if(str[i] == str[j])
16                  {
17                      found = true;
18                      break; // No need to check further.
19                  }
20              }
21              if(!found)
```

```
22              return str[i];
23          }
24          // No non-repeating character found
25          return NULL;
26      }
```

Time Complexity:

Worst Case: $O(n^2)$

Best Case: O(1). When the first character is repeating at second position itself.

Extra Space: O(1).

Source: http://www.ritambhara.in/print-first-non-repeating-character-in-the-string/