**Batch: A1 Roll No.: 1911004**

**Experiment / assignment / tutorial No. 9**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE: To remove all occurrences of a word in string** |

**AIM:**

**Accept a string and a word from the user. Write a C program to delete the given word at all places from the string \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

**Demonstrate the concepts of modular programming through functions and dynamic memory allocation through use of pointers.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. **Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.**
2. **Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.**
3. **Introduction to programming and problem solving, G. Michael Schneider ,Wiley India edition.**
4. **Let’s C by Yashwant Kanetkar**
5. [**http://cse.iitkgp.ac.in/~rkumar/pds-vlab/**](http://cse.iitkgp.ac.in/~rkumar/pds-vlab/)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

**Accept two strings from the user. Identify the given word in the string. If a word match is found then remove all occurrences of word in given string.**

**Enter the string:**

**Hello World hello world Hello W**

**Enter the word to be removed:**

**Hello**

**OUTPUT:**

**World hello world W**

**Algorithm:**

**1) Start**

**2) input strings in s[] and w[]**

**3) l=strlen(s); l1 =strlen(w);**

**4) i=0;**

**5) if i<l -l1**

**then goto step 6**

**else**

**printf w and s ,**

**then goto step18**

**6) f=1;**

**7) j=0;**

**8) if j<l1**

**goto step 9**

**else**

**goto step 10**

**9) if str[i + j] != w[j]**

**then set f=0 and goto step 10**

**else**

**j=j+1 and goto step 8**

**10) if(str[i + j] != ' ' && str[i + j] != '\n' && str[i + j] != '\0')**

**f = 0**

**11) if(f == 1)**

**then goto step 12**

**else**

**goto step 5**

**12) j=i**

**13) if j<=l – l1**

**goto step 14**

**else**

**goto step 15**

**14) s[j] = s [j + l1] ;**

**j=j+1;**

**goto step 13**

**15) l = l – 1l ;**

**i=i-1;**

**16) i=i+1;**

**17) goto step 5**

**18) Stop**

**Implementation details:**

#include <stdio.h>

#include <string.h>

int main()

{ int i, j,l,l1,f;

char s[100],w[100];

printf("Enter any string: ");

fflush(stdin); scanf("%[^\n]",s); getc(stdin);

printf("Enter word to remove: ");

fflush(stdin); scanf("%[^\n]",w); getc(stdin);

printf("String before removing '%s' : \t %s\n",w, s);

l = strlen(s);

l1= strlen(w);

for(i=0;i<=l-l1;i++)

{

f = 1;

for(j=0;j<l1;j++)

{

if(s[i + j]!=w[j])

{

f = 0;

break;

}

}

if(s[i+j]!=' ' && s[i+j]!='\n' && s[i+j]!='\0') {

f = 0;

}

if(f==1)

{ for(j=i;j<=l-l1; j++)

{ s[j]= s[j+l1]; }

l = l - l1;

i--;

}

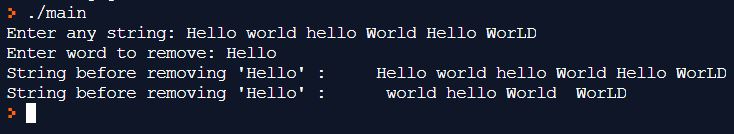
}

printf("String before removing '%s' : \t %s\n",w, s);

return 0;

}

**Output(s):**

****

**Conclusion:**

**The required word (all the occurrence ) was successfully removed from the given string.**

**Post Lab Descriptive Questions:**

1. **Write the output of the following C program code snippet with Justification**

**char c[] = "WORLDCUP2020";**

**char \*p =c;**

**printf("%c, %c", \*p,\*(p+p[3]-p[1]));**

**ANS:**

**W , T**

**char c[]=”WORLDCUP2020”;**

**There are 12 characters in this .**

**char \*p =c;**

**Now the address of c[] is c which is assigned to char \*p;**

**Thus \*p when printed give the value ‘W’ as c is base address ie c =&c[0].**

**(p+p[3]-p[1]) =&(c[0]+c[3]-c[1]); // <’W’ + ’L’ - ‘O’ = ’T’>**

1. **Explain the different ways of taking string input from the user with the help of an example of each**

**ANS: char string[100];**

* **scanf(“%s”, string);**

**here the format specifier %s is used to input string and string which is address of string is passed as &string[0] is the base address. This only allows to accept the string before ‘ ’ / ‘\t’ .we cant enter the sentence this takes only 1 word. It is automatically terminated by ‘\0’.**

* **gets(string );**

**here we pass the address of the string ie char array name string and here we can input entire sentence/paragraph. Here the string ends only when enter is pressed and string is automatically terminated by ‘\0’.**

* **scanf(“%[^\n]”,string);**

**here the format specifier %[^\n] is used to input string and string which is address of string is passed as &string[0] is the base address. But unlike scanf(“%s”,string);**

**it helps us to input string until \n is encountered thus the %[^\n] is used which means accept input until \n comes. It is automatically terminated by ‘\0’**

1. **Write a program which reads your name from keyboard and outputs a list of ASCII codes, which represent your name.**

**ANS:**

#include<stdio.h>

#include<string.h>

int main()

{ char n[20]; int j;

fflush(stdin);

printf("\t NAME : \n");

getc(stdin);

scanf("%[^\n]",n);

for(j=0;n[j]!='\0';j++)

printf("%d \t",n[j]);

return 0;

}



**Date: 03-05-2020 Signature of faculty in-charge**