**STRINGS**

**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 <stdbool.h>

#include <string.h>

bool isAnagram(char \*s, char \*t) {

if (strlen(s) != strlen(t)) {

return false;

}

int count[26] = {0};

for (int i = 0; s[i] != '\0'; i++) {

count[s[i] - 'a']++;

}

for (int i = 0; t[i] != '\0'; i++) {

count[t[i] - 'a']--;

}

for (int i = 0; i < 26; i++) {

if (count[i] != 0) {

return false;

}

}

return true;

}

int main() {

char s[100], t[100];

printf("Enter the first string: ");

scanf("%s", s);

printf("Enter the second string: ");

scanf("%s", t);

if (isAnagram(s, t)) {

printf("The strings are anagrams.\n");

} else {

printf("The strings are not anagrams.\n");

}

return 0;

}

**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 <stdlib.h>

#include <string.h>

char\* longestCommonPrefix(char\*\* strs, int strsSize) {

if (strsSize == 0) {

char\* result = (char\*)malloc(1);

result[0] = '\0';

return result;

}

int minLen = strlen(strs[0]);

for (int i = 1; i < strsSize; i++) {

int len = strlen(strs[i]);

if (len < minLen) {

minLen = len;

}

}

char\* result = (char\*)malloc(minLen + 1);

if (result == NULL) {

fprintf(stderr, "Memory allocation failed\n");

exit(1);

}

for (int i = 0; i < minLen; i++) {

char currentChar = strs[0][i];

for (int j = 1; j < strsSize; j++) {

if (strs[j][i] != currentChar) {

result[i] = '\0';

return result;

}

}

result[i] = currentChar;

}

result[minLen] = '\0';

return result;

}

int main() {

int strsSize;

printf("Enter the number of strings: ");

scanf("%d", &strsSize);

char\*\* strs = (char\*\*)malloc(strsSize \* sizeof(char\*));

if (strs == NULL) {

fprintf(stderr, "Memory allocation failed\n");

exit(1);

}

for (int i = 0; i < strsSize; i++) {

printf("Enter string %d: ", i + 1);

char buffer[100];

scanf("%s", buffer);

strs[i] = (char\*)malloc(strlen(buffer) + 1);

if (strs[i] == NULL) {

fprintf(stderr, "Memory allocation failed\n");

exit(1);

}

strcpy(strs[i], buffer);

}

char\* result = longestCommonPrefix(strs, strsSize);

printf("Longest Common Prefix: %s\n", result);

for (int i = 0; i < strsSize; i++) {

free(strs[i]);

}

free(strs);

free(result);

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.

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>

void generateCombinations(char\* digits, char\* mapping[], int index, char\* current, char\*\* result, int\* resultSize) {

if (index == strlen(digits)) {

current[index] = '\0';

result[\*resultSize] = strdup(current);

(\*resultSize)++;

return;

}

char\* letters = mapping[digits[index] - '2'];

for (int i = 0; i < strlen(letters); i++) {

current[index] = letters[i];

generateCombinations(digits, mapping, index + 1, current, result, resultSize);

}

}

char\*\* letterCombinations(char\* digits, int\* returnSize) {

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

int totalCombinations = 1;

for (int i = 0; i < strlen(digits); i++) {

totalCombinations \*= strlen(mapping[digits[i] - '2']);

}

char\*\* result = (char\*\*)malloc(sizeof(char\*) \* totalCombinations);

\*returnSize = 0;

if (strlen(digits) > 0) {

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

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

free(current);

}

return result;

}

int main() {

char digits[20];

printf("Enter digits (2-9): ");

scanf("%s", digits);

int returnSize;

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

printf("Output: [");

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;

}