/* B5: An RR(1) parser is similar to an LL(1) parser except that it works from right-to-left, tracing out a top-down, *rightmost* derivation. The parser continuously consults the rightmost symbol of the derivation and the rightmost symbol of the input string that has not yet been matched to make its decisions about what production to use. The construction and operation of this parser is analogous to the LL(1) case, except that instead of having FIRST and FOLLOW sets we have LAST and PRECEDE sets, where LAST(A) is the set of terminals that could come at the *end* of a production of A (plus ε if A can derive the empty string), and PRECEDE(A) is the set of the terminals that could come *before* a production of A in some sentential form. Similarly, instead of having an end-of-input marker, we have a *beginning*-of-input marker, which we denote $. Write a program to implement RR(1) parsing. */

```
#include<bits/stdc++.h>
using namespace std;
vector< pair<char,string> > p;
vector<char> nt,term;
vector<char> ep;
map<char,int> m,mt;
int z,k;
vector<char> last[10],precede[10];
int visited[10];
vector<int> tt[20][20];

//find non-terminal symbols ie variables
bool is_nonterm(char c)
{
	if(c>='A' && c<='Z') { return 1; }
	else { return 0; }
}


void mark_epsilon()
{
	for(int i=0;i<p.size();i++)
	{
		if(p[i].second.size()==0) { ep.push_back(p[i].first);
}
	}
}

//find last of each non terminal
void last_util(char lhs,vector<char> &v)
{
	if(visited[mt[lhs]]==1) { return; }
	visited[mt[lhs]]=1;
	for(int i=0;i<p.size();i++)
	{
		if(lhs==p[i].first && p[i].second.size()>0)
		{
			int j=p[i].second.size()-1;
```

NIT Warangal

```
                while(j>=0)
                {
                        if(!is_nonterm(p[i].second[j]))
                        {

      if(find(v.begin(),v.end(),p[i].second[j])==v.end()) {
v.push_back(p[i].second[j]); }
                                break;
                        }
                        else
                        {
                                last_util(p[i].second[j],v);
                                if(
find(ep.begin(),ep.end(),p[i].second[j])==ep.end() ) { break; }
                                j--;
                        }
                }
                if(j==-1) { ep.push_back(lhs); }
            }
      }
}

//find last of each non terminal
void find_last()
{
      for(int i=0;i<nt.size();i++)
      {
            memset(visited,0,10*sizeof(int));
            char c = nt[i];
            last_util(c,last[mt[nt[i]]]);
      }
}

void precede_util(char lhs,vector<char> &v)
{
      if(visited[mt[lhs]]==1) { return; }
      visited[mt[lhs]]=1;
      if(lhs==p[0].first) {
if(find(v.begin(),v.end(),'$')==v.end()) { v.push_back('$'); }  }
      for(int i=0;i<p.size();i++)
      {
            int max_l=p[i].second.size();
            for(int j=0;j<p[i].second.size();j++)
            {
                  if(p[i].second[j]==lhs)
                  {
                        j--;
                        while(j>=0)
                        {
                              if(!is_nonterm(p[i].second[j]))
                              {
```

NIT Warangal

```
      if(find(v.begin(),v.end(),p[i].second[j])==v.end()) {
v.push_back(p[i].second[j]); }
                                    break;
                              }
                              else
                              {
                                    int index=mt[p[i].second[j]];
                                    for(int
l=0;l<last[index].size();l++)
                                    {
if(find(v.begin(),v.end(),last[index][l])==v.end()) {
v.push_back(last[index][l]); } }
                                    if(
find(ep.begin(),ep.end(),p[i].second[j])==ep.end() ) { break; }
                                    j--;
                              }
                        }
                        if(j==-1) { precede_util(p[i].first,v); }
                        break;
                  }
            }
      }
}

//find precede of each non terminal
void find_precede()
{
      for(int i=0;i<nt.size();i++)
      {
            memset(visited,0,10*sizeof(int));
            char c = nt[i];
            precede_util(c,precede[mt[nt[i]]]);
      }
}

void create_table()
{
      for(int i=0;i<p.size();i++)
      {
            int flag=1;
            if(p[i].second.size()!=0)
            {
                  flag=0;
                  for(int j=p[i].second.size()-1;j>=0;j--)
                  {
                        char lhs=p[i].first;
                        if(!is_nonterm(p[i].second[j]))
                        {
                              char c=p[i].second[j];
                              tt[mt[lhs]][m[c]].push_back(i);
                              break;
```


NIT Warangal

```
                        }
                        else
                        {
                              char c=p[i].second[j];
                              for(int k=0;k<last[mt[c]].size();k++)
                              {
                                    char ch=last[mt[c]][k];
                                    tt[mt[lhs]][m[ch]].push_back(i);
                              }
                              if(
find(ep.begin(),ep.end(),c)==ep.end() ) { break; }
                        }
                        if(j==0) { flag=1; }
                  }
            }
            if(flag==1)
            {
                  char lhs=p[i].first;
                  for(int j=0;j<precede[mt[lhs]].size();j++)
                  {
                        char c=precede[mt[lhs]][j];
                        tt[mt[lhs]][m[c]].push_back(i);
                  }
            }
      }
}

//print the parse table
void print_table()
{
      int flag=0;
      cout<<"   ";
      for(int i=0;i<term.size();i++)
      { cout<<term[i]<<" "; }
      cout<<"\n";
      for(int i=0;i<nt.size();i++)
      {
            cout<<nt[i]<<"  ";
            for(int j=0;j<term.size();j++)
            {
                  for(int k=0;k<tt[i][j].size();k++)
                  {
                        cout<<tt[i][j][k]+1<<" ";
                  }
                  if(tt[i][j].size()==0) { cout<<"  "; }
                  else if(tt[i][j].size()>1) { flag=1; }
            }
            cout<<"\n";
      }
      if(flag) { cout<<"Multiple Entries\n"; exit(0); }
}
```

NIT Warangal

```
//print productions
void print_productions()
{
      for(int i=0;i<p.size();i++)
      {
            cout<<i+1<<". "<<p[i].first<<" -> ";
            if(p[i].second.size()==0) { cout<<"epsilon\n"; }
            else { cout<<p[i].second<<"\n"; }
      }
}

void printstack(stack<char> s1)
{
      if(!s1.empty())
      {
            char c = s1.top(); s1.pop();
            printstack(s1);
            cout<<c;
      }
}

//parse given input using generated parse table
void parsing()
{
      string input;
      cout<<"\nEnter input string to be parsed : ";
      cin>>input;
      int ip=input.size()-1;
      stack<char> s;
      s.push('$'); s.push(p[0].first);
      while(1)
      {
            if(s.top()=='$' && input[ip]=='$') { break; }
            /////
            printstack(s);
            for(int i=1;i<20-s.size()-ip;i++) { cout<<" "; }
            for(int i=ip;i>=0;i--) { cout<<input[i]; }
            cout<<"\n";
            /////
            if(s.top()==input[ip]) { ip--; s.pop(); }
            else if(is_nonterm(s.top()))
            {
                  char x=s.top(),y=input[ip];
                  if(tt[mt[x]][m[y]].size()==0) { cout<<"Error in
parsing\n"; exit(0); }
                  else
                  {
                        s.pop();
                        int pn=tt[mt[x]][m[y]][0];
                        for(int i=0;i<p[pn].second.size();i++)
                        { s.push(p[pn].second[i]); }
                  }
```

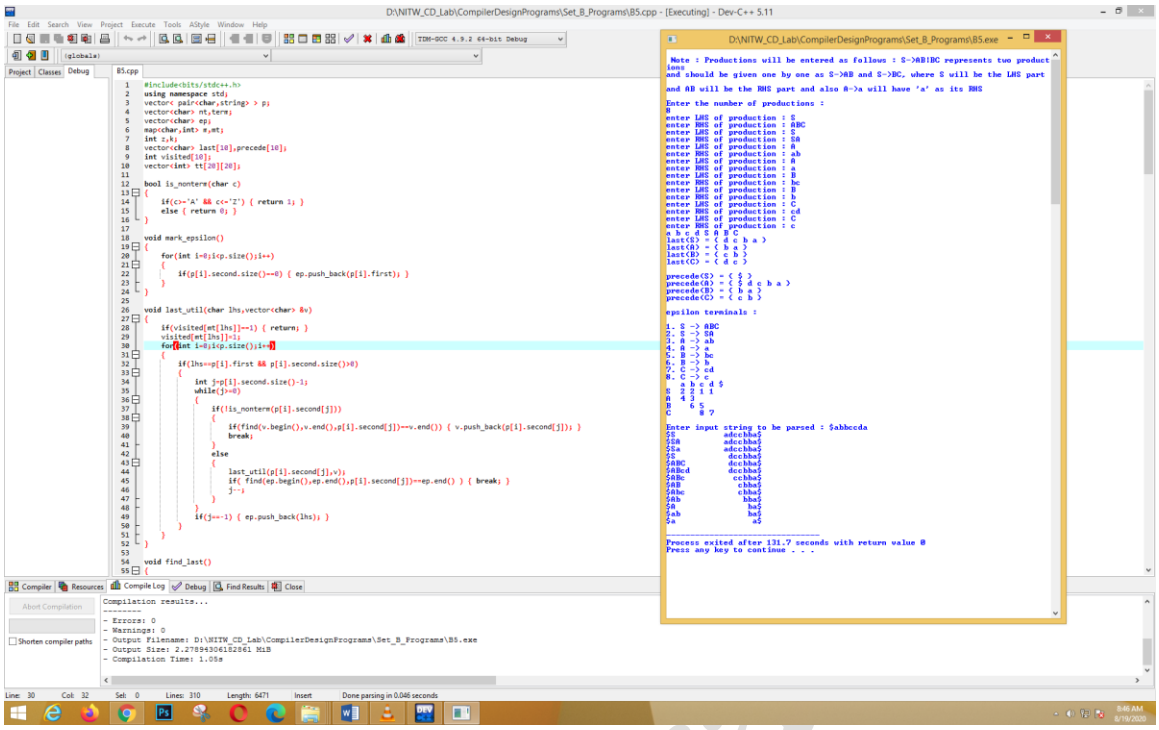
NIT Warangal

```
            }
            else { cout<<"Error in parsing\n"; exit(0); }
      }
}

int main()
{
      int n;
      char c;
      string s;
      cout<<"\n Note : Productions will be entered as follows :
S->AB|BC represents two productions \n";
    cout<<"and should be given one by one as S->AB and S->BC,
where S will be the LHS part \n";
    cout<<"and AB will be the RHS part and also A->a will have
'a' as its RHS \n\n";
      cout<<"Enter the number of productions : \n";
      cin>>n;
      //taking input
      for(int i=0;i<n;i++)
      {
            cout<<"enter LHS of production : ";
            cin>>c;
            cout<<"enter RHS of production : ";
            cin>>s;
            if(s=="%") { s=""; }
            p.push_back(make_pair(c,s));
      }

      // Finding terminals and non-terminals
      k=0;
      for(int i=0;i<p.size();i++)
      {
            int x=1;
            if(find(nt.begin(),nt.end(),p[i].first)==nt.end())
            {
                  nt.push_back(p[i].first);
            }
            for(int j=0;j<p[i].second.size();j++)
            {
                  if(!is_nonterm(p[i].second[j]) &&
find(term.begin(),term.end(),p[i].second[j])==term.end())
                  {
                        term.push_back(p[i].second[j]);
m[p[i].second[j]]=k++; cout<<p[i].second[j]<<" ";
                  }
            }
      }
      // add $ which represents end of input
      term.push_back('$');
      m['$']=k++;
```

NIT Warangal

```
	for(int i=0;i<nt.size();i++) { m[nt[i]]=k++; mt[nt[i]]=i;
cout<<nt[i]<<" "; }
	cout<<"\n";

	mark_epsilon();

	//find last of each non terminal
	find_last();
	for(int i=0;i<nt.size();i++)
	{
		cout<<"last("<<nt[i]<<") = { ";
		for(int j=0;j<last[mt[nt[i]]].size();j++)
		{
			cout<<last[mt[nt[i]]][j]<<" ";
		}
		cout<<"}\n";
	}

	cout<<endl;

	//find precede of each non terminal
	find_precede();
	for(int i=0;i<nt.size();i++)
	{
		cout<<"precede("<<nt[i]<<") = { ";
		for(int j=0;j<precede[mt[nt[i]]].size();j++)
		{
			cout<<precede[mt[nt[i]]][j]<<" ";
		}
		cout<<"}\n";
	}

	cout<<"\nepsilon terminals : "; for(int
i=0;i<ep.size();i++)  { cout<<ep[i]<<" "; }
	cout<<"\n\n";
	/////

	create_table();

	print_productions();
	print_table();

	parsing();

	return 0;
}
```

NIT Warangal



NIT Warangal