# Practical: 6

**Aim:** Implement a program that specifies the type of the Grammar (Follow the Chomsky Hierarchy)

**Program:s**

```
/*
    author: mr_bhishm
    created: 25-08-2020 12:36:25
    "Make it work, make it right, make it fast."
                                                    – Kent Beck
*/
#include<bits/stdc++.h>
using namespace std;
#define debug(x) cout<<#x<<" "<<x<<endl

vector< string > RHS;
bool startNull = false;

bool check_type0(string lhs){
    int l_length = lhs.length();
    for(int i = 0 ;  i <  l_length ; i++){
        if(!isupper(lhs[i]) && !islower(lhs[i])){
            return false;
        }
    }
    return true;
}

bool check_type1(string lhs, string rhs,char start){
    int flag = 0;
    if(rhs.length() == 1 && rhs[0] == '~'){
        if(lhs.length() == 1 && lhs[0] == start){
            startNull = true;
            return true;
        }
        return false;
    }
    if(lhs.length() <= rhs.length()){
        return true;
    }else{
        return false;
    }
}

bool check_type2(string lhs){
    if(lhs.length() == 1 && isupper(lhs[0])){
        return true;
```

```
    }
    return false;
}

bool check_type3(string rhs){
    int r_length = rhs.length();
    if(r_length > 1){
        if(isupper(rhs[0])){
            // cheak for LL
            for(int i = 1 ; i < r_length ; i++){
                if(isupper(rhs[i])){
                    return false;
                }
            }
        }else if(isupper(rhs[r_length-1])){
            //check for RL
            for(int i = 0 ; i < r_length-1 ; i++){
                if(isupper(rhs[i])){
                    return false;
                }
            }
        }
    }else if(rhs[0] == '~'){
        startNull = true;
    }
    return true;
}

int main(){
    int n, ans = -1;
    cout<<"Enter the number of production functions: ";
    cin>>n;
    char S;
    cout<<"Enter Start Symbol: ";
    cin>>S;
    cout<<"Enter production rules: "<<endl;
    cout<<"[use CAPITAL for non-terminal and small case for terminal and ~ for
NULL]"<<endl;
    cout<<"Exmaple: S->aB without space"<<endl;
    vector< int > type;
    string input,left_side,right_side;
    string del = "->";
    while(n--){
        cin>>input;
        unsigned int index = input.find(del);
        // debug(index);
```

```
        left_side = input.substr(0,index);
        right_side = input.substr(index+2,input.length());
        // debug(left_side);
        // debug(right_side);
        RHS.push_back(right_side);
        if(check_type0(left_side)){
            ans = 0;
        }
        if(ans == 0){
            if(check_type1(left_side,right_side,S)){
                ans = 1;
            };
            if(right_side.length() == 1 && right_side[0] == '~' && left_side.length()
== 1 && isupper(left_side[0])){
                ans = 3;
            }
        }
        if(ans == 1){
            if(check_type2(left_side)){
                ans = 2;
            };
        }
        if(ans == 2){
            if(check_type3(right_side)){
                ans = 3;
            }
        }
        // debug(ans);
        type.push_back(ans);
    }
    sort(type.begin(), type.end());
    if((type[0]==1 || type[0]==3) && startNull==true){
        for(int i = 0;  i < RHS.size() ; i++){
            size_t found = RHS[i].find(S);
            if(found != -1){
                type[0]--;
                break;
            }
        }
    }
    cout<<"Grammar is of Type "<<type[0]<<endl;
}

//Reference: https://en.wikipedia.org/wiki/Chomsky_hierarchy
```

**Output:**

```
(base) PS D:\DLP_lab\Practical_6> g++ .\grammar_classifier.cpp
(base) PS D:\DLP_lab\Practical_6> .\a.exe
Enter the number of production functions: 4
Enter Start Symbol: S
Enter production rules:
[use CAPITAL for non-terminal and small case for terminal and ~ for NULL]
Exmaple: S->aB without space
S->ACaB
Bc->acB
CB->DB
aD->Db
Grammar is of Type 1
```

**Conclusion:** From this practical I have Learnt about Chomsky hierarchy for grammar classification and also learnt about how to implement program that identifies the type of grammar.