Name	**PRATIK PUJARI**
UID no.	**2020300054**
Experiment No.	7

AIM:	Implement various text processing problems.
Program 1	
PROBLEM STATEMENT :	Write a program to delete all repeated words in string.
PROGRAM:	ALGORITHM: **void main()** STEP 1: START. STEP 2: Initialize the char array str[100] input the user's input. STEP 3: Initialize the loop counter's i,j,k to zero and declare len ,nwords and count variables. STEP 4: Input the string from the user and store it in the "str" string. STEP 5: Call the predefined function space and store the returned value to nwords. STEP 6: Printf("The sentence without duplicates is :") STEP 7: Call the predefined function fill(str_dup,str) ,dup_remove(str_dup,str,nwords) and display(str_dup,nwords). STEP 8:END. **int space(char str[],int len)** STEP 1: START. STEP 2: Initalize the loop counter i,nspaces to zero. STEP 3: For I equal to zero and less than len-1 ,Repeat the steps 3.1 and 3.2 or else if the condition fails go to step 4. STEP 3.1: If str[i] is equal to ' ' and increment the nspaces by one or else go to step 3.2. STEP 3.2: Increment the loop counter by one. STEP 4: END. **void fill_str(char str_dup[][15],char str[])** STEP 1: START. STEP 2: Initialize the variables row to zero and i,j to zero. STEP 3: For i equal to zero and str[i] not equal to '\0', Repeat the steps 3.1,3.2 and 3.3 or else go to step 4 STEP 3.1: If str[i] equal to ' ' then str_dup[row][j] = '\0' and increment row by one and set j to zero or else go to step 3.2. STEP 3.2: Do str_dup[row][j]=str[i] and increment j by one. STEP 4: Do str_dup[i][j+1]='\0' STEP 5: END.

void dup_remove(char str_dup[][15],char str[],int nwords)
STEP 1: START.
STEP 2: Initialize the loop counters i,j.
STEP 3: For I equal to zero and less than nwords, Repeat the steps 3.1 and 3.2 or else if the condition fails go to step 4.
STEP 3.1: For j equal to i+1 and less than nwords ,Repeat the steps 3.1.1 and 3.1.2 or else if the condtion fails go to step 3.2.
STEP 3.1.1: If strcmp(str_dup[i],str[j]) equal to zero then initialize col to zero and while (str_dup[j][col]= '\0' and increment col by one or else go to
STEP 3.1.2: Increment the loop counter j by one .
STEP 3.2: Increment the loop counter i by one .
STEP 4: END.

void display(char str_dup[][15],int nwords)
STEP 1: START.
STEP 2: Initialize i,j to zero.
STEP 3: For i equal to zero and less than nwords ,Repeat the steps 3.1, 3.2 or else if the condition fails go to step 4.
STEP 3.1: For j equal to zero and str_dup[i][j] not equal to '\0' ,Repeat the steps 3.1.1 ,3.1.2 and 3.1.3 or else if the condition fails go to step 3.2.
STEP 3.1.1: If str_dup[i][j] equal to '\0' then continue or else go to 3.1.2.
STEP 3.1.2: Printf("%c ",str_dup[i][j]).
STEP 3.1.3: Increment the loop counter j by one.
STEP 3.2: Increment the loop counter i by one.
STEP 4: END.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
int space(char str1[],int);
void fill_str(char str_dup[][15],char str[]);
void dup_remove(char str_dup[][15],char str[],int);
void display(char str_dup[][15],int);
void main()
{
   char str[100];
   int len,count;
   int i,j=0,row=0;
   printf("Enter a Sentence:\n");
   gets(str);
   len=strlen(str);
   int nwords=space(str,len);
   char str_dup[nwords][15];
   printf("The sentence without duplicates is :");
   fill_str(str_dup,str);
   dup_remove(str_dup,str,nwords);
   display(str_dup,nwords);
}
```

```
int space(char str[],int len)
{
   int nspaces=0;
   int i;
   for(i=0;i<len-1;i++)
   {
      if(str[i]==' ')
      nspaces++;
   }
   return nspaces+1;
}
void fill_str(char str_dup[][15],char str[])
{
   int row=0;
   int i,j=0;
   for(i=0;str[i]!='\0';i++)
   {
      if(str[i]==' ')
      {
         str_dup[row][j]='\0';
         row++;
         j=0;
      }
      else
      {
         str_dup[row][j]=str[i];
         j++;
      }
   }
   str_dup[i][j+1]='\0';
}
void dup_remove(char str_dup[][15],char str[],int nwords)
{
   int i,j;
   for(i=0;i<nwords;i++)
   {
      for(j=i+1;j<nwords;j++)
      {
         if(strcmp(str_dup[i],str_dup[j])==0)
         {
            int col=0;
            while(str_dup[j][col]!='\0')
            {
               str_dup[j][col]='\0';
               col++;
            }
         }
      }
   }
```

	}
	void display(char str_dup[][15],int nwords)
	{
	int i,j;
	for(i=0;i<nwords;i++)
	{
	for(j=0;str_dup[i][j]!='\0';j++)
	{
	if(str_dup[i][j]=='\0')
	continue;
	printf("%c ",str_dup[i][j]);
	}
	}
	}

RESULT: All the duplicates in the string were removed.	
INPUT:	Hello welcome to C programming ,hello we welcome you again to C class
OUTPUT:	Enter a Sentence: Enter a Sentence: Hello welcome to C programming ,hello we welcome you again to C class The sentence without duplicates is :Hello welcome to C programming ,hello we welcome you again class

Program 2	
PROBLEM STATEMENT :	Write a program to find and replace a particular word from the string.
PROGRAM:	ALGORITHM: **void main()** STEP 1: START. STEP 2: Initialize the char array str[100] input the user's input. STEP 3: Initialize the loop counter's i,j,k to zero and declare len and count variables and nwords. STEP 4: Input the string from the user and store it in the "str" string. STEP 5: Call the predefined function space and store the returned value to nwords. STEP 6: Declare a char array str_dup[nwords][15] STEP 7: Call the predefined function fill(str_dup,str). STEP 8: Declare two char array rep_word[20] and new[20] STEP 9: Printf("Enter the word to be replaced ") STEP 10: Input the word and store it in the char array rep_word[20] STEP 11: If check(str_dup,str,rep_word,nwords equal to one then go to step 11.1 and 11.2 or else go to step 12. STEP 11.1: Printf(" Word found Enter the word to be replaced it with .") and store it in char array new[20]. STEP 11.2: Call the predefined function replace(str_dup,rep_word,new,nwords), display(str_dup,nwords). STEP 12: Printf("Word not found"). STEP 13: END **int space(char str[],int len)** STEP 1: START. STEP 2: Initalize the loop counter i,nspaces to zero. STEP 3: For I equal to zero and less than len-1 ,Repeat the steps 3.1 and 3.2 or else if the condition fails go to step 4. STEP 3.1: If str[i] is equal to ' ' and increment the nspaces by one or else go to step 3.2. STEP 3.2: Increment the loop counter by one. STEP 4: END. **void fill_str(char str_dup[][15],char str[],int)** STEP 1: START. STEP 2: Initialize the variables row to zero and i,j to zero. STEP 3: For i equal to zero and str[i] not equal to '\0', Repeat the steps 3.1,3.2 and 3.3 or else go to step 4 STEP 3.1: If str[i] equal to ' ' then str_dup[row][j] = '\0' and increment row by one and set j to zero or else go to step 3.2. STEP 3.2: Do str_dup[row][j]=str[i] and increment j by one. STEP 4: Do str_dup[i][j+1]='\0' STEP 5: END.

int check(char str_dup[][15],char str[],char rep_word[],int nwords)
STEP 1: START.
STEP 2: Initialize the loop counters i,j.
STEP 3: For i equal to zero and less than nwords, Repeat step 3.1 and 3,2 or else if the condition fails go to step 4.
STEP 3.1: If strcmp(str_dup[i],rep_word) is equal to zero then return 1
STEP 3.2: Increment the loop counter i by one.
STEP 4: END.

void replace(char str_dup[][15],char str[],char rep_word[],char new[],int nwords)
STEP 1: START.
STEP 2: Initialize the loop counter i.
STEP 3: For i equal to zero and nwords, Repeat the steps 3.1 and 3.2 or else if the condtion fails go to step 4.
STEP 3.1: If strcmp(str_dup[i],rep_word) equal to zero then strcpy(str_dup[i],new) or else go to step 3.2.
STEP 3.2: Increment the loop counter by one.
STEP 4: END

void display(char str_dup[][15],int nwords)
STEP 1: START.
STEP 2: Initialize i,j to zero.
STEP 3: For i equal to zero and less than nwords ,Repeat the steps 3.1, 3.2 or else if the condition fails go to step 4.
STEP 3.1: For j equal to zero and str_dup[i][j] not equal to '\0' ,Repeat the steps 3.1.1 ,3.1.2 and 3.1.3 or else if the condition fails go to step 3.2.
STEP 3.1.1: If str_dup[i][j] equal to '\0' then continue or else go to 3.1.2.
STEP 3.1.2: Printf("%c ",str_dup[i][j]).
STEP 3.1.3: Increment the loop counter j by one.
STEP 3.2: Increment the loop counter i by one.
STEP 4: END.

PROGRAM:

```
#include<stdio.h>
#include<string.h>
int space(char str1[],int);
void fill_str(char str_dup[][15],char str[]);
int check(char str_dup[][15],char str[],char rep_word[],int);
void replace(char str_dup[][15],char rep_word[],char new[],int);
void display(char str_dup[][15],int );
void main()
{
   char str[100];
   int len,count;
   int i,j=0,row=0;
   printf("Enter a Sentence:\n");
```

```
    scanf("%[^\n]s", str);
    len=strlen(str);
    nwords=space(str,len);
    char str_dup[nwords][15];
    fill_str(str_dup,str);
    char rep_word[20],new[20];
    printf("Enter the word to be replaced\n");
    scanf("%s",rep_word);
    if(check(str_dup,str,rep_word,nwords)==1)
    {
        printf("Word found....Enter the word to be replaced it with\n");
        scanf("%s",new);
        replace(str_dup,rep_word,new,nwords);
        display(str_dup,nwords);
    }
    else
    printf("Word not found.");
}
int space(char str[],int len)
{
    int nspaces=0;
    int i;
    for(i=0;i<len-1;i++)
    {
        if(str[i]==' ')
        nspaces++;
    }
    return nspaces+1;
}
void fill_str(char str_dup[][15],char str[])
{
    int row=0;
    int i,k,j=0;
    for(i=0;str[i]!='\0';i++)
    {
        if(str[i]==' ')
        {
            str_dup[row][j]='\0';
            row++;
            j=0;
        }
        else
        {
            str_dup[row][j]=str[i];
            j++;
        }
    }
    str_dup[row][j]='\0';
}
```

```
int check(char str_dup[][15],char str[],char rep_word[],int nwords)
{
   int i,j;
   for(i=0;i<nwords;i++)
   {
      if(strcmp(str_dup[i],rep_word)==0)
      {
         return 1;
      }
   }
}
void replace(char str_dup[][15],char rep_word[],char new[],int nwords)
{
   int i;
   for(i=0;i<nwords;i++)
   {
      if(strcmp(str_dup[i],rep_word)==0)
      {
         strcpy(str_dup[i],new);
      }
   }
}
void display(char str_dup[][15],int nwords)
{
   int i,j;
   for(i=0;i<nwords;i++)
   {
      for(j=0;str_dup[i][j]!='\0';j++)
      {
         if(str_dup[i][j]=='\0')
         continue;
         printf("%c",str_dup[i][j]);
      }
      printf(" ");
   }
}
```

RESULT: The particular word to be changed is replaced accordingly

INPUT:	Enter a sentence: I love C programming more than Java

OUTPUT:	Enter a Sentence: I love C programming more than Java Enter the word to be replaced C Word found....Enter the word to be replaced it with C++ I love C++ programming more than Java
CONCLUSION:	To use a string we make use of char array in C and use string,h header file to make use of string related functions