DATE:23.12.23 QUIZ-2

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..

CODE:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int compareChars(const void *a, const void *b) {
   return (*(char *)a - *(char *)b);
}
int isAnagram(char *s, char *t) {
   int len_s = strlen(s);
   int len_t = strlen(t)
   if (len_s != len_t)
      return 0;
   qsort(s, len_s, sizeof(char), compareChars);
   qsort(t, len_t, sizeof(char), compareChars);
   return (strcmp(s, t) == 0);
}

int main() {

   char s1[] = "anagram";
   char t1[] = "nagaram";
   printf("%s\n", isAnagram(s1, t1) ? "true" : "false");

   return 0;
}
```

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

Example 1:

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

PROGRAM CODE:

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

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

    return strdup(strs[0]);
}

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

    char** strs = (char**)malloc(n * sizeof(char*));
    for (int i = 0; i < n; ++i) {
        printf("Enter string %d: ", i + 1);
        char buffer[256];  // Assuming a maximum string length of 255
        scanf("%s", buffer);
        strs[i] = strdup(buffer);
    }
    char* result = longestCommonPrefix(strs, n);
    printf("Longest Common Prefix: %s\n", result);
 free(result);
    for (int i = 0; i < n; ++i) {
        free(strs[i]);
    }
    free(strs);

    return 0;
}
```

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.

PROGRAM CODE:

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

char* mapping[] = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};

void generateCombinations(char* digits, int index, char* current, char** result, int* resultSize) {
    if (index == strlen(digits)) {
        result[*resultSize] = strdup(current);
        (*resultSize)++;
        return;
    }

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

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

char** letterCombinations(char* digits, int* returnSize) {
    if (digits == NULL || strlen(digits) == 0) {
        *returnSize = 0;
        return NULL;
    }

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

    char** result = (char**)malloc(maxCombinations * sizeof(char*));
    *returnSize = 0;

    char* current = (char*)malloc(strlen(digits) + 1);

    generateCombinations(digits, 0, current, result, returnSize);

    free(current);

    return result;
}

int main() {
    printf("Enter the digits: ");
    char digits[256];  // Assuming a maximum length of 255 for digits
    scanf("%s", digits);

    int returnSize;
```

```
    char** result = letterCombinations(digits, &returnSize);

    printf("[");
    for (int i = 0; i < returnSize; ++i) {
        printf("\"%s\"", result[i]);
        if (i < returnSize - 1) {
            printf(", ");
        }
        free(result[i]);
    }
    printf("]\n");

    free(result);

    return 0;
}
```