MADHAN SHANKAR (220701149)

EXP NO:1 DATE:

DEVELOP A SIMPLE C PROGRAM TO DEMONSTRATE A BASIC STRING OPERATIONS

Questions

1. Input and Output

- Question: Modify the program to take a string input from the user and display it in uppercase.
- Hint: Use the toupper function from <ctype.h> to convert characters to uppercase.

2. String Length

- Question: Write a C program to check if a given substring exists within a string without using the strstr() function. If the substring is found, print its starting index; otherwise, print "Substring not found."

3. String Comparison

- Question: Extend the program to compare two strings entered by the user and print whether they are the same.
- Hint: Use the strcmp function from <string.h> for comparison.

4. Remove Spaces

- Question: Write a program to remove all spaces from a string entered by the user.
- Hint: Use a loop to copy non-space characters to a new string.

5. Frequency of Characters

- Question: Modify the program to calculate the frequency of each character in the string.
- Hint: Use an array of size 256 to store the count of each ASCII character.

6. Concatenate Strings

- Question: Extend the program to concatenate two strings entered by the user.
- Hint: Use the strcat function from <string.h>.

7. Replace a Character

- Question: Write a program to replace all occurrences of a specific character in the string with another character.
- Hint: Traverse the string and replace the character conditionally in a loop.

AIM:

To write a C program that takes a string input from the user and converts all its characters to uppercase using the toupper() function from the library.

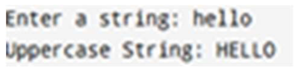
ALGORITHM:

1. Start
2. Declare a character array str to store the input string.
3. Prompt the user to enter a string.
4. Use fgets() to read the string input from the user.
5. Check if the last character is a newline (\n) and replace it with \0 (null terminator).
6. Loop through each character of the string:
7. Use toupper() to convert each character to uppercase.
8. Store the converted character back in the string.
9. Print the modified uppercase string. End

PROGRAM:

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

int main() {
   char str[100];

   printf("Enter a string: ");
scanf("%s", str);

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

   printf("Uppercase String: %s\n", str);
return 0;
}
```

OUTPUT:



```
Enter a string: hello
Uppercase String: HELLO
```

AIM:

To write a C program that checks whether a given substring exists within a string without using the strstr() function. If found, print its starting index; otherwise, print "Substring not found."

ALGORITHM:

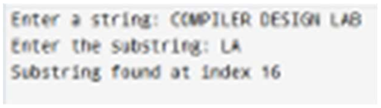
1. Start
2. Declare two character arrays: one for the main string and one for the substring.
3. Take input for both strings from the user.
4. Compute the lengths of both strings.
5. Loop through the main string and check for a match with the substring: o Compare characters one by one. o If a match is found, print the starting index and exit.
6. If no match is found, print "Substring not found."
7. End

PROGRAM:

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

int findSubstring(char str[], char substr[]) {
   int len1 = strlen(str);
   int len2 = strlen(substr);

   for (int i = 0; i <= len1 - len2; i++) {
int j;
       for (j = 0; j < len2; j++) {
if (str[i + j] != substr[j]) {
               break;
           }
}
       if (j == len2) {
return i;
```

```
        }
}
    return -1;
}

int main() {     char
str[100], substr[50];

    printf("Enter the main string: ");
scanf("%s", str);

    printf("Enter the substring: ");
    scanf("%s", substr);

    int index = findSubstring(str, substr);

    if (index != -1)         printf("Substring found at
index %d\n", index);     else         printf("Substring
not found\n");

    return 0;
}
```

OUTPUT:



```
Enter a string: COMPILER DESIGN LAB
Enter the substring: LA
Substring found at index 16
```

AIM:

To write a C program that compares two strings entered by the user and determines whether they are the same.

ALGORITHM:

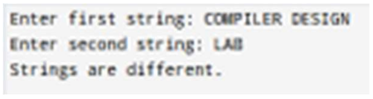
1. Start
2. Declare two character arrays to store the strings.
3. Take input for both strings from the user.
4. Use strcmp() to compare the two strings.
5. If the result is 0, print "Strings are the same."
6. Otherwise, print "Strings are different."
7. End

PROGRAM:

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

int main() {     char
str1[100], str2[100];

   printf("Enter first string: ");
scanf("%s", str1);

   printf("Enter second string: ");
scanf("%s", str2);

   if (strcmp(str1, str2) == 0)
printf("Strings are the same.\n");
else         printf("Strings are
different.\n");
```

```
   return 0;
}
```

OUTPUT:



```
Enter first string: COMPILER DESIGN
Enter second string: LAB
Strings are different.
```

AIM:

To write a C program that removes all spaces from a string entered by the user.

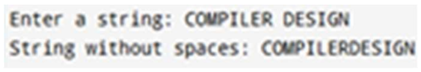
ALGORITHM:

1. Start
2. Declare a character array for input.
3. Take string input from the user.
4. Traverse the string: o Copy only non-space characters to a new position in the array.
5. Print the modified string.
6. End

PROGRAM:

```
#include <stdio.h>

int main() {     char
str[100], result[100];
   int i, j = 0;

   printf("Enter a string: ");
   scanf(" %[^\n]", str);

   for (i = 0; str[i] != '\0'; i++) {
       if (str[i] != ' ') {
result[j++] = str[i];
       }     }
result[j] = '\0';

   printf("String without spaces: %s\n", result);
return 0;
}
```

OUTPUT:



```
Enter a string: COMPILER DESIGN
String without spaces: COMPILERDESIGN
```

AIM:

To write a C program that calculates the frequency of each character in a given string.

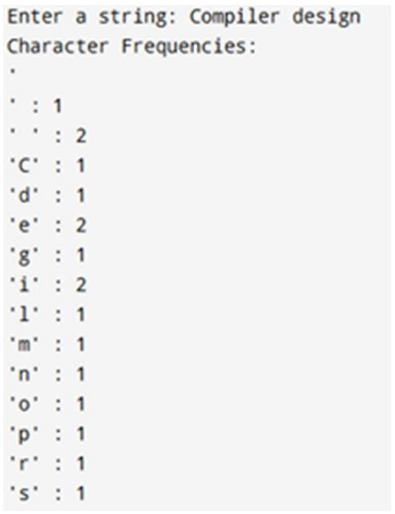
ALGORITHM:

1. Start
2. Declare a character array for input.
3. Declare an integer array freq[256] initialized to 0 (for ASCII character frequencies).
4. Take string input from the user.
5. Traverse the string: o Increment the frequency count for each character.
6. Print characters with their respective frequencies.
7. End

PROGRAM:

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

int main() {
char str[100];
   int freq[256] = {0};

   printf("Enter a string: ");
scanf("%s", str);

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

   printf("Character frequencies:\n");
for (int i = 0; i < 256; i++) {
       if (freq[i] > 0) {
printf("%c = %d\n", i, freq[i]);
       }     }
return 0;
}
```

OUTPUT:



```
Enter a string: Compiler design
Character Frequencies:
'
' : 1
' ' : 2
'C' : 1
'd' : 1
'e' : 2
'g' : 1
'i' : 2
'l' : 1
'm' : 1
'n' : 1
'o' : 1
'p' : 1
'r' : 1
's' : 1
```

AIM:

To write a C program that concatenates two strings entered by the user.

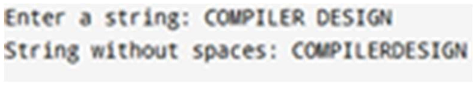
ALGORITHM:

1. Start
2. Declare two character arrays for input. 3 . Take input for both strings.
4. Use strcat() to concatenate the second string to the first.
5. Print the concatenated result.
6. End

PROGRAM:

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

int main() {     char
str1[100], str2[100];

    printf("Enter first string: ");
    scanf("%s", str1);

    printf("Enter second string: ");
scanf("%s", str2);

    strcat(str1, str2);

    printf("Concatenated String: %s\n",
str1);     return 0; }
```

OUTPUT:



```
Enter a string: COMPILER DESIGN
String without spaces: COMPILERDESIGN
```

AIM:

To write a C program that replaces all occurrences of a specific character in a string with another character.

ALGORITHM:

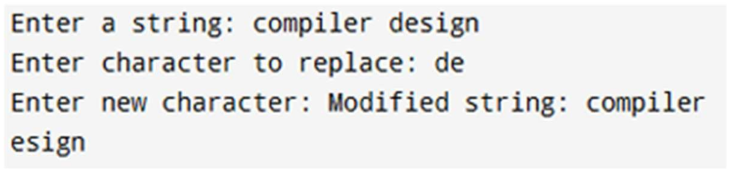
1. Start
2. Declare a character array for input.

3. Take string input from the user.
4. Take input for the character to replace and its replacement.
5. Traverse the string: o Replace occurrences of the old character with the new one.
6. Print the modified string.
7. End

PROGRAM:

```
#include <stdio.h>

int main() {     char str[100],
oldChar, newChar;

   printf("Enter a string: ");
scanf("%s", str);

   printf("Enter the character to replace: ");
   scanf(" %c", &oldChar);

   printf("Enter the new character: ");
scanf(" %c", &newChar);

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

   printf("Modified String: %s\n", str);
return 0;
}
```

OUTPUT:



```
Enter a string: compiler design
Enter character to replace: de
Enter new character: Modified string: compiler
esign
```

Implementation	
Output/Signature	

RESULT:

Thus the above program takes a string input, calculates and displays its length, copies and prints the string, concatenates it with a second input string, and finally compares both strings to check if they are the same or different.