| **Name** | Mahadev Balla |
| --- | --- |
| **UID no.** | 2023300010 |
| **Experiment No.** | 7 |

| **AIM:** | To solve given problems related to strings. |
| --- | --- |
| **Program 1** | |
| **PROBLEM STATEMENT :** | Write a program to check whether a string is Palindrome or not. |
| **ALGORITHM:** | 1) Start  2) Declare character arrays `str1` and `str2` of size 20.  3) Print a message to the user: `printf("Enter a string : ");`  4) Read a line of text until a newline character is encountered and store it in `str1`: `scanf("%[^\n]s", str1);`  5) Print the entered string: `printf("\nThe entered string is: "); puts(str1);`  6) Initialize a variable `i` to 0.  7) Loop from `i=0` to `i<strlen(str1)`:  a. Assign the character at the corresponding position from the end of `str1` to `str2[i]`.  8) Null-terminate `str2` to ensure it is a valid C string: `str2[strlen(str1)] = '\0';`  9) Compare `str1` and `str2` using `strcmp` function:  a. If the comparison result is 0:  i. Print "Entered string is palindrome."  b. Else:  i. Print "Entered string isn't palindrome."  10) End |
| **PROGRAM:** | #include <stdio.h>  #include <string.h>  int main ()  {  char str1[20], str2[20];  printf("Enter a string : ");  scanf("%[^\n]s",str1);  printf("\nThe entered string is : ");  puts(str1);  for(int i=0; i<strlen(str1);i++)  {  str2[i]=str1[strlen(str1)-i-1];  }  if(strcmp(str1,str2)==0)  printf("\nEntered string is palindrome.\n");  else  printf("\nEntered string isn't palindrome.\n");  return 0;  } |
| **RESULT:** | |
| **Program 2** | |
| **PROBLEM STATEMENT :** | Write a program to count the number of vowels, consonants, total characters and words in the given string. |
| **ALGORITHM:** | 1) Start  2) Define a function `count` that takes a character array `x` as input and calculates the number of vowels, consonants, words, and characters:  a. Initialize variables `v` (vowels), `c` (consonants), `sp` (spaces), and `i`.  b. Loop through the characters of the input string `x` from index 0 to the length of the string:  i. If the current character is a space (' '), increment the `sp` counter by 1.  ii. Else, if the current character is a vowel (a, e, i, o, or u), increment the `v` counter by 1.  iii. Else, increment the `c` counter by 1.  c. Calculate the total number of words as `sp + 1`.  d. Calculate the total number of characters excluding spaces as `strlen(x) - sp`.  e. Print the number of vowels, consonants, words, and characters.  3) In the `main()` function:  a. Declare a character array `var1` of size 60.  b. Prompt the user to enter a string: `printf("Enter a string : ");`  c. Read the input string until a newline character is encountered: `scanf("%[^\n]s", var1);`  d. Call the `count` function with the input string `var1`.  e. Return 0 to indicate successful execution.  4) End |
| **PROGRAM:** | #include<stdio.h>  #include<string.h>  int count(char x[]){  int v=0,c=0,sp=0,i;  for(i = 0 ; i<strlen(x) ; i++){  if(x[i] == ' '){  sp=sp+1;  }  else if(x[i] == 'a' || x[i] == 'e' || x[i] == 'i' || x[i] == 'o' || x[i] == 'u'){  v=v+1;  }  else {  c=c+1;  }  }  printf("The number of vowels are %d\n " ,v);  printf("The number of consonants are %d\n ",c);  printf("The number of words are %d\n ",(sp+1));  printf("The number of characters are %ld\n " ,(strlen(x)-sp));  }  int main(){  char var1[60];  printf("Enter a string : ");  scanf("%[^\n]s",var1);  count(var1);  return 0;  } |
| **RESULT:** | |
| **Program 3** | |
| **PROBLEM STATEMENT:** | Write a program to find and replace a particular word from the string.  Input: I LOVE CANADA BECAUSE CANADA IS A GREAT COUNTRY  Word to find: CANADA  Word to Replace: INDIA  Output: I LOVE INDIA BECAUSE INDIA IS A GREAT COUNTRY |
| **ALGORITHM:** | 1) Start  2) Define the function `char\* replace(const char\* s, const char\* old, const char\* new)`  a. Initialize variables `result`, `i`, and `cnt` to 0.  b. Calculate the lengths of `old` and `new` strings and store them in `oldlen` and `newlen`.  c. Loop through the input string `s` until the end of the string:  i. If the substring from the current position matches `old`:  - Increment `cnt` by 1.  - Move the index `i` by the length of `old` minus 1.  d. Allocate memory for the `result` string: `malloc(i + cnt \* (newlen - oldlen) + 1)`.  e. Reset the index `i` to 0.  f. Loop through the input string `s`:  i. If the substring from the current position matches `old`:  - Copy the `new` string into `result` at index `i`.  - Increment `i` by `newlen`.  - Increment `s` by `oldlen`.  ii. Else, copy the current character from `s` into `result` at index `i`.  - Increment both `i` and `s`.  g. Add the null terminator at the end of `result`.  h. Return `result`.  3) In the `main()` function:  a. Declare variables `str`, `x`, `y`, and `result`.  b. Initialize `str` with the input string.  c. Initialize `x` with the substring to be replaced (`old`).  d. Initialize `y` with the replacement string (`new`).  e. Print the original string: `printf("Original string is : %s\n", str)`.  f. Call the `replace` function with `str`, `x`, and `y` and store the result in `result`.  g. Print the modified string: `printf("New string is : %s\n", result)`.  h. Free the allocated memory for `result`: `free(result)`.  i. Return 0 to indicate successful execution.  4) End |
| **PROGRAM:** | #include <stdio.h>  #include <string.h>  #include <stdlib.h>  char\*replace(const char\* s, const char\* old, const char\* new)  {    char\* result;  int i, cnt=0;  int newlen = strlen(new);  int oldlen = strlen(old);    for(i=0; s[i]!='\0'; i++)  if (strstr(&s[i], old) == &s[i])  {  cnt++;  i += oldlen - 1;  }  result = (char\*)malloc(i+cnt\*(newlen-oldlen)+1);  i=0;  while(\*s)  {  if(strstr(s,old) == s)  {  strcpy(&result[i],new);  i+=newlen;  s+=oldlen;  }  else  result[i++]=\*s++;  }    result[i]='\0';  return result;  }  int main ()  {  char str[]="I LOVE CANADA BECAUSE CANADA IS A GREAT COUNTRY";  char x[]="CANADA";  char y[]="INDIA";  char\* result = NULL;  printf("Original string is : %s\n",str);  result=replace(str,x,y);  printf("New string is : %s\n",result);  free(result);  return 0;  } |
| **RESULT:** | |
| **CONCLUSION:** | Studied the application of functions to solve given string related problems. |