FACULTY OF ENGINEERING AND TECHNOLOGY
BACHELOR OF TECHNOLOGY

COMPILER DESIGN

LABRATORY (203105361)

6TH SEMESTER

COMPUTER SCIENCE & ENGINEERING

LABORATORY MANUAL

CERTIFICATE

This is to certify that

Mr./Ms**. DARA HARHSITHA** with Enrollment no. **210303124354** has successfully completed his/her laboratory experiments in the **COMPILER DESIGN (203105361)** from the department of **COMPUTER SCIENCE AND ENGINEERING** during the academic year **2023-2024.**

Date of Submission: ..... **Staff In charge: .....**

Head Of Department: .....

TABLE OF CONTENT

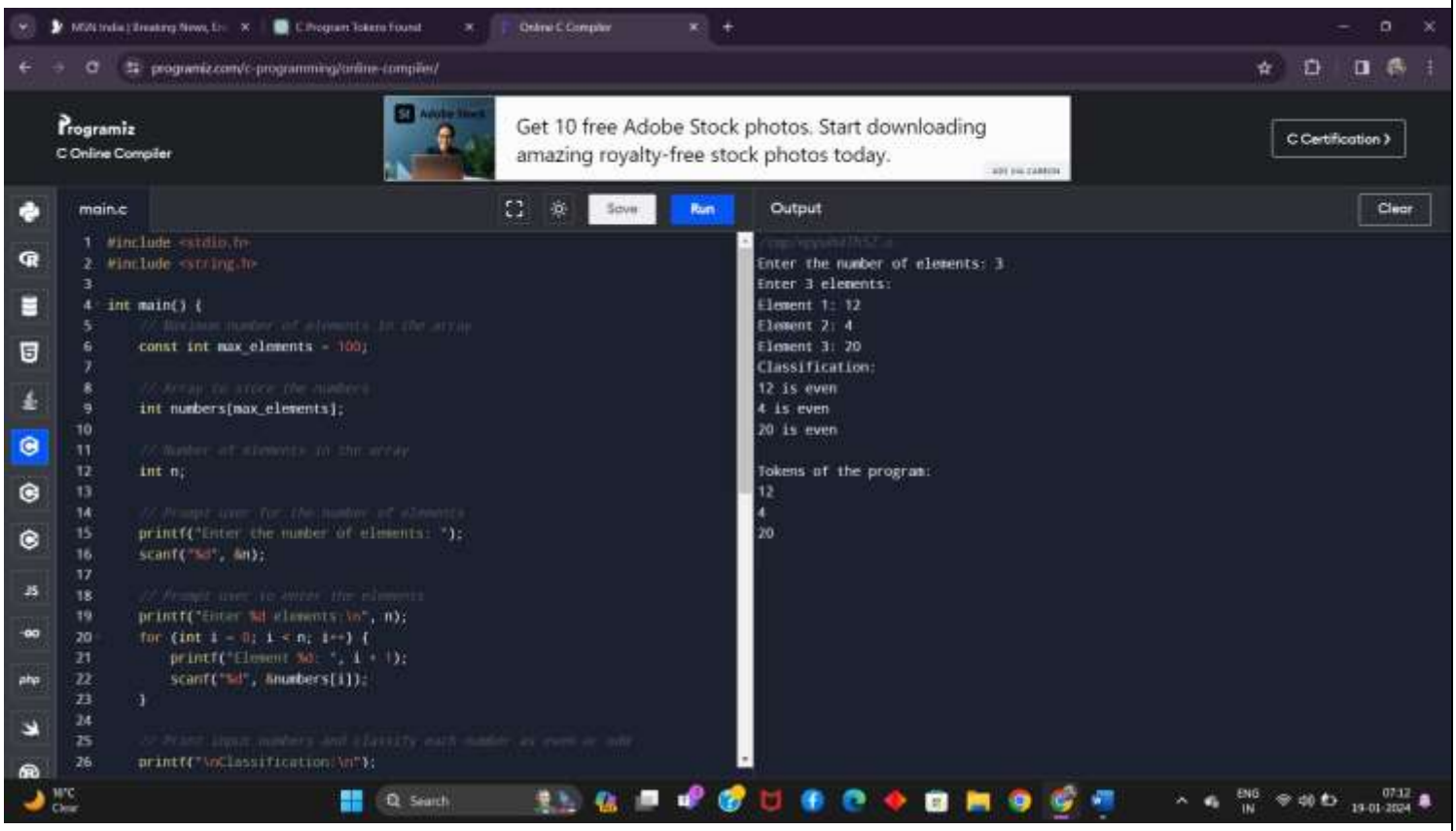
EXPERIMENT -01

1:- Write a program to find out the tokens.

CODE:

```
#include <stdio.h>
#include <string.h>

int main() {
   // Maximum number of elements in the array
   const int max_elements = 100;

   // Array to store the numbers
   int numbers[max_elements];

   // Number of elements in the array
   int n;

   // Prompt user for the number of elements
   printf("Enter the number of elements: ");
   scanf("%d", &n);

   // Prompt user to enter the elements
   printf("Enter %d elements:\n", n);
   for (int i = 0; i < n; i++) {
      printf("Element %d: ", i + 1);
      scanf("%d", &numbers[i]);
   }

   // Print input numbers and classify each number as even or odd
   printf("\nClassification:\n");
   for (int i = 0; i < n; i++) {
```

```
        if (numbers[i] % 2 == 0) {
            printf("%d is even\n", numbers[i]);
        } else {
            printf("%d is odd\n", numbers[i]);
        }
    }

    // Tokenize the program based on whitespace characters
    printf("\nTokens of the program:\n");
    for (int i = 0; i < n; i++) {
        char buffer[20];  // Assuming a maximum token length of 19 characters
        snprintf(buffer, sizeof(buffer), "%d", numbers[i]);
        char *token = strtok(buffer, " \t\n");

        // Loop through tokens and print them
        while (token != NULL) {
            printf("%s\n", token);

            // Get the next token
            token = strtok(NULL, " \t\n");
        }
    }

    return 0;
}
```


programiz.com/c-programming/online-compiler/
Programiz
C Online Compiler
Get 10 free Adobe Stock photos. Start downloading amazing royalty-free stock photos today.
C Certification
main.c
Save
Run
Output
Clear
#include <stdio.h>
#include <string.h>
int main() {
const int max_elements = 100;
int numbers[max_elements];
int n;
printf("Enter the number of elements: ");
scanf("%d", &n);
printf("Enter %d elements:\n", n);
for (int i = 0; i < n; i++) {
printf("Element %d: ", i + 1);
scanf("%d", &numbers[i]);
}
printf("\nClassification:\n");
Enter the number of elements: 3
Enter 3 elements:
Element 1: 12
Element 2: 4
Element 3: 20
Classification:
12 is even
4 is even
20 is even
Tokens of the program:
12
4
20

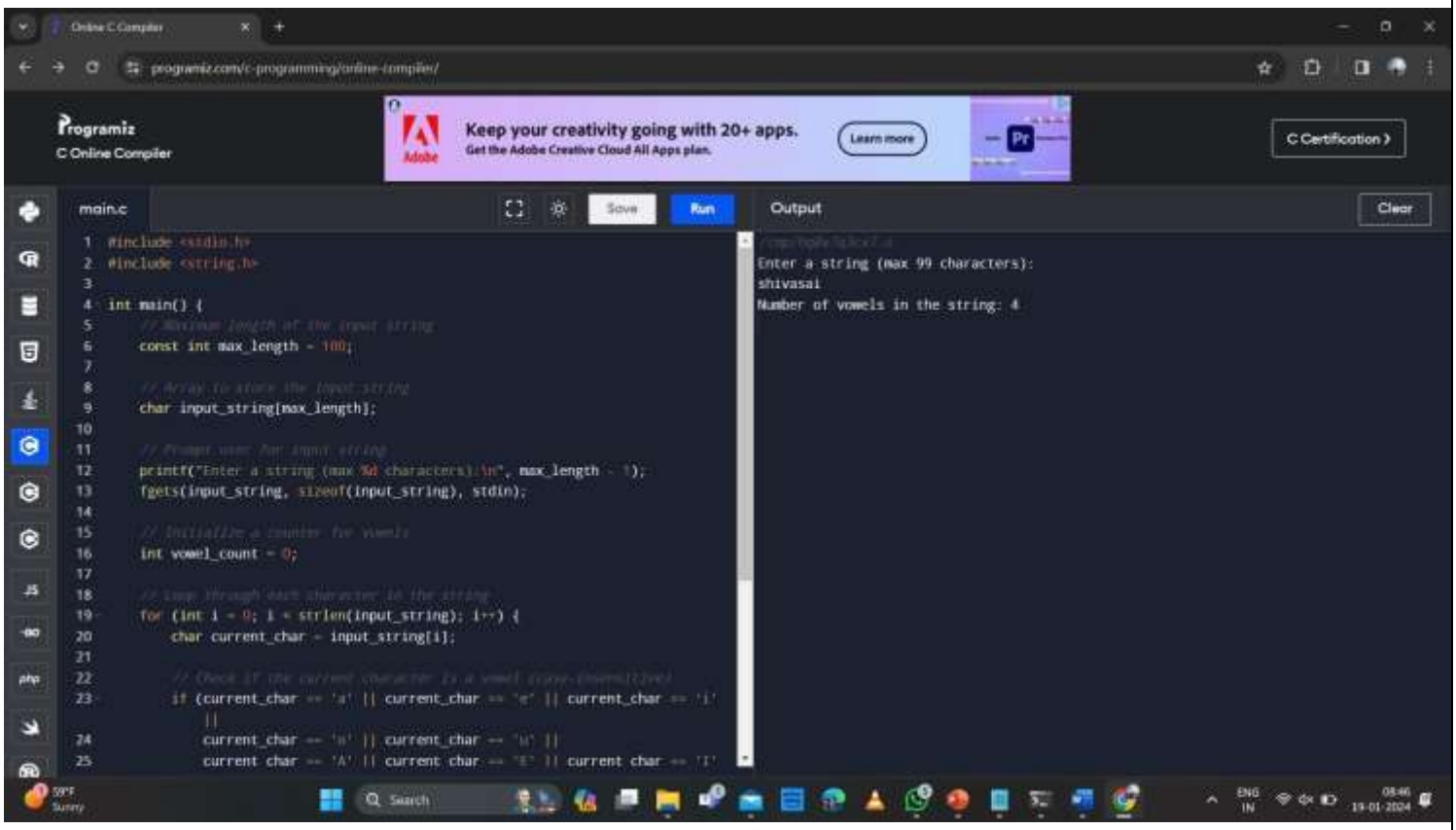
EXPERIMENT-2

AIM: Write a program to find out the vowels in a program.

CODE:

```
#include <stdio.h>
#include <string.h>

int main() {
   // Maximum length of the input string
   const int max_length = 100;

   // Array to store the input string
   char input_string[max_length];

   // Prompt user for input string
   printf("Enter a string (max %d characters):\n", max_length - 1);
   fgets(input_string, sizeof(input_string), stdin);

   // Initialize a counter for vowels
   int vowel_count = 0;

   // Loop through each character in the string
   for (int i = 0; i < strlen(input_string); i++) {
      char current_char = input_string[i];

      // Check if the current character is a vowel (case-insensitive)
      if (current_char == 'a' || current_char == 'e' || current_char == 'i' ||
         current_char == 'o' || current_char == 'u' ||
         current_char == 'A' || current_char == 'E' || current_char == 'I' ||
         current_char == 'O' || current_char == 'U') {
         vowel_count++;
      }
   }
```

```
    // Display the count of vowels
    printf("Number of vowels in the string: %d\n", vowel_count);

    return 0;
}
```



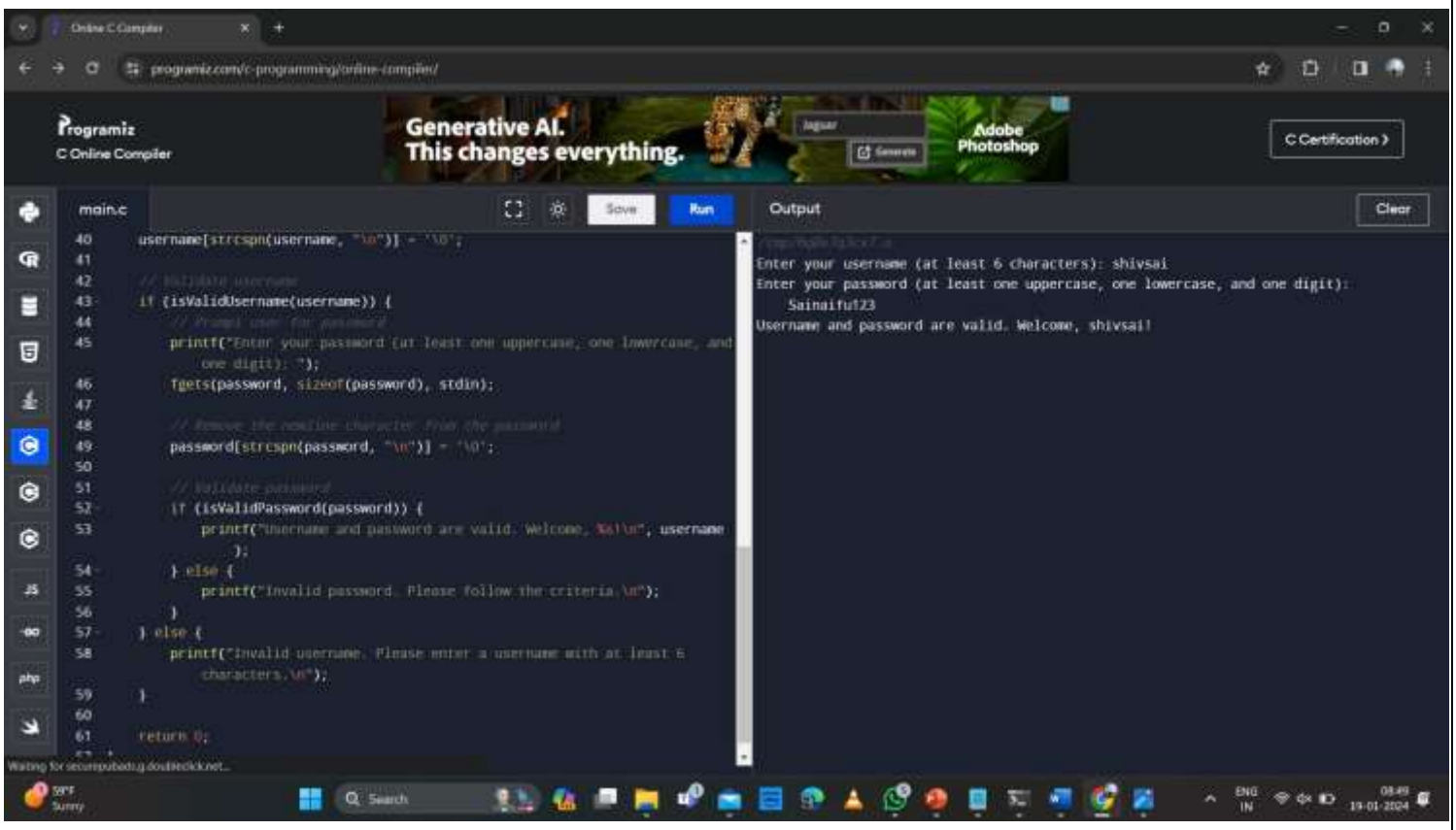
EXPERIMENT-3

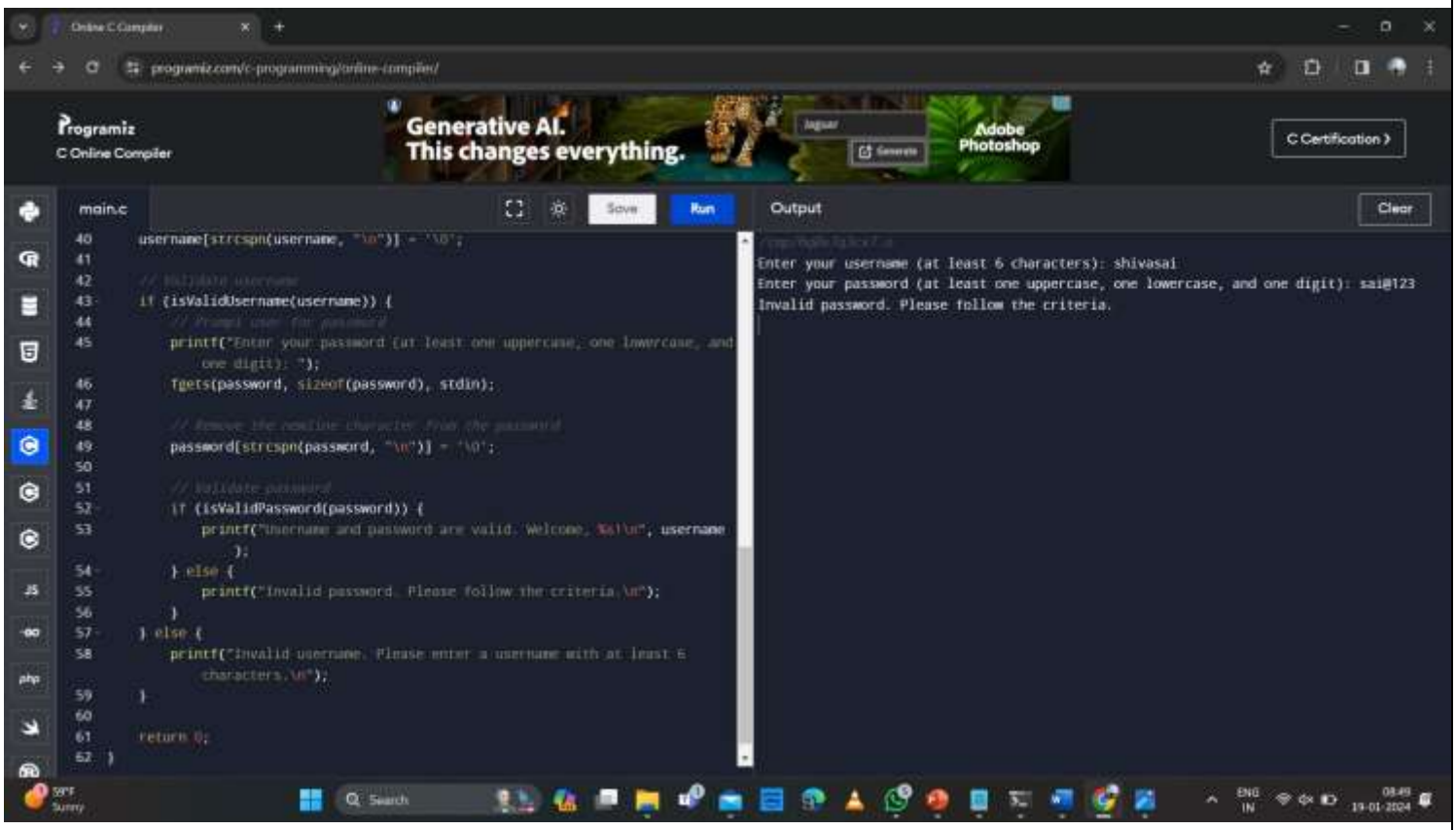
AIM: Write a program to check validation of user name and password.

CODE:

```
#include   <stdio.h>
#include  <ctype.h>
#include <string.h>

int isValidUsername(char *username) {
   // Check if the username is at least 6 characters long
   return (strlen(username) >= 6);
}

int isValidPassword(char *password) {
   int hasUpperCase = 0;
   int hasLowerCase = 0;
   int hasDigit = 0;

   // Check each character in the password
   for (int i = 0; i < strlen(password); i++) {
      if (isupper(password[i])) {
         hasUpperCase = 1;
      } else if (islower(password[i])) {
         hasLowerCase = 1;
      } else if (isdigit(password[i])) {
         hasDigit = 1;
      }
   }

   // Check if the password meets the criteria
   return (hasUpperCase && hasLowerCase && hasDigit);}
int main() {
   // Maximum length for username and password
```

```
    const int max_length = 50;

    char username[max_length];
    char password[max_length];

    // Prompt user for username
    printf("Enter your username (at least 6 characters): ");
    fgets(username, sizeof(username), stdin);

    // Remove the newline character from the username
    username[strcspn(username, "\n")] = '\0';

    // Validate username
    if (isValidUsername(username)) {
        // Prompt user for password
        printf("Enter your password (at least one uppercase, one lowercase, and one digit): ");
        fgets(password, sizeof(password), stdin);

        // Remove the newline character from the password
        password[strcspn(password, "\n")] = '\0';

        // Validate password
        if (isValidPassword(password)) {
            printf("Username and password are valid. Welcome, %s!\n", username);
        } else {
            printf("Invalid password. Please follow the criteria.\n");
        }
    } else {
        printf("Invalid username. Please enter a username with at least 6 characters.\n");
    }

    return 0;
}
```


Online C Compiler
programiz.com/c-programming/online-compiler/
Programiz
C Online Compiler
Generative AI.
This changes everything.
Adobe Photoshop
C Certification
main.c
Save
Run
Output
Clear
username[strcspn(username, "\n")] = '\0';
// Validate username
if (isValidUsername(username)) {
// Prompt user for password
printf("Enter your password (at least one uppercase, one lowercase, and one digit): ");
fgets(password, sizeof(password), stdin);
// Remove the newline character from the password
password[strcspn(password, "\n")] = '\0';
// Validate password
if (isValidPassword(password)) {
printf("Username and password are valid. Welcome, %s!\n", username);
} else {
printf("Invalid password. Please follow the criteria.\n");
}
} else {
printf("Invalid username. Please enter a username with at least 6 characters.\n");
}
return 0;
Enter your username (at least 6 characters): shivsai
Enter your password (at least one uppercase, one lowercase, and one digit):
Sainaifu123
Username and password are valid. Welcome, shivsai!


Online C Compiler
programiz.com/c-programming/online-compiler/
Programiz
C Online Compiler
Generative AI.
This changes everything.
Adobe Photoshop
C Certification
main.c
Save
Run
Output
Clear
username[strcspn(username, "\n")] = '\0';
// Validate username
if (isValidUsername(username)) {
// Prompt user for password
printf("Enter your password (at least one uppercase, one lowercase, and one digit): ");
fgets(password, sizeof(password), stdin);
// Remove the newline character from the password
password[strcspn(password, "\n")] = '\0';
// Validate password
if (isValidPassword(password)) {
printf("Username and password are valid. Welcome, %s!\n", username);
} else {
printf("Invalid password. Please follow the criteria.\n");
}
} else {
printf("Invalid username. Please enter a username with at least 6 characters.\n");
}
return 0;
}
Enter your username (at least 6 characters): shivasai
Enter your password (at least one uppercase, one lowercase, and one digit): sai@123
Invalid password. Please follow the criteria.

EXPERIMENT -04

Aim:- Program to implement Predictive Parsing LL(1) in C

Code:-

INPUT:-

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

char prol[10][10]={"E","E'","E'","T","T'","T'","F","F"};
char pror[10][10]={"TE'","+TE'","@","FT'","*FT'","@","(E)","%"};
char prod[10][10]={"E->TE'","E'->+TE'","T->FT'","T->*F","F->(E)","F->%"};
char first[10][10]={"(%","+@","(%","*@","(%"};
char follow[10][10]={"$)","$)","+$)","+$)","*+$)"}; char table[5][6][10];
numr(char c)
{
   switch(c)
   {
      case 'E': return 0;
      case 'T': return 1;
      case 'F': return 2;
      case '+': return 0;
      case '*': return 1;
      case '(': return 2;
      case ')': return 3;
      case '%': return 4;
      case '$': return 5;
   }
   return(2);
}
   void main()
   {
      int i,j,k;
      // clrscr(); for(i=0;i<5;i++) for(j=0;j<6;j++)
```

```
strcpy(table[i][j]," ");
 printf("\n predictive parsing LL(1):\n");
 for(i=0;i<10;i++) printf("%s\n",prod[i]);
 printf("\nPredictive parsing table is\n");
 fflush(stdin);
 for(i=0;i<10;i++)
 {
    k=strlen(first[i]);
    for(j=0;j<10;j++)
    if(first[i][j]!='@')
    strcpy(table[numr(prol[i][0])+1][numr(first[i][j])+1],prod[i]);
 }
 for(i=0;i<10;i++)
 {
    if(strlen(pror[i])==1)
    {
       if(pror[i][0]=='@')
       {
          k=strlen(follow[i]);
          for(j=0;j<k;j++)
             strcpy(table[numr(prol[i][0])+1][numr(follow[i][j])+1],prod[i]);
       }
    }
 }
 strcpy(table[0][0]," ");
 strcpy(table[0][1],"+");
 strcpy(table[0][2],"*");
 strcpy(table[0][3],"(");
 strcpy(table[0][4],")");
 strcpy(table[0][5],"%");
 strcpy(table[0][5],"$");
 strcpy(table[1][0],"E");
 strcpy(table[2][0],"T");
 strcpy(table[3][0],"F");
```

```
    printf("\n_____________________________________________\n");
    for(i=0;i<5;i++)
    for(j=0;j<6;j++)
    {
        printf("%-10s",table[i][j]);
        if(j==5)
      printf("\n-------------------------------------------------------------\n");
    }
    getch();
}
```

OUTPUT:-

```
 predictive parsing LL(1):
E->TE'
E'->+TE'
T->FT'
T->*F
F->(E)
F->%



Predictive parsing table is
-------------------------------------------------------------
            +           *           (           )           $
-------------------------------------------------------------
E           T->FT'                  T->FT'      T->FT'      T->FT'
-------------------------------------------------------------
T                       T->*F       F->%                    F->(E)
-------------------------------------------------------------
F
-------------------------------------------------------------
```

EXPERIMENT -05

Aim:- Program to implement Recursive Descent Parsing in C.

Code:-

INPUT:-

```
#include <stdio.h>
#include <string.h>

#define SUCCESS 1
#define FAILED 0

int E(), Edash(), T(), Tdash(), F();

const char *cursor;
char string[64];

int main()
{
    puts("Enter the string");
    // scanf("%s", string);
    sscanf("i+(i+i)*i", "%s", string);
    cursor = string;
    puts("");
    puts("Input      Action");
    puts("--------------------------------------");

    if (E() && *cursor == '\0') {
        puts("--------------------------------------");
        puts("String is successfully parsed");
        return 0;
    } else {
        puts("--------------------------------------");
        puts("Error in parsing String");
```

```
        return 1;

    }
}

int E()
{
    printf("%-16s E -> T E'\n", cursor);
    if (T()) {
        if (Edash())
            return SUCCESS;
        else
            return FAILED;
    } else
        return FAILED;
}

int Edash()
{
    if (*cursor == '+') {
        printf("%-16s E' -> + T E'\n", cursor);
        cursor++;
        if (T()) {
            if (Edash())
                return SUCCESS;
            else
                return FAILED;
        } else
            return FAILED;
    } else {
        printf("%-16s E' -> $\n", cursor);
        return SUCCESS;
    }
}

int T()
{
    printf("%-16s T -> F T'\n", cursor);
    if (F()) {
        if (Tdash())
            return SUCCESS;
```

```
        else

       return FAILED;
    } else
        return FAILED;
}

int Tdash()
{
    if (*cursor == '*') {
        printf("%-16s T' -> * F T'\n", cursor);
        cursor++;
        if (F()) {
            if (Tdash())
                return SUCCESS;
            else
                return FAILED;
        } else
            return FAILED;
    } else {
        printf("%-16s T' -> $\n", cursor);
        return SUCCESS;
    }
}

int F()
{
    if (*cursor == '(') {
        printf("%-16s F -> ( E )\n", cursor);
        cursor++;
        if (E()) {
            if (*cursor == ')') {
                cursor++;
                return SUCCESS;
            } else
                return FAILED;
        } else
            return FAILED;
    } else if (*cursor == 'i') {
```

```
        cursor++;
        printf("%-16s F ->i\n", cursor);
        return SUCCESS;

    } else
        return FAILED;
}
```

OUTPUT:-

```
Enter the string

Input        Action
--------------------------------
i+(i+i)*i          E -> T E'
i+(i+i)*i          T -> F T'
+(i+i)*i           F ->i
+(i+i)*i           T' -> $
+(i+i)*i           E' -> + T E'
(i+i)*i            T -> F T'
(i+i)*i            F -> ( E )
i+i)*i             E -> T E'
i+i)*i             T -> F T'
+i)*i              F ->i
+i)*i              T' -> $
+i)*i              E' -> + T E'
i)*i               T -> F T'
)*i                F ->i
)*i                T' -> $
)*i                E' -> $
*i                 T' -> * F T'
                   F ->i
                   T' -> $
                   E' -> $
--------------------------------
String is successfully parsed
```

EXPERIMENT -06

Aim:- Program to implement Operator Precedence Parsing in C.

Code:-

INPUT:-

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
   char stack[20],ip[20],opt[10][10][1],ter[10];
   int i,j,k,n,top=0,row,col;
   int len;
   for(i=0;i<10;i++)
   {
      stack[i]=NULL;ip[i]=NULL;
      for(j=0;j<10;j++)
      {
         opt[i][j][1]=NULL;
      }
   }
   printf("Enter the no.of terminals:");
   scanf("%d",&n);
   printf("\nEnter the terminals:");
   scanf("%s",ter);
   printf("\nEnter the table values:\n");
   for(i=0;i<n;i++)
   {
      for(j=0;j<n;j++)
      {
         printf("Enter the value for %c %c:",ter[i],ter[j]);
         scanf("%s",opt[i][j]);
      }

   }
   printf("\nOPERATOR PRECEDENCE TABLE:\n");
```

```
for(i=0;i<n;i++)
{
   printf("\t%c",ter[i]);
}
printf("\n_________________________");
printf("\n");
for(i=0;i<n;i++)
{
   printf("\n%c |",ter[i]);
   for(j=0;j<n;j++)
   {
      printf("\t%c",opt[i][j][0]);
   }
}
stack[top]='$';
printf("\n\nEnter the input string(append with $):");
scanf("%s",ip); i=0;
printf("\nSTACK\t\t\tINPUT STRING\t\t\tACTION\n");
printf("\n%s\t\t\t%s\t\t\t",stack,ip);
len=strlen(ip);
while(i<=len)
{
   for(k=0;k<n;k++)
   {
      if(stack[top]==ter[k])
      row=k;
      if(ip[i]==ter[k])
      col=k;
   }
   if((stack[top]=='$')&&(ip[i]=='$'))
   {
      printf("String is ACCEPTED");
      break;
   }
   else if((opt[row][col][0]=='<') ||(opt[row][col][0]=='='))
   {
      stack[++top]=opt[row][col][0];


      stack[++top]=ip[i];
      ip[i]=' ';
```

```
        printf("Shift %c",ip[i]); i++;
        }
        else
        {
          if(opt[row][col][0]=='>')
          {
            while(stack[top]!='<')
            {
              --top;
            }
            top=top-1;
            printf("Reduce");
          }
          else
          {
            printf("\nString is not accepted");
            break;
          }
        }
        printf("\n");
        printf("%s\t\t\t%s\t\t\t",stack,ip);
    }
    getch();
}
```

OUTPUT:-