Strings

In C, strings are essentially **arrays of characters** that are terminated by a **null character ('\0').**

Declaring Strings :

Array of Characters:

char str[20] = “SOA” ;

Char str[10] ={‘S’, ‘O’, ‘A’} ;

String Literal Initialization :

char str[ ] = "Hello, World!";

Common String Operations :

1. **String Length:** Use **`strlen()`** from the **`<string.h>` library** to find the length of a string (not counting the null terminator).

Example :

```
#include <stdio.h>

#include <string.h>

int main() {

   char str[ ] = "Hello, World!";

   printf("Length: %s", strlen(str));     Output: Length is 13

   return 0;

}
```

2. **String Copy:** Use `strcpy()` to copy one string to another.

Example :

```
char s[ ] = "Hello";

char d [20];

strcpy(d, s);   string s will copy to string d
```

3. **String Concatenation:** Use `strcat()` to concatenate two strings.

Example :

```
char str1[20] = "Hello, ";

char str2[ ] = "World!";

strcat(str1, str2);   str1 now contains "Hello, World!"
```

4. **String Comparison:** Use `strcmp()` to compare two strings.

Example :

```
if (strcmp(str1, str2) == 0) {

    printf("Strings are equal.\n");

} else {

    printf("Strings are not equal.\n");

}
```

Example :

Given the string **pres** (value is "**Adams, John Quincy**") and the 40-character temporary variables **tmp1 and tmp2**, what string is displayed by the following code fragment?

```
strcpy(tmp1, &pres[7], 4);

tmp1[4] = ’\0’;

strcat(tmp1, " ");

strcpy(tmp2, pres, 5);

tmp2[5] = ’\0’;

printf("%s\n", strcat(tmp1, tmp2));
```

```
pres = "Adams, John Quincy"
```

`tmp1` and `tmp2` are temporary character arrays with a size of 40 characters.

1. **`strncpy(tmp1, &pres[7], 4);`**

`&pres[7]` points to the character `'J'`

copies the next 4 characters starting from the J into `tmp1`.

`tmp1` will hold the string `"John"`.

2. **`tmp1[4] = '\0';`**

adds a null terminator at index 4 of `tmp1`. `tmp1` is now `"John\0"`.

3. **`strcat(tmp1, " ");`**

This concatenates a space " " to the end of tmp1. tmp1 becomes "John ".

4. **`strncpy(tmp2, pres, 5);`**

copies the first 5 characters of the `pres` string into `tmp2`.

`tmp2` will hold the string `"Adam"`.

5. **`tmp2[5] = '\0';`**

adds a null terminator at index 5 of `tmp2`. `tmp2` is `"Adam\0"`.

printf("%s\n", strcat(tmp1, tmp2));

concatenates `tmp2` to `tmp1`.

`tmp1` will contain `"John Adam"`.

Counting vowels and consonants in a sentence.

Input	Output
`Enter a sentence: Programming in C`	Vowels: 5 Consonants: 10

```
#include <stdio.h>
int main() {
    char s[20];
    int v = 0, c = 0;
    printf("Enter a sentence: ");
    gets(s);
    for (int i = 0; s[i] != '\0'; i++) {
        char ch = s[i];
        if (ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U' ||
            ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
            v++;
        }
         else if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')) {
            c++;
        }
    }
    printf("Vowels: %d\n", vowels);
    printf("Consonants: %d\n", consonants);
   return 0;
}
```

Converting the vowels in a sentence to uppercase.

Input	Output
`Enter a sentence: Hello World`	**Modified sentence: HEllo WOrld**

`‘a'` is 97 and `'A'` is 65.

`'e'` is 101 and `'E'` is 69.

`'i'` is 105 and `'I'` is 73.

`'o'` is 111 and `'O'` is 79.

`'u'` is 117 and `'U'` is 85.

To convert a lowercase letter to uppercase:

For example, the difference between `'a'` and `'A'` is `97 - 65 = 32`.

`sentence[i] - 'a' + 'A'`, essentially shifting the lowercase character to its uppercase equivalent:

If `sentence[i]` is `'a'` (ASCII 97), then `sentence[i] - 'a'` evaluates to `0`

`Adding 'A' (ASCII 65) to this result gives 65, which is 'A'.`

`For 'e', sentence[i] - 'a' + 'A' calculates as`

`101 - 97 + 65 = 69, which is 'E'.`

```
#include <stdio.h>

int main() {

    char s[20];

    printf("Enter a sentence: ");

   gets(s);

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

         if (s[i] == 'a' || s[i] == 'e' || s[i] == 'i' ||

            s[i] == 'o' || s[i] == 'u') {

            s[i] = s[i] - 'a' + 'A';

        }

    }

    printf("Modified sentence: %s", s);

    return 0;

}
```

Displaying the count of each character in a string.

Input	Output
Enter a string: hello world	Character counts: 'h' : 1 'e' : 1 'l' : 3 'o' : 2 ' ' : 1 'w' : 1 'r' : 1 'd' : 1

Declare one Array to **store the frequency of each ASCII character (from 0 to 255). This array uses the ASCII value of each character as an index.**

```
 int count[256] = {0};

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

        if (str[i] != '\n') {

            count[(int)str[i]]++;

        }

    }
```

`str[i]` is cast to an integer, which gives the ASCII value of the character. For example, if `str[i]` is `'A'`, its ASCII value is `65`, and `count[65]` is incremented.

```
#include <stdio.h>

#include <string.h>

int main() {

    char str[100];

    int count[256] = {0};

    printf("Enter a string: ");

    gets(str);

    int length = strlen(str);

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

        if (str[i] != '\n') {

            count[(int)str[i]]++;

        }

    }

    printf("Character counts:\n");

    for (int i = 0; i < 256; i++) {

        if (count[i] > 0) {

            printf("'%c' : %d\n", i, count[i]);

        }

    }

return 0;

}
```

Finding the occurrence of the first repetitive character in a string.

Input	Output
`Enter a string: Hello World`	First repetitive character: 'l'

```
#include <stdio.h>

#include <string.h>

int main() {

    char str[20];

   int a[256] = {0};

    printf("Enter a string: ");

    gets(str);

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

        if (a[(int)str[i]] > 0) {

                printf("First repetitive character: '%c'\n", str[i]);

               return 0;

        }

        a[(int)str[i]]++;

    }

    printf("No repetitive characters found.\n");

    return 0;

}
```

```
#include <stdio.h>

#include <string.h>

int main() {

    char str[100];

    printf("Enter a string: ");
    gets(str);

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

        for (int j = i + 1; str[j] != '\0'; j++) {

         if (str[i] == str[j]) {

            printf("First repetitive character: '%c'\n", str[i]);

            return 0;

         }

       }

    }

    printf("No repetitive characters found.\n");

    return 0;

}
```

Create a function **`reverseInputWords`** that is designed to read words recursively and print them in reverse order.

Input	Output

`Enter the number of words: 3`
`Enter 3 words:`
`hello world coding`
	`ollehd dlrow gnidoc`

```c
#include <stdio.h>

#include <string.h>

void reverseInputWords(int n);

int main() {

    int n;

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

    scanf("%d", &n);

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

    reverseInputWords(n);

    return 0;

}

void reverseInputWords(int n) {

    char word[50];

     if (n == 0) {

        return;

    }

   scanf("%s", word);

    reverseInputWords(n - 1);

    int len = strlen(word);

    for (int i = len - 1; i >= 0; i--) {

        printf("%c", word[i]);

    }

    printf(" ");

}
```

Numbers of words in a sentence.

Input	Output
`Enter the sentence: Hello World`	`Word Count : 2`

```
#include <stdio.h>

int main() {

    char sentence[100];

    int count = 0;

    int isWord = 0;

    printf("Enter a sentence: ");

    scanf("%s",sentence);

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

        if (sentence[i] == ' ' || sentence[i] == '\t' || sentence[i] == '\n') {

            isWord = 0;

        }

          else if ((sentence[i] >= 'a' && sentence[i] <= 'z') || (sentence[i]
>= 'A' && sentence[i] <= 'Z') || (sentence[i] >= '0' && sentence[i] <=
'9')) {

            if (isWord == 0) {

                count++;

            isWord = 1;

            }

        }

    }

printf("Number of words in the sentence: %d\n", count);

 return 0;

}
```

Determining whether the string is a palindrome.

Input	Output
`Enter a string:`	

Assume you have two strings. Design a program to compare the two strings for equal or not without using any string library functions. Also Implement concatenate two strings without library functions.

Input	Output

`Enter a string: aabbcc`
`Enter a string: acbdef`
	Strings are not equal