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

#include <stdio.h>

int is\_anagram(char a[], char b[]) {

int i = 0;

int freq\_a[26] = {0};

int freq\_b[26] = {0};

// converting uppercase to lowercase and updating frequency

while (a[i] != '\0') {

if (a[i] >= 'A' && a[i] <= 'Z') {

a[i] = 'a' + (a[i] - 'A');

}

freq\_a[a[i] - 'a']++;

i++;

}

// converting uppercase to lowercase and updating frequency

i = 0;

while (b[i] != '\0') {

if (b[i] >= 'A' && b[i] <= 'Z') {

b[i] = 'a' + (b[i] - 'A');

}

freq\_b[b[i] - 'a']++;

i++;

}

// if the frequency of all 26 characters is same for both strings it returns 1 else returns 0.

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

if (freq\_a[i] != freq\_b[i]) {

return 0;

}

}

return 1;

}

int main() {

char a[50], b[50];

printf("Enter first string : ");

scanf("%s", a);

printf("Enter second string : ");

scanf("%s", b);

if (is\_anagram(a, b) == 1) {

printf("True");

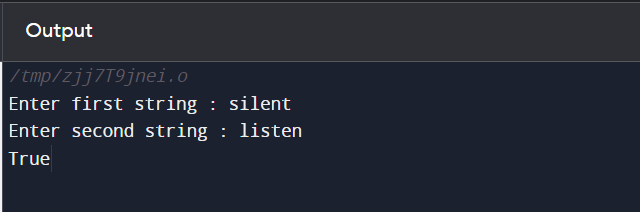
} else {

printf("False");

}

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 "".**

#include <stdio.h>

#include <string.h>

// Function to find the longest common prefix

char\* longestCommonPrefix(char strs[][50], int strsSize) {

if (strsSize == 0) {

static char empty[] = "";

return empty;

}

char \*prefix = strs[0];

int len = strlen(prefix);

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

int j = 0;

while (prefix[j] && strs[i][j] && prefix[j] == strs[i][j]) {

j++;

}

len = (len < j) ? len : j;

prefix[j] = '\0';

}

if (len == 0) {

static char empty[] = "";

return empty;

}

prefix[len] = '\0';

return prefix;

}

int main() {

char strings[100][50];

int numStrings;

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

scanf("%d", &numStrings);

printf("Enter %d strings:\n", numStrings);

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

scanf("%s", strings[i]);

}

char \*result = longestCommonPrefix(strings, numStrings);

if (strcmp(result, "") == 0) {

printf("''''\n");

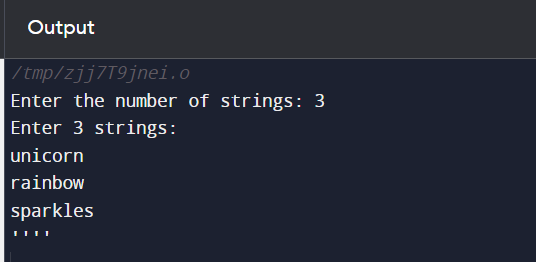
} else {

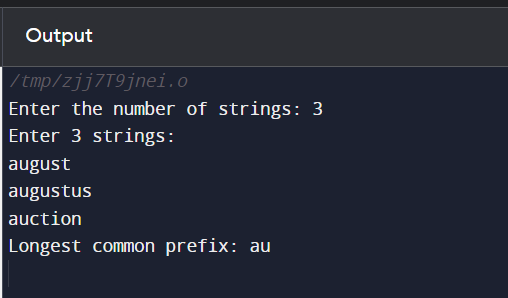
printf("Longest common prefix: %s\n", result);

}

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.**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

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

void generateCombinations(char \*digits, int index, char \*current, char \*\*result, int \*count) {

if (index == strlen(digits)) {

result[\*count] = (char \*)malloc(strlen(current) + 1);

strcpy(result[\*count], current);

(\*count)++;

return;

}

int digit = digits[index] - '0';

int len = strlen(mapping[digit]);

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

current[index] = mapping[digit][i];

generateCombinations(digits, index + 1, current, result, count);

}

}

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

if (digits == NULL || strlen(digits) == 0) {

\*returnSize = 0;

return NULL;

}

int totalCombinations = 1;

int digitsLen = strlen(digits);

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

int digit = digits[i] - '0';

totalCombinations \*= strlen(mapping[digit]);

}

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

int count = 0;

char \*current = (char \*)malloc(digitsLen + 1);

current[digitsLen] = '\0';

generateCombinations(digits, 0, current, result, &count);

\*returnSize = count;

free(current);

return result;

}

int main() {

char digits[100];

printf("Enter a string of digits: ");

fgets(digits, sizeof(digits), stdin);

digits[strcspn(digits, "\n")] = '\0';

int returnSize;

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

printf("Output: [");

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

printf("\"%s\"", combinations[i]);

if (i != returnSize - 1) {

printf(",");

}

free(combinations[i]);

}

printf("]\n");

free(combinations);

return 0;

}

