EXP NO: 1. a DATE:

DEVELOP A SIMPLE C PROGRAM TO DEMONSTRATE A BASIC STRING OPERATIONS

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 <ctype.h> 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.
- 10. **End**

PROGRAM:

```
#include <stdio.h>
#include <ctype.h> #include
<string.h> int main() {    char
str[100];    printf("Enter a string:
");    fgets(str, sizeof(str), stdin);
size_t len = strlen(str);    if (len > 0
&& str[len - 1] == '\n') {
        str[len - 1] = '\0';
    }
    for (int i = 0; str[i] != '\0'; i++) {
str[i] = toupper((unsigned char)str[i]);
    }
    printf("Uppercase String: %s\n", str);
return 0;
}
```

```
$ gcc -o upper upper.c
$ ./upper
Enter a string: Hello World!
Uppercase String: HELLO WORLD!
```

EXP NO: 1.b DATE:

DEVELOP A SIMPLE C PROGRAM TO DEMONSTRATE A BASIC STRING OPERATIONS

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

```
#include <stdio.h>
#include <string.h>
int findSubstring(char str[], char sub[]) {
int strLen = strlen(str), subLen = strlen(sub);
  for (int i = 0; i \le strLen - subLen; i++) {
int j;
     for (j = 0; j < \text{subLen}; j++) \{
if (str[i+j] != sub[j]) {
          break;
     if (j == subLen) {
       return i; // Found at index i
       return -1; // Not
found
} int main() {
                 char str[100],
            printf("Enter a string:
sub[50];
      fgets(str, sizeof(str), stdin);
");
printf("Enter the substring: ");
fgets(sub, sizeof(sub), stdin);
str[strcspn(str, "\n")] = '\0';
sub[strcspn(sub, "\n")] = '\0';
index = findSubstring(str, sub);
```

```
if (index != -1)
    printf("Substring found at index %d\n", index);
else
    printf("Substring not found\n");
return 0;
}
```

```
$ gcc -o substring substring.c
$ ./substring
Enter a string: programming in C is powerful
Enter the substring: C
Substring found at index 14
```

EXP NO: 1.c DATE:

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: ");
fgets(str1, sizeof(str1), stdin);
printf("Enter second string: ");
fgets(str2, sizeof(str2), stdin);
str1[strcspn(str1, "\n")] = '\0';
str2[strcspn(str2, "\n")] = '\0';
if(strcmp(str1, str2) == 0)
printf("Strings are the same.\n");
printf("Strings are different.\n");
return 0;
}
```

```
$ gcc -o compare compare.c
$ ./compare
Enter first string: HelloWorld
Enter second string: HelloWorld
Strings are the same.
```

EXP NO: 1.d DATE:

DEVELOP A SIMPLE C PROGRAM TO DEMONSTRATE A BASIC STRING OPERATIONS

AIM:

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

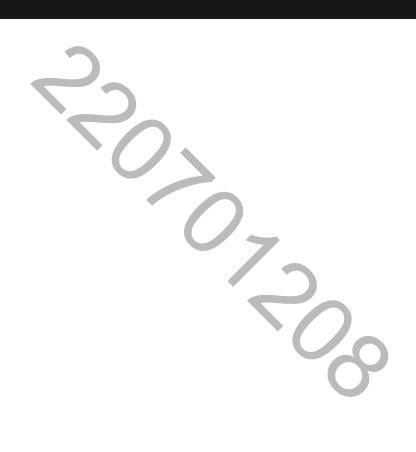
ALGORITHM:

- 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

```
#include <stdio.h>
void removeSpaces(char str[]) {
  int i, j = 0;
  for (i = 0; str[i] != '\0'; i++) {
    if (str[i] != ' ') {
      str[j++] = str[i];
    }
  }
  str[j] = '\0';
}
int main() {
    char str[100];
    printf("Enter a string: ");
  fgets(str, sizeof(str), stdin);
  removeSpaces(str);
```

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

```
$ gcc -o remove_spaces remove_spaces.c
$ ./remove_spaces
Enter a string: Welcome to Fedora Linux
String without spaces: WelcometoFedoraLinux
```



EXP NO: 1.e DATE:

DEVELOP A SIMPLE C PROGRAM TO DEMONSTRATE A BASIC STRING OPERATIONS

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

```
#include <stdio.h>
#include <string.h>
void countFrequency(char str[]) {
int freq[256] = \{0\};
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]);
int main() {
char str[100];
printf("Enter a string: ");
fgets(str, sizeof(str), stdin);
countFrequency(str);
return 0;
}
```

```
$ gcc -o char_freq char_freq.c
$ ./char_freq
Enter a string: Fedora Linux
Character Frequencies:
'F' : 1
'e' : 1
'd' : 1
'o' : 1
'r' : 1
'L' : 1
'i' : 1
'n' : 1
                     'x' : 1
```

EXP NO: 1. f DATE:

DEVELOP A SIMPLE C PROGRAM TO DEMONSTRATE A BASIC STRING OPERATIONS

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 streat() to concatenate the second string to the first.
- 5. Print the concatenated result.
- 6. End

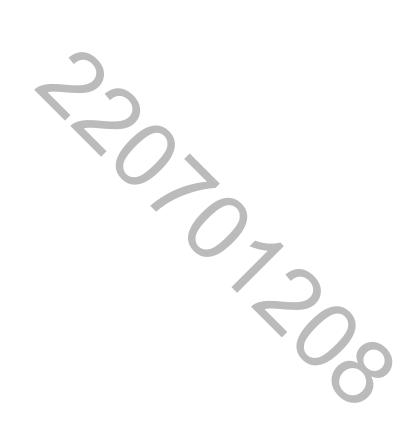
```
#include <stdio.h>
#include <string.h>
int main() {
    char str1[100], str2[50];
    printf("Enter first string: ");
    fgets(str1, sizeof(str1), stdin);
    printf("Enter second string: ");
    fgets(str2, sizeof(str2), stdin);
    str1[strcspn(str1, "\n")] = '\0';
    str2[strcspn(str2, "\n")] = '\0';
    strcat(str1, str2);
    printf("Concatenated string: %s\n", str1);
    return 0;
}
```

\$ gcc -o concat concat.c

\$./concat

Enter first string: Fedora Enter second string: Linux

Concatenated string: FedoraLinux



EXP NO: 1.g DATE:

DEVELOP A SIMPLE C PROGRAM TO DEMONSTRATE A BASIC STRING OPERATIONS

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

```
#include <stdio.h>
void replaceChar(char str[], char oldChar, char newChar) {
for (int i = 0; str[i] != '\0'; i++) {
  if (str[i] == oldChar) {
    str[i] = newChar;
}

int main() {
  char str[100], oldChar, newChar;
  printf("Enter a string: ");
  fgets(str, sizeof(str), stdin);
  printf("Enter character to replace: ");
```

```
scanf("%c", &oldChar);
getchar(); // Consume leftover newline character
printf("Enter new character: ");
scanf("%c", &newChar);
replaceChar(str, oldChar, newChar);
printf("Modified string: %s\n", str);
return 0;
}
```

```
$ gcc -o replace_char replace_char.c
$ ./replace_char
Enter a string: Fedora Linux
Enter character to replace: o
Enter new character: x
Modified string: Fedxra Linux
```

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.