1.#include <stdio.h>

#include <stdbool.h>

#include <string.h>

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

int count[26] = {0}; // Assuming only lowercase English letters

// Increment count for characters in string s

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

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

}

// Decrement count for characters in string t

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

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

}

// Check if all counts are zero

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

if (count[i] != 0) {

return false;

}

}

return true;

}

int main() {

char s[100], t[100];

// Input from the user

printf("Enter the first string: ");

scanf("%s", s);

printf("Enter the second string: ");

scanf("%s", t);

// Check if strings are anagrams and print the result

if (isAnagram(s, t)) {

printf("Output: true\n");

} else {

printf("Output: false\n");

}

return 0;

}

2. #include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Function to find the longest common prefix

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

if (strsSize == 0) {

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

result[0] = '\0';

return result;

}

int prefixLen = strlen(strs[0]);

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

int j = 0;

while (j < prefixLen && strs[0][j] == strs[i][j]) {

j++;

}

prefixLen = j;

}

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

strncpy(result, strs[0], prefixLen);

result[prefixLen] = '\0';

return result;

}

int main() {

// Example 1

char\* strs1[] = {"flower", "flow", "flight"};

char\* result1 = longestCommonPrefix(strs1, 3);

printf("Example 1: %s\n", result1);

free(result1);

// Example 2

char\* strs2[] = {"dog", "racecar", "car"};

char\* result2 = longestCommonPrefix(strs2, 3);

printf("Example 2: %s\n", result2);

free(result2);

return 0;

}

3. #include <stdio.h>

#include <stdlib.h>

#include <string.h>

const char\* digitMapping[] = {

"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"

};

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

if (index == strlen(digits)) {

current[index] = '\0';

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

(\*resultIndex)++;

return;

}

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

const char\* letters = digitMapping[digit];

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

current[index] = letters[i];

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

}

}

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

int n = strlen(digits);

if (n == 0) {

\*returnSize = 0;

return NULL;

}

int totalCombinations = 1;

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

totalCombinations \*= strlen(digitMapping[digits[i] - '0']);

}

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

\*returnSize = 0;

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

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

free(current);

return result;

}

int main() {

char digits[50]; // Assuming a reasonable maximum length for user input

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

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;

}