## Day 14: String Handling - Part 1

### 1. Write a program to read and display a string.

#include <stdio.h>  
  
int main() {  
 char str[100]; // Declare a character array (string)  
  
 printf("Enter a string: ");  
 // Using fgets to read string with spaces  
 // sizeof(str) ensures buffer overflow prevention  
 // stdin specifies to read from standard input  
 fgets(str, sizeof(str), stdin);  
  
 printf("You entered: %s", str); // %s is format specifier for string  
 return 0;  
}

### 2. Write a program to calculate the length of a string without using strlen().

#include <stdio.h>  
  
int main() {  
 char str[100];  
 int length = 0;  
  
 printf("Enter a string: ");  
 fgets(str, sizeof(str), stdin);  
  
 // Iterate through the string until the null terminator is found  
 while (str[length] != '\0' && str[length] != '\n') {  
 length++;  
 }  
  
 printf("Length of the string: %d\n", length);  
 return 0;  
}

### 3. Write a program to reverse a string.

#include <stdio.h>  
#include <string.h> // Required for strlen() if used, though we'll do it manually  
  
int main() {  
 char str[100];  
 char reversedStr[100];  
 int length = 0;  
 int i, j;  
  
 printf("Enter a string: ");  
 fgets(str, sizeof(str), stdin);  
  
 // Remove the newline character if fgets added it  
 str[strcspn(str, "\n")] = 0;  
  
 // Calculate length manually  
 while (str[length] != '\0') {  
 length++;  
 }  
  
 // Reverse the string  
 j = 0;  
 for (i = length - 1; i >= 0; i--) {  
 reversedStr[j] = str[i];  
 j++;  
 }  
 reversedStr[j] = '\0'; // Add null terminator to the reversed string  
  
 printf("Original string: %s\n", str);  
 printf("Reversed string: %s\n", reversedStr);  
 return 0;  
}

### 4. Write a program to convert a string to uppercase.

#include <stdio.h>  
#include <ctype.h> // Required for toupper()  
  
int main() {  
 char str[100];  
 int i = 0;  
  
 printf("Enter a string: ");  
 fgets(str, sizeof(str), stdin);  
  
 printf("Original string: %s", str);  
  
 // Convert to uppercase  
 while (str[i] != '\0') {  
 str[i] = toupper(str[i]);  
 i++;  
 }  
  
 printf("Uppercase string: %s", str);  
 return 0;  
}

### 5. Write a program to convert a string to lowercase.

#include <stdio.h>  
#include <ctype.h> // Required for tolower()  
  
int main() {  
 char str[100];  
 int i = 0;  
  
 printf("Enter a string: ");  
 fgets(str, sizeof(str), stdin);  
  
 printf("Original string: %s", str);  
  
 // Convert to lowercase  
 while (str[i] != '\0') {  
 str[i] = tolower(str[i]);  
 i++;  
 }  
  
 printf("Lowercase string: %s", str);  
 return 0;  
}

### 6. Write a program to check if a string is a palindrome.

#include <stdio.h>  
#include <string.h> // Required for strlen() and strcmp()  
  
int main() {  
 char str[100];  
 char reversedStr[100];  
 int i, j, length = 0;  
 int isPalindrome = 1; // Flag to check if it's a palindrome  
  
 printf("Enter a string: ");  
 fgets(str, sizeof(str), stdin);  
  
 // Remove the newline character if fgets added it  
 str[strcspn(str, "\n")] = 0;  
  
 // Calculate length  
 while (str[length] != '\0') {  
 length++;  
 }  
  
 // Reverse the string  
 j = 0;  
 for (i = length - 1; i >= 0; i--) {  
 reversedStr[j] = str[i];  
 j++;  
 }  
 reversedStr[j] = '\0'; // Add null terminator  
  
 // Compare original and reversed strings  
 for (i = 0; i < length; i++) {  
 if (str[i] != reversedStr[i]) {  
 isPalindrome = 0;  
 break;  
 }  
 }  
  
 if (isPalindrome) {  
 printf("The string \"%s\" is a palindrome.\n", str);  
 } else {  
 printf("The string \"%s\" is not a palindrome.\n", str);  
 }  
  
 return 0;  
}

### 7. Write a program to count vowels and consonants in a string.

#include <stdio.h>  
#include <ctype.h> // Required for tolower()  
  
int main() {  
 char str[100];  
 int i, vowels = 0, consonants = 0;  
  
 printf("Enter a string: ");  
 fgets(str, sizeof(str), stdin);  
  
 // Remove the newline character if fgets added it  
 str[strcspn(str, "\n")] = 0;  
  
 for (i = 0; str[i] != '\0'; ++i) {  
 char ch = tolower(str[i]); // Convert to lowercase for easier comparison  
  
 if (ch >= 'a' && ch <= 'z') { // Check if it's an alphabet  
 if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {  
 ++vowels;  
 } else {  
 ++consonants;  
 }  
 }  
 }  
  
 printf("Number of vowels: %d\n", vowels);  
 printf("Number of consonants: %d\n", consonants);  
 return 0;  
}

### 8. Write a program to count words in a string.

#include <stdio.h>  
#include <string.h> // Required for strlen()  
  
int main() {  
 char str[200];  
 int i, words = 0;  
 int inWord = 0; // Flag to indicate if we are currently inside a word  
  
 printf("Enter a string: ");  
 fgets(str, sizeof(str), stdin);  
  
 // Remove the newline character if fgets added it  
 str[strcspn(str, "\n")] = 0;  
  
 for (i = 0; str[i] != '\0'; ++i) {  
 // If the current character is a space, tab, or newline, it's a word separator  
 if (str[i] == ' ' || str[i] == '\t' || str[i] == '\n') {  
 inWord = 0; // We are no longer in a word  
 }  
 // If the current character is not a separator and we were not in a word, it's the start of a new word  
 else if (inWord == 0) {  
 inWord = 1; // We are now in a word  
 words++; // Increment word count  
 }  
 }  
  
 printf("Number of words: %d\n", words);  
 return 0;  
}

### 9. Write a program to find the frequency of a character in a string.

#include <stdio.h>  
#include <string.h>  
  
int main() {  
 char str[100];  
 char ch;  
 int count = 0;  
 int i;  
  
 printf("Enter a string: ");  
 fgets(str, sizeof(str), stdin);  
  
 // Remove the newline character if fgets added it  
 str[strcspn(str, "\n")] = 0;  
  
 printf("Enter a character to find its frequency: ");  
 scanf(" %c", &ch); // Space before %c to consume any leftover newline character  
  
 for (i = 0; str[i] != '\0'; ++i) {  
 if (ch == str[i]) {  
 ++count;  
 }  
 }  
  
 printf("Frequency of %c = %d\n", ch, count);  
 return 0;  
}

### 10. Write a program to copy one string to another without using strcpy().

#include <stdio.h>  
  
int main() {  
 char source[100], destination[100];  
 int i = 0;  
  
 printf("Enter the source string: ");  
 fgets(source, sizeof(source), stdin);  
  
 // Remove the newline character if fgets added it  
 source[strcspn(source, "\n")] = 0;  
  
 // Copy characters one by one  
 while (source[i] != '\0') {  
 destination[i] = source[i];  
 i++;  
 }  
 destination[i] = '\0'; // Add null terminator to the destination string  
  
 printf("Source string: %s\n", source);  
 printf("Destination string: %s\n", destination);  
 return 0;  
}