Practical 2

Name: Rajatkumar Patel
Roll No.: 18BCE191

Aim

To implement a Recursive Descent Parser Algorithm for the grammar.

Code

```
#include<bits/stdc++.h>
#define FAST ios_base::sync_with_stdio(false);cin.tie();cout.tie();
#define FILE_READ_IN freopen("input2.txt","r",stdin);
#define FILE_READ_OUT freopen("output.txt","w",stdout);
using namespace std;
typedef long long ll;

class Parser{
   private:
       string input;
       bool isError;
   public:
       int pos;
       Parser(string input){
           this->input=input;
           pos=0;
           isError=0;
       }
       void Error(){
           isError=1;
           pos=0;
       }
       void E(){
           if(isError) return;
           T();
           E2();
       }
       void T(){
```

```
            if(isError) return;
            if(input.length() > pos+2 && input.substr(pos,3)=="int"){
                pos+=3;
            }
            else if(input[pos]=='('){
                pos++;
                E();
                if(input[pos]==')')
                    pos++;
            }
            else Error();
        }
        void E2(){
            if(isError) return;
            if(pos < input.length() && input[pos]=='+'){
                pos++;
                E();
            }

        }

};


int main(){
  #ifndef ONLINE_JUDGE
     FILE_READ_IN
     FILE_READ_OUT
  #endif
   /*
        Grammar used:

        E ->TE'
        E'-> epsilon/+E
        T -> int/(E)
   */
   string input;
   cin>>input;
```

```
    cout<<"input: "<<input<<"\n";

    Parser obj(input);
    obj.E();
    bool res = (obj.pos == input.length());

    if(res) cout<<"Syntactically correct";
    else cout<<"Syntactically incorrect";
    return 0;
}
```

Output

output.txt

```
1   input: int+(int+int+(int))
2   Syntactically correct
```

output.txt

```
1   input: int+(int+int+(int+))
2   Syntactically incorrect
```

Conclusion

Implemented recursive descent parser, studied the necessary conditions that must be satisfied for proper working of recursive descent parser. Like, the grammar must not be left recursive, grammar should not have productions for a non terminal with common prefix.