Experiment No: 1

Program Title: Find the total number of alphabets, digits, and special character from a given string input.

Source Code:

```
#include <iostream>
#include <string>
#include <ctype.h>
using namespace std;
int main() {
    // Write C++ code here
    string inp;
    int alphabet = 0 , sp_char = 0 , digit = 0;
    getline(cin, inp);

    for(int i=0; inp[i] != '\0'; i++){

        if(isalpha(inp[i])){
            alphabet++;
        }
        else if(isdigit(inp[i])){
            digit++;
        } else {
            sp_char++;
        }
    }

    cout << endl << "Total No. of Alphabets : " << alphabet;
    cout << endl << "Total No. of digits : " << digit;
    cout << endl << "Total No. of Special Character : " << sp_char;

    return 0;
}
```

Output:

```
This 789 &
Total No. of Alphabets : 4
Total No. of digits : 3
Total No. of Special Character : 3
```

Experiment No: 2

Program Title: Find the total number of characters, words, and lines from a given string input.

Source Code:

```
#include<bits/stdc++.h>
#include<string>
using namespace std;
int main(){
   char ch[500];
   cout << "Enter a string : ";
   scanf("%[^~]", ch);
   int character = 0, word = 0, line = 0, i;
   for(i=0; ch[i] != '\0'; i++){
      if(ch[i] == '\n'){
         word++;
         line++;
      }
      else{
         if(ch[i] == ' ' || ch[i] == '\t'){
            word++;
         }
         else{
            character++;
         }
      }
   }
   line++;
   word++;
   cout << endl << "Total no. of Lines : " << line;
   cout << endl << "Total no. of Characters : " << character;
   cout << endl << "Total no. of Words : " << word << endl;
}
```

Output:

```
Enter a string : HElo
This is
NO~
Total no. of Lines : 3
Total no. of Characters : 12
Total no. of Words : 5
```

Experiment No: 3

Program Title: Write a program to check for Valid Identifier.

Source Code:

```
#include<bits/stdc++.h>
#include<ctype.h>
#include<string>
using namespace std;
int main(){
    string ch;
    cout << "Enter a string : ";
    getline(cin, ch);
    int flag;
    if(isalpha(ch[0]) || ch[0] == '_'){
        flag = 1;
    }
    for(int i=1; ch[i]!='\0'; i++){
        if(!isalpha(ch[i]) && !isdigit(ch[i])){
            flag = 0;
            break;
        }
    }
    if(flag == 1){
        cout << endl << "Valid Identifier";
    }
    else{
        cout << endl << "Not a Valid Identifier";
    }

}
```

Output:

```
Enter a string : const
Valid Identifier
```

```
Enter a string : @hello
Not a Valid Identifier
```

Experiment No: 4

Program Title: Write a program to remove all white space from a string.

Source Code:

```
#include<bits/stdc++.h>
#include<string>

using namespace std;

int main(){
    char str[500];
    cout << "Enter a String ending with ~ : ";
    scanf("%[^~]", str);

    cout << endl << "The string before removing spaces: " << str;
    int length = 0, j, i;
    length = sizeof(str) / sizeof(str[0]);

    for(i = 0; i<length; i++){
        if(str[i] == ' '){
            for(j = i; j<length; j++){
                str[j] = str[j+1];
            }
        }
        length--;
    }
    cout << endl << "The string after removing spaces: " << str << endl;
}
```

Output:

```
Enter a String ending with ~ : Ba n g l ad es h~
The string before removing spaces: Ba n g l ad es h
The string after removing spaces: Bangladesh
```

Experiment No: 5

Program Title: Checking for valid keyword.

Source Code:

```
#include<bits/stdc++.h>
#include<string>
using namespace std;
int main()
{
    char str[100];
    char keyword[32][10] = {"auto","double","int","struct","break","else","long",
       "switch","case","enum","register","typedef","char",
       "extern","return","union","const","float","short",
       "unsigned","continue","for","signed","void","default",
       "goto","sizeof","voltile","do","if","static","while"};
    cout << "Enter a string to chekc whether it is keyword or not : ";
    cin >> str;
    int i, flag = 0;
    for(i = 0; i<32; i++){
        if(strcmp(str,keyword[i]) == 0){
            flag = 1;
        }
    }
    if(flag == 1){
        cout << endl << str << " is a valid keyword" << endl;
    }
    else{
        cout << endl << str << " is not a valid keyword" << endl;
    }
}
```

Output:

```
Enter a string:
double
double is a valid keyword
```

```
Enter a string:
hello
hello is not a valid keyword
```

Experiment No: 6

Program Title: Write a program to check a string is a comment or not.

Source Code:

```
#include<bits/stdc++.h>
#include<string>
using namespace std;
int main(){
    char com[500];
    cout << "Enter a comment ending with a ~ : ";
    scanf("%[^~]", com);
    int length = strlen(com);
    int flag = 0;
    if(com[0] == '/'){
        if(com[1] == '/'){
            flag = 1;
        }
        else if(com[1] == '*'){
            if(com[length-2] == '*' && com[length-1] == '/'){
                flag = 1;
            }
        }
    }
    if(flag==1){
        cout << endl << "This is a comment";
    }
    else{
        cout << endl << "This is not a comment";
    }
}
```

Output:

```
//comment~
This is a comment
```

```
/* This
is a
commetn */
~
This is not a comment
```

Experiment No: 7

Program Title: Write a program to check a grammar is either left recursive or not.

Source Code:

```
#include<stdio.h>
#include<string.h>
#define SIZE 10
int main () {
   char non_terminal;
   char beta,alpha;
   int num,i;
   char production[10][SIZE];
   int index=3; /* starting of the string following "->" */
   printf("Enter Number of Production : ");
   scanf("%d",&num);
   printf("Enter the grammar as E->E-A :\n");
   for(i=0;i<num;i++){
      scanf("%s",production[i]);
   }
   for(i=0;i<num;i++){
      printf("\nGRAMMAR :  %s",production[i]);
      non_terminal=production[i][0];
      if(non_terminal==production[i][index]){
         alpha=production[i][index+1];
         printf(" is left recursive.\n");
      }
      else
      {
         printf(" is not left recursive.\n");
      }
   }
   return 0;
}
```

Output:

```
Enter Number of Production : 2
Enter the grammar as E->E-A :
A->F-c
S->S+a
GRAMMAR :  A->F-c is not left recursive.

GRAMMAR :  S->S+a is left recursive.
```

Experiment No: 8

Program Title: Write a program to evaluate postfix expression.

Source Code:

```
#define SIZE 50             /* Size of Stack */
#include <ctype.h>
#include<stdio.h>
int s[SIZE];
int top=-1;         /* Global declarations */
int push(int elem)
{                           /* Function for PUSH operation */
    s[++top]=elem;
}
int pop()
{                           /* Function for POP operation */
    return(s[top--]);
}
int main()
{                           /* Main Program */
    char pofx[50],ch;
    int i=0,op1,op2;
    printf("\n\nRead the Postfix Expression ? ");
    scanf("%s",pofx);
    while( (ch=pofx[i++]) != '\0')
    {
        if(isdigit(ch)) push(ch-'0'); /* Push the operand */
        else
        {          /* Operator,pop two  operands */
            op2=pop();
            op1=pop();
            switch(ch)
            {
            case '+':push(op1+op2);break;
            case '-':push(op1-op2);break;
            case '*':push(op1*op2);break;
            case '/':push(op1/op2);break;
            }
        }
    }
    printf("\n Given Postfix Expn: %s\n",pofx);
    printf("\n Result after Evaluation: %d\n",s[top]);
}
```

Output:

```
Read the Postfix Expression ? 345*+
Given Postfix Expn: 345*+

 Result after Evaluation: 23
```

Experiment No: 8

Program Title: Write a program to convert infix expression to postfix expression.

Source Code:

```
#define SIZE 50             /* Size of Stack */
#include <ctype.h>
#include<stdio.h>
int s[SIZE];
int top=-1;         /* Global declarations */
int push(int elem)
{                           /* Function for PUSH operation */
    s[++top]=elem;
}
int pop()
{                          /* Function for POP operation */
    return(s[top--]);
}
int main()
{                            /* Main Program */
    char pofx[50],ch;
    int i=0,op1,op2;
    printf("\n\nRead the Postfix Expression ? ");
    scanf("%s",pofx);
    while( (ch=pofx[i++]) != '\0')
    {
        if(isdigit(ch)) push(ch-'0'); /* Push the operand */
        else
        {          /* Operator,pop two  operands */
            op2=pop();
            op1=pop();
            switch(ch)
            {
            case '+':push(op1+op2);break;
```

```
            case '-':push(op1-op2);break;
            case '*':push(op1*op2);break;
            case '/':push(op1/op2);break;
            }
        }
    }
    printf("\n Given Postfix Expn: %s\n",pofx);
    printf("\n Result after Evaluation: %d\n",s[top]);
}
```

Output:

```
Enter the expression :: a+b*c
abc*+
```