**1. Given two strings s and t, return true if t is an anagram of s, and false otherwise.**

**An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.**

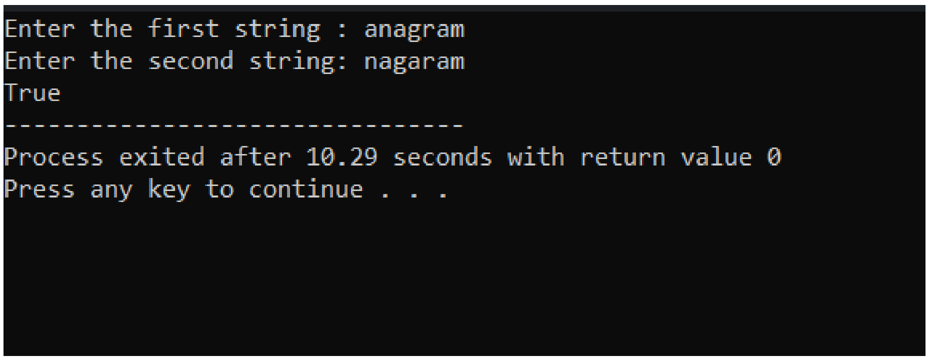
**Example 1:**

**Input: s = "anagram", t = "nagaram"**
**Output: true**
**Example 2:**

**Input: s = "rat", t = "car"**
**Output: false**

**CODE**

```
#include <stdio.h>
#include <string.h>

int areAnagrams(char str1[], char str2[])
 {
    int i;
    if (strlen(str1) != strlen(str2))
    {
        return 0;
    }
   int count[50] = {0};
   for (i = 0; str1[i] != '\0'; i++)
   {
        count[str1[i]]++;
    }
   for (i = 0; str2[i] != '\0'; i++)
   {
        count[str2[i]]--;
    }
   for (i = 0; i < strlen(str1); i++)
   {
        if (count[i] != 0)
        {
            return 0;
        }
    }

    return 1;
}

int main()
{

    printf("Enter the first string : ");
    char str1[50];
    scanf("%s", str1);
    printf("Enter the second string: ");
    char str2[50];
    scanf("%s", str2);

    if (areAnagrams(str1, str2))
    {
        printf("True");
    }
    else
    {
        printf("False");
    }

    return 0;
}
```



```
Enter the first string : anagram
Enter the second string: nagaram
True
--------------------------------
Process exited after 10.29 seconds with return value 0
Press any key to continue . . .
```

**2. Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string "".**

**Example 1:**

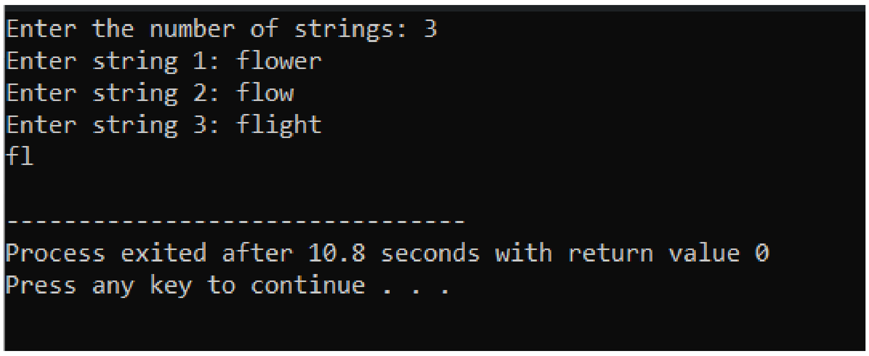
**Input: strs = ["flower","flow","flight"]**
**Output: "fl"**

**Example 2:**

**Input: strs = ["dog","racecar","car"]**
**Output: ""**
**Explanation: There is no common prefix among the input strings.**

**CODE**

```
#include <stdio.h>
#include <string.h>

void longestCommonPrefix(char strs[][100], int strsSize, char result[])
{
    if (strsSize == 0)
    {
        result[0] = '\0';
        return;
    }
    int i, j;
    for (i = 0; i < strlen(strs[0]); i++)
    {
        for (j = 1; j < strsSize; j++)
        {
            if (strs[j][i] != strs[0][i] || strs[j][i] == '\0')
            {
                strncpy(result, strs[0], i);
                result[i] = '\0';
                return;
            }
        }
    }
    strcpy(result, strs[0]);
}

int main()
{
    int size,i;
    printf("Enter the number of strings: ");
    scanf("%d", &size);

    char strs[size][100];

    for (i = 0; i < size; i++)
    {
        printf("Enter string %d: ", i + 1);
        scanf("%s", strs[i]);
    }

    char result[100];
    longestCommonPrefix(strs, size, result);
    printf("%s\n", result);

    return 0;
}
```



```
Enter the number of strings: 3
Enter string 1: flower
Enter string 2: flow
Enter string 3: flight
fl

--------------------------------
Process exited after 10.8 seconds with return value 0
Press any key to continue . . .
```

**3. Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.**

**A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.**

**Example 1:**

**Input: digits = "23"**
**Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]**

**Example 2:**

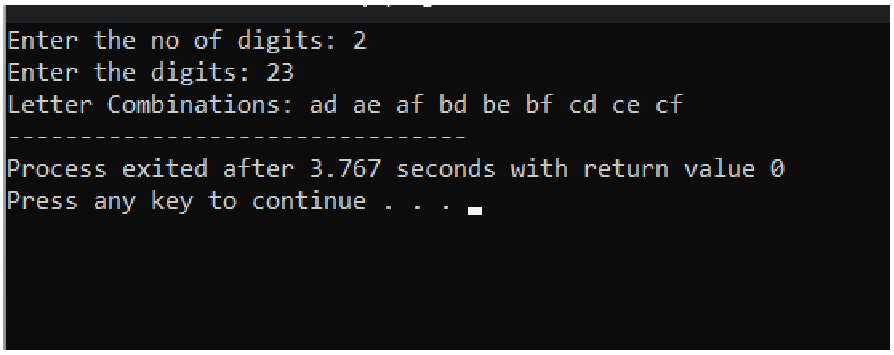
**Input: digits = ""**
**Output: []**

**Example 3:**

**Input: digits = "2"**
**Output: ["a","b","c"]**

**CODE**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

const char digit_mapping[10][5] = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};

void generateCombinations(char digits[], int index, char currentCombination[], char result[][5], int
*resultSize)
{
        int i;
  if (index == strlen(digits)) {
    strcpy(result[*resultSize], currentCombination);
    (*resultSize)++;
    return;
  }

  const char *letters = digit_mapping[digits[index] - '0'];

  for (i = 0; i < strlen(letters); i++)
        {
    currentCombination[index] = letters[i];
    generateCombinations(digits, index + 1, currentCombination, result, resultSize);
  }
}

void letterCombinations(char digits[], char result[][5], int *returnSize)
{
  int i;
  if (digits == NULL || strlen(digits) == 0) {
    *returnSize = 0;
    return;
  }

  int maxCombinations = 1;
  for (i = 0; i < strlen(digits); i++)
 {
    maxCombinations *= strlen(digit_mapping[digits[i] - '0']);
  }

  *returnSize = 0;

  char currentCombination[5] = "";

  generateCombinations(digits, 0, currentCombination, result, returnSize);
}

int main() {
  int n,i;
  printf("Enter the no of digits: ");
  scanf("%d", &n);

  char digits[n + 1];  // +1 to account for the null terminator
  printf("Enter the digits: ");
  scanf("%s", digits);

  char result[100][5];
  int returnSize;

  for (i = 0; i < 100; i++)
 {
    strcpy(result[i], "");
  }

  letterCombinations(digits, result, &returnSize);

  printf("Letter Combinations: ");
  for (i = 0; i < returnSize; i++)
  {
    printf("%s ", result[i]);
  }

  return 0;
}
```



```
Enter the no of digits: 2
Enter the digits: 23
Letter Combinations: ad ae af bd be bf cd ce cf
--------------------------------
Process exited after 3.767 seconds with return value 0
Press any key to continue . . .
```