WEEK-END ASSIGNMENT-10
Strings in C
Operating Systems Workshop (CSE 3541)

Problem Statement:

Experiment with character arrays, strings and operations on strings.

Assignment Objectives:

To understand how a string constant is stored in an array of characters. To learn about the placeholder %s and how it is used in **`printf`** and **`scanf`** operations. To learn operations performed on strings and also operations that can be performed on individual characters.

Instruction to Students (If any):

This assignment is designed to give practice with **strings, strings processing, and array of pointers in C**. Students are required to create their own programs to solve each and every question/problem as per the specification of the given question/problem to meet the basic requirement of systems programming. **Students are required to write the output/ paste the output screen shots onto their laboratory record after each question.**

Programming/ Output Based Questions:

1. We know a string in C is implemented as an array. So, declare and initialize the string **``It is very interesting''** and display the string.

   Code here▼

2. Declare and initialize the string using array reference and pointer.

0	1	2	3	4	5	6	7	8	9	10	11
I	T	E	R		S	O	A	\0	?	?	?

   Code here▼

3. Write a program to read a string from the keyboard and print each character with their address on the screen.

Code here▼

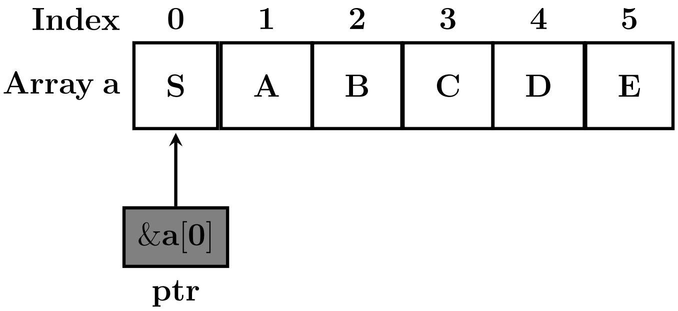
4. Declare and initialize the two arrays to hold the values as shown in the given rectangular boxes. Write the equivalent C statement to print their values and addresses using pointer.

0	1	2	3	4
I	B	C	S	\0

0	1	2	3	4
S	O	A	D	U

Code here▼

5. Write the C statement to declare and initialize the pointer variable for the given structure and display the array content using pointer.



Index	0	1	2	3	4	5
Array a	S	A	B	C	D	E

Code here▼

6. For the given declarations `int a[10]; int *pa;` and assignment `pa=a;`, select the legal /illegal statements from the followings

(a) `pa=a;`
(b) `pa=&a[0];`
(c) `pa++;`
(d) `a=pa`
(e) `a++`
(f) `is pa[i] identical to *(pa+i)?`
(g) `is &a[i] identical to (a+i)?`
(h) `is a[i] identical to *(a+i)?`
(i) `is pa[i] identical to a[i]?`
(j) `is void f(char str[]){...}identical to void f(char *str){...}?`
(k) `If a is an array, is f(&a[2]) identical to f(a+2);`

Code here▼

7. Let `p` be a pointer to an integer array and `n` is a scalar value. State the significance of the statement `p+n`.

Code here▼

8. If `arname` is an array, the function call `f(&arname[2]);` passes part of an array to the function by passing a pointer to the begining of the sub-array. Write an equivalent statement for the call `f(&arname[2]);`.

Code here▼

9. Write an equivalent statement for the function's formal parameter **`a`**, whose header/definition is given as **`f(int a[], int n, float y)`** {.......}.

Code here▼

10. Find the output of the following code segment for the function call **`bc=bytescount(``COVID-19 Still Active'')`**

```
int bytescount(char *s){
   char *p=s;
   while(*p!='\0'){
       p++;
   }
   return p-s;
}
```

Code here▼

11. Find the output of the following code segment for the function call **`cc=countchar(``Encourged to Vaccinate'')`**

```
int countchar(char *s){
   int n;
   for(n=0; *s!='\0'; s++){
       n++;
   }
   return n;
}
```

Code here▼

12. Identify the type of variable **amsg** and **pmsg** from the following declaration and initialization statements.

```
char pmsg[]="I am in 5th Sem CSE";
char *amsg="I am in 5th Sem CSE";
```

Code here▼

13. Find the output of the code fragment

```
char pmsg[60];
int nc;
nc=charcopy(pmsg,"I am in 5th Sem CSE");
printf("%d...%s\n",nc,pmsg);
```

The function definition/header is given as

```
int  charcopy(char *s, char *t)
{
   int i=0;
   while((s[i]=t[i])!=’\0’)
         i++;
   s[i]=’\0’;
   return(i);
}
```

Code here▼

14. The function header is given as;

```
int  charcopy(char *s, char *t){
  int i=0;
  while((*s=*t)!='\0')
  {
        s++;
        t++;
        i++;
  }
  *s='\0';
  return(i);
}
```

Compute the output of the following code segment

```
char pmsg[60];
int nc;
nc=charcopy(pmsg,"Studied in CSE");
printf("%d...%s\n",nc,pmsg);
```

Code here▼

15. The function header is given as;

```
int  charcopy(char *s, char *t){
   int i=0;
   while((*s++=*t++)!='\0'){
         i++;
   }
   *s='\0';
   return(i);
}
```

Compute the output of the following code segment

```
char pmsg[60];
int nc;
nc=charcopy(pmsg,"ITER CSE ");
printf("%d...%s\n",nc,pmsg);
```

Output here▼

16. Write a pointer version of string concatenation program using the user-defined function, **`stringconcate(s,t);`**, copies the string **`t`** to the end **`s`**.

Code here▼

```
#include <stdio.h>

// Custom implementation of string concatenation
void stringconcat(char *s, const char *t) {
    // Move the pointer to the end of the string s
    while (*s) {
        s++;
    }

    // Copy characters from t to the end of s
    while ((*s++ = *t++)) {
        // Empty loop body
    }
}
```

```
int main() {
    char str1[50] = "Hello, ";
    const char *str2 = "world!";

    printf("Before concatenation:\n");
    printf("str1: %s\n", str1);
    printf("str2: %s\n", str2);

    // Concatenate str2 to str1 using stringconcat
function
    stringconcat(str1, str2);

 printf("\nAfter concatenation:\n");
    printf("str1: %s\n", str1);

    return 0;
}
```

17. Write your own versions of the library functions **`strncpy`**, **`strncat`**, and **`strncmp`** which operate on at most the first **`n`** characters of their argument strings. For example **`strncpy(s,t,n)`** copies at most **`n`** characters of **`t`** to **`s`**.

Code here▼

```
#include <stdio.h>

// Custom implementation of strncpy
char *custom_strncpy(char *dest, const char *src, size_t n) {
    size_t i;
    for (i = 0; i < n && src[i] != '\0'; ++i) {
        dest[i] = src[i];
    }
    for (; i < n; ++i) {
        dest[i] = '\0';
    }
    return dest;
}
```

Code here▼

```
// Custom implementation of strncat
char *custom_strncat(char *dest, const char *src, size_t n) {
    size_t dest_len = strlen(dest);
    size_t i;
    for (i = 0; i < n && src[i] != '\0'; ++i) {
        dest[dest_len + i] = src[i];
    }
    dest[dest_len + i] = '\0';
    return dest;
}

// Custom implementation of strncmp
int custom_strncmp(const char *s1, const char *s2, size_t n) {
    for (size_t i = 0; i < n; ++i) {
        if (s1[i] != s2[i]) {
            return (unsigned char)s1[i] - (unsigned char)s2[i];
        }
        if (s1[i] == '\0') {
            return 0; // Reached the end of both strings
        }
    }
    return 0; // First n characters are equal
}

int main() {
    // Testing custom_strncpy
    char dest1[20] = "Hello";
    const char *src1 = "World";
    custom_strncpy(dest1, src1, 3);
    printf("custom_strncpy: %s\n", dest1);

    // Testing custom_strncat
    char dest2[20] = "Hello";
    const char *src2 = "World";
    custom_strncat(dest2, src2, 3);
    printf("custom_strncat: %s\n", dest2);

    // Testing custom_strncmp
    const char *s1 = "Hello";
    const char *s2 = "Hella";
    int result = custom_strncmp(s1, s2, 4);
    printf("custom_strncmp: %d\n", result);

    return 0;
}
```

18. Write a program to take a product code from Millies Mail-Order Catalog (MMOC) and separate it into its component parts. An MMOC product code begins with one or more letters identifying the warehouse where the product is stored. Next come the one or more digits that are the product ID. The final field of the string starts with a capital letter and represents qualifiers such as size, color, and so on. For example, ATL1203S14 stands for product 1203, size 14, in the Atlanta warehouse. Write a program that takes a code, finds the position of the first digit and of the first letter after the digits, and uses **`strcpy`** and **`strncpy`** to display a report such as the following:

```
Warehouse: ATL
Product: 1203
Qualifiers: S14
```

Code here▼

Code here▼

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {
    char productCode[50];

    // Input the MMOC product code
    printf("Enter MMOC product code: ");
    scanf("%s", productCode);

    // Find the position of the first digit
    int digitPosition = 0;
    while (productCode[digitPosition] && !isdigit(productCode[digitPosition])) {
        digitPosition++;
    }

    // Find the position of the first letter after the digits
    int letterPosition = digitPosition;
    while (productCode[letterPosition] && isdigit(productCode[letterPosition])) {
        letterPosition++;
    }

    // Use strcpy and strncpy to separate and display the components
    char warehouse[10], productID[10], qualifiers[30];

    strncpy(warehouse, productCode, digitPosition);
    warehouse[digitPosition] = '\0';

    strcpy(productID, productCode + digitPosition);

    strncpy(qualifiers, productCode + letterPosition, sizeof(qualifiers) - 1);
    qualifiers[sizeof(qualifiers) - 1] = '\0';

    // Display the report
    printf("Warehouse: %s\n", warehouse);
    printf("Product: %s\n", productID);
    printf("Qualifiers: %s\n", qualifiers);

    return 0;
}
```

19. Complete function **`trim_blanks(...)`** whose purpose is to take a single string input parameter (**`to_trim`**) and return a copy of the string with leading and trailing blanks removed. Use **`strncpy`** in **`trim_blanks`**.

`a_string` (before)

		a		p	h	r	a	s	e					\0

`n_string` (after the call: `trim_blanks(n_string, a_string);`)

a		p	h	r	a	s	e	\0						

```
char * trim_blanks(char *trimmed,                         /* output */
                        const char *to_trim) /* input */
{
     /* Find subscript of first nonblank in to_trim */
     /* Find subscript of last nonblank in to_trim */
    /* Use strncpy to store trimmed string in trimmed */
}
```

Code here▼

20. Draw an array to show the output of the function, **`strcat(s1, s2)`**, used in the following code snippet and also write the output.

```
#define STRSIZ 20
char s1[STRSIZ]="Jupiter ", s2[STRSIZ]="Symphony";
printf("%d %d\n", strlen(s1), strlen(strcat(s1, s2)));
printf("%s\n", s1);
```

Code here▼

21. 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?

```
strncpy(tmp1, &pres[7], 4);
tmp1[4] = '\0';
strcat(tmp1, " ");
strncpy(tmp2, pres, 5);
tmp2[5] = '\0';
printf("%s\n", strcat(tmp1, tmp2));
```

Code here▼

22. Write a program to check a string is palindrome or not. For example, **madam** is a palindrome, **computer** is not a palindrome.

Code here▼

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int isPalindrome(char str[]) {
   int left = 0, right = strlen(str) - 1;

   while (left < right) {
      while (!isalnum(str[left])) left++;
      while (!isalnum(str[right])) right--;

      if (tolower(str[left++]) != tolower(str[right--])) return 0;
   }

   return 1;
}

int main() {
   char inputString[100];

   printf("Enter a string: ");
   fgets(inputString, sizeof(inputString), stdin);

   if (inputString[strlen(inputString) - 1] == '\n') {
      inputString[strlen(inputString) - 1] = '\0';
   }

   printf(isPalindrome(inputString) ? "Palindrome\n" : "Not a palindrome\n");

   return 0;
}
```

23. Write a program in C to input a string using **getchar()** function only (Do not use **scanf()** or **gets()** function) and then count the total number of alphabets, number of alphabets in uppercase, number of alphabets in lowercase, number of digits, number of punctuation symbols, and number of spaces using character library functions.

Sample Run:

```
Input a string: I'm 2 bz 4 now.
Total number of alphabets: 7
Number of uppercase alphabets: 1
Number of lowercase alphabets: 6
Number of digits: 2
Number of punctuation mark: 2
Number of spaces: 4
```

Code here▼

```
#include <stdio.h>
#include <ctype.h>

int main() {
    char c;
    int totalAlphabets = 0, uppercaseAlphabets = 0, lowercaseAlphabets = 0, digits = 0,
     punctuationMarks = 0, spaces = 0;

    printf("Input a string: ");

    // Read characters until Enter is pressed
    while ((c = getchar()) != '\n') {
        totalAlphabets++;

        if (isupper(c)) {
            uppercaseAlphabets++;
        } else if (islower(c)) {
            lowercaseAlphabets++;
        } else if (isdigit(c)) {
            digits++;
        } else if (ispunct(c)) {
            punctuationMarks++;
        } else if (isspace(c)) {
            spaces++;
        }
    }

    // Display the results
    printf("Total number of alphabets: %d\n", totalAlphabets);
    printf("Number of uppercase alphabets: %d\n", uppercaseAlphabets);
    printf("Number of lowercase alphabets: %d\n", lowercaseAlphabets);
    printf("Number of digits: %d\n", digits);
    printf("Number of punctuation marks: %d\n", punctuationMarks);
    printf("Number of spaces: %d\n", spaces);

    return 0;
}
```

24. Write a program in C to read N strings from user and then sort them using bubble sort.

Sample Run:

```
Input number of strings :3
Input 3 strings:
hello
world
fun
The sorted strings are:
fun
hello
world
```

Code here▼

Code here▼

```
#include <stdio.h>
#include <string.h>

void bubbleSort(char arr[][100], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (strcmp(arr[j], arr[j + 1]) > 0) {
                // Swap the strings
                char temp[100];
                strcpy(temp, arr[j]);
                strcpy(arr[j], arr[j + 1]);
                strcpy(arr[j + 1], temp);
            }
        }
    }
}

int main() {
    int n;

    // Input the number of strings
    printf("Input number of strings: ");
    scanf("%d", &n);

    char strings[n][100];

    // Input N strings
    printf("Input %d strings:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%s", strings[i]);
    }

    // Sort the strings
    bubbleSort(strings, n);

    // Display the sorted strings
    printf("The sorted strings are:\n");
    for (int i = 0; i < n; i++) {
        printf("%s\n", strings[i]);
    }

    return 0;
}
```

25. What is the value of **t1** after execution of these statements if the value of **t2** is **‘‘Merry Christmas’’**?

```
 strncpy(t1, &t2[3], 5);
 t1[5] = ’\0’;
```

Code here▼

26. What does this program fragment display?

```
char x[80] = "gorilla";
char y[80] = "giraffe";
strcpy(x, y);
printf("%s %s\n", x, y);
```

Code here▼

27. What does this program fragment display?

```
char x[80] = "gorilla";
char y[80] = "giraffe";
strcat(x, y);
printf("%s %s\n", x, y);
```

Code here▼