Aayush Shah
19BCE245
1 October 2022

Compiler Construction

Practical 2

Implement Recursive Descent Parser algorithm for grammer

- **Code :**

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

char input[100];
int i,l;
int E();
int EP();
int T();
int TP();
int F();

int main()
{
     printf("Recursive descent parsing for the following
grammar : \n");
     printf("\tE->TE'\n");
     printf("\tE'->+TE'/@\n");
     printf("\tT->FT'\n");
     printf("\tT'->*FT'/@\n");
     printf("\tF->(E)/ID\n");
     printf("\nEnter the string to be checked : ");
     scanf("%[^\n]s",input);

     if(E()){
               if(input[i+1]=='\0')
                    printf("\nString is accepted");
```

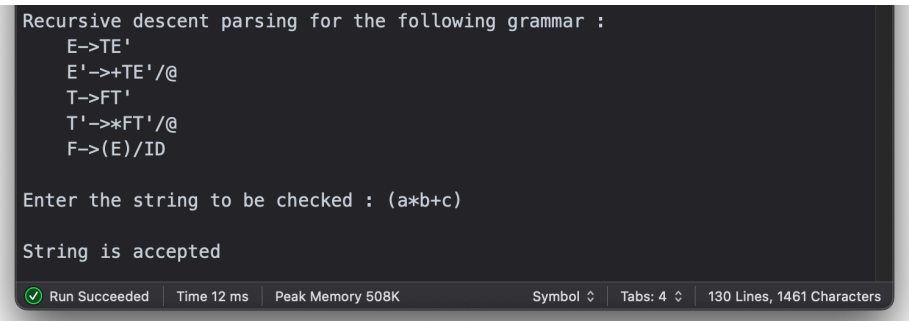
```
                else
                    printf("\nString is not accepted");
    }
    else
        printf("\nString not accepted");
}

int E(){
    if(T()){
        if(EP())
            return(1);

        else
            return(0);
    }
    else{
        return(0);
    }
}

int EP(){
    if(input[i]=='+')
        {
            i++;
            if(T())
                {
                    if(EP())
                        return(1);
                    else
                        return(0);
                }
            else
                return(0);
        }
    else
        return(1);
}

int T()
{
    if(F())
        {
            if(TP())
```

```
                    return(1);

               else
                    return(0);
          }
     else
          return(0);
}

int TP()
{
     if(input[i]=='*')
          {
               i++;
               if(F())
                    {
                         if(TP())
                              return(1);
                         else
                              return(0);
                    }
               else
                    return(0);
          }
     else
          return(1);
}

int F()
{
     if(input[i]=='(')
          {
               i++;
               if(E())
                    {
                         if(input[i]==')')
                              {
                                   i++;
                                   return(1);
                              }
                         else
                              return(0);
                    }
```

```
                else
                    return(0);
            }
        else if(input[i]>='a'&&input[i]<='z'||
input[i]>='A'&&input[i]<='Z')
            {
                i++;
                return(1);
            }
        else
            return(0);
}
```

output

- Accepted strings : *(a*b+c)* , *a+b* , *a+b+c*d*e*f*



- Not accepted strings : *a+*+*b+c)* , *(a*b)+c*d+)*