# Complier Design Lab Assignment

## CEN-692


Submitted By:
Habibur Rahman
17BCS071

Submitted To:
Mr. Shahzad Alam


**B-Tech VI$^{th}$ Semester ( Computer Science )**


Department of Computer Engineering,
Faculty of Engineering and Technology,
Jamia Millia Islamia ,
New Delhi


Session: 2019-20

| Index | |
|---|---|
| 1. | Write a program to find out the FIRST & FOLLOW values for a given Context Free Grammar. The program should read the C.F.G. from a file. |
| 2. | Write a program that verifies whether a given CFG is suitable for LL(1) parsing or not. If not then the program should convert the given CFG to a form which is suitable for the LL parsing. |
| 3. | Write a program in C to generate SLR parse table from CFG grammar. |
| 4. | Write a program to find the Leaders and Basic Blocks for a Three Address Code given through a file. |

**Question 1 ) Write a program to find out the FIRST & FOLLOW values for a given Context Free Grammar. The program should read the C.F.G. from a file.**

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Scanner;

public class firstFollow {
   public static void main(String[] args) throws FileNotFoundException {
      hrkhan();
      File file = new File("first.txt");
      Scanner s = new Scanner(file) ;
      int non_terminal = Integer.parseInt(s.nextLine());

      HashMap<String , String[]> map = new HashMap<>();
      String[] terminal = new String[non_terminal];
      for (int i = 0; i <non_terminal ; i++) {
          String str = s.nextLine();
          String[] str_split = str.split("-->" );
          String[] terminal_str = str_split[1].split("/");
         //System.out.println(Arrays.toString(terminal_str));
         map.put(str_split[0] , terminal_str) ;
         terminal[i] = str_split[0];
      }
      HashMap<String , ArrayList<String> >  first = new HashMap<>();
      HashMap<String , ArrayList<String> >  follow = new HashMap<>();
      System.out.println("First of All non-Terminal  ");
      first_func(map , first,terminal);
      System.out.println("Follow of all non-Terminal ");
      follow_func(map ,first ,terminal , follow);
   }

   public static void follow_func(HashMap<String, String[]> map,
HashMap<String, ArrayList<String>> first, String[] terminal,
HashMap<String, ArrayList<String>> follow) {
      for (int i = 0; i <terminal.length ; i++) {
         String str = terminal[i] ;
         int j = 0 ;
```

```
        ArrayList<String> follow_str = new ArrayList<>();
        if(i==0)
        {
           follow_str.add("$") ;
        }
        for (int k = 0; k <map.size(); k++) {
           String st[]  = map.get(terminal[k]);
           for (int l = 0; l <st.length ; l++) {
              if(st[l].contains(terminal[i]))
              {
                 int m = st[l].indexOf(terminal[i] , 0 );
                 if(m + 1< st[l].length())
                 {
                    if (map.containsKey(st[l].charAt(m+1) + ""))
                    {
                       if(!map.get(terminal[k]).equals(st[l].charAt(m+1) +
""))
                       {
                          ArrayList<String>  ft = first.get(st[l].charAt(m+1) +
"") ;
                          for (String sttt:ft) {
                             if (!follow_str.contains(sttt)) {
                                if (sttt.equals("#"))
                                {
                                   ArrayList<String>  ff =
follow.get(terminal[k]);
                                   for (String epl:ff) {
                                      if (!follow_str.contains(epl)) {
                                         follow_str.add(epl);
                                      }
                                   }
                                }
                                else
                                {
                                   follow_str.add(sttt);
                                }

                             }
                          }
                       }
                    }
                    else
```

```
                    {
                        follow_str.add(st[l].charAt(m+1) + "") ;
                    }
                }
                else
                {
                    if (map.containsKey(st[l].charAt(m) + ""))
                    {
                        if(!terminal[k].equals(st[l].charAt(m) + ""))
                        {
                            ArrayList<String>  ft = follow.get(terminal[k]);
                            for (String sttt:ft) {
                                if (!follow_str.contains(sttt)) {
                                    follow_str.add(sttt);
                                }
                            }
                        }
                    }
                }
            }
        }

        follow.put(terminal[i] , follow_str) ;
        System.out.println("follow ( " + terminal[i] + " )    =     " +
follow_str);
    }

}


   private static void hrkhan() {
      System.out.println("\nName :- Habiburrahman \nRollno 17BCS071
\nB-Tech Computer Engineering \n");
   }

   public static void first_func(HashMap<String, String[]> map,
HashMap<String, ArrayList<String>> first, String[] terminal){
      for (int i = 0; i <terminal.length ; i++) {
         String str = terminal[i];
         ArrayList<String> first_str = new ArrayList<>();
         String[] st = map.get(str);
```

```
            int j = 0 ;
            while(j < st.length)
            {
                if(map.containsKey(st[j].charAt(0) + ""))
                {
                    st = map.get(st[j].charAt(0) + "") ;
                }
                else
                {
                    first_str.add(st[j].charAt(0) + "");
                    j++ ;
                }
            }
            st = map.get(str);
            first.put(terminal[i] , first_str) ;
        }

        for (int i = 0; i <first.size() ; i++) {
            System.out.println("first( " +terminal[i] + " )   =       "  +
first.get(terminal[i]));
        }
    }
}
```

Output: -

```
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » more first.txt
5
E-->TR
R-->+TR/#
T-->FY
Y-->*FY/#
F-->(E)/i
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » javac firstFollow.java
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » java firstFollow

Name :- Habiburrahman
Rollno 17BCS071
B-Tech Computer Engineering

First of All non-Terminal
first( E )   =       [(, i]
first( R )   =       [+, #]
first( T )   =       [(, i]
first( Y )   =       [*, #]
first( F )   =       [(, i]
Follow of all non-Terminal
follow ( E )    =     [$, )]
follow ( R )    =     [$, )]
follow ( T )    =     [+, $, )]
follow ( Y )    =     [+, $, )]
follow ( F )    =     [*, +, $, )]
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› »
```

**Question 2 ) Write a program that verifies whether a given CFG is suitable for LL(1) parsing or not. If not then the program should convert the given CFG to a form which is suitable for the LL parsing.**

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Scanner;

public class checkLL1 {
   public static void main(String[] args) throws FileNotFoundException {
      hrkhan();
      File file = new File("cfg");
      Scanner s = new Scanner(file) ;
      HashMap<String , String[]> map = new HashMap<>();
      HashMap<String , String[]> leftRecursion = new HashMap<>();
      HashMap<String , String[]> leftFactoring = new HashMap<>();

      int i =0  ;
      while (s.hasNextLine())
      {
         String str = s.nextLine() ;
         String[] str_split = str.split("-->" );
         String[] terminal_str = str_split[1].split("/");
         //System.out.println(Arrays.toString(terminal_str));
         map.put(str_split[0] , terminal_str) ;
      }
      String[] non_terminal = new String[map.size()];
      for (String str: map.keySet()
           ) {
         non_terminal[i++] = str ;
      }

      boolean lr = checkLeftRecursion(map , non_terminal ,
leftRecursion);
      boolean lf = checkLeftfactoring(map , non_terminal , leftFactoring);
      if (lr && lf)
      {
         System.out.println("CFG is valid for LL1");
```

```
        }
        else
        {
            System.out.println("CFG is Not  valid for LL1");
            System.out.println("the CFG suitable for LL parsing ");
        }
        for (String st : map.keySet()) {
            System.out.print(st + " --> ");
            for (String ss : map.get(st)) {
                System.out.print(ss + " / ");
            }
            System.out.println();
        }
    }

    private static boolean checkLeftfactoring(HashMap<String, String[]>
map, String[] non_terminal, HashMap<String, String[]> leftFactoring) {
        int w = (int) 'W';

        HashMap<String , Boolean> del = new HashMap<>();
        for (String str: map.keySet()) {
            String stt = map.get(str)[0];
                for (int i = 1; i < map.get(str).length; i++) {
                      stt =  commonPrefix(stt , map.get(str)[i]);
                }
                String alpha = stt.length()!=map.get(str)[0].length() ? stt:"";
                if (alpha.length()!=0)
                {
                    del.put(str , true);
                    ArrayList<String> beeta = new ArrayList<>();
                    ArrayList<String> gamma = new ArrayList<>();
                    for (int i = 0; i <map.get(str).length ; i++) {
                        if (map.get(str)[i].contains(alpha))
                        {
                            int m = map.get(str)[i].indexOf(alpha , 0);
                            beeta.add(map.get(str)[i].substring(m+1));
                        }
                        else
                        {
                            gamma.add(map.get(str)[i]);
                        }
                    }
                    String[] beeta_ = new String[beeta.size()];
```

```
                String[] gamma_ = new String[gamma.size() + 1 ];
                gamma_[0] = alpha + (char)w ;
                for (int j = 1; j < gamma_.length; j++) {
                    gamma_[j] = gamma.get(j-1);
                }

                for (int j = 0; j <beeta.size() ; j++) {
                    beeta_[j] = beeta.get(j);
                }
                leftFactoring.put(str , gamma_ );
                leftFactoring.put((char)w+""  , beeta_);

                w++;
            }
        }

        for (String de:del.keySet()) {
            if (map.containsKey(de))
            {
                map.remove(de);
            }
        }
        for (String sr : leftFactoring.keySet()) {
            map.put(sr , leftFactoring.get(sr));
        }
            if (del.size()==0)
            {
                System.out.println("No Factoring (Non - Deterministic
Grammar) in the production After Left Recursion removing  ");
                return true;
            }
            else
            {
                return false ;
            }
    }
    static String commonPrefix(String str1, String str2) {
        String result = "";
        int n1 = str1.length(), n2 = str2.length();
        if (str1.charAt(0)!=str2.charAt(0))
            return str1 ;
        // Compare str1 and str2
        for (int i = 0, j = 0; i <= n1 - 1 && j <= n2 - 1; i++, j++) {
```

```
            if (str1.charAt(i) != str2.charAt(j)) {
                break;
            }
            result += str1.charAt(i);
        }

        return (result);
    }

    private static boolean checkLeftRecursion(HashMap<String, String[]>
map, String[] non_terminal , HashMap<String , String[]> leftRe) {
        int p = 80 ;
        HashMap<String , Boolean> del = new HashMap<>();
        for (String str: map.keySet()) {
            for (int i = 0; i < map.get(str).length ; i++) {
                if(map.get(str)[i].charAt(0) == str.charAt(0))
                {
                    System.out.println(" Left Recursion found in the production "
+str +"-->"  + Arrays.toString(map.get(str)));
                    del.put(str , true);
                    ArrayList<String> alpha = new ArrayList<>();
                    ArrayList<String> beeta = new ArrayList<>();
                    for (String ab:map.get(str)) {
                        if (ab.charAt(0) == str.charAt(0))
                        {
                            alpha.add(ab.substring(1));
                        }
                        else
                        {
                            beeta.add(ab);
                        }
                    }
                    String[] beeta_ = new String[beeta.size()];
                    String[] alpha_ = new String[alpha.size() + 1];
                    for (int j = 0; j <beeta.size() ; j++) {
                        beeta_[j] = beeta.get(j)+(char)p;
                    }

                    for (int j = 0; j <alpha.size() ; j++) {
                        alpha_[j] = alpha.get(j)+(char)p;
                    }
                    alpha_[alpha.size()] = "#" ;
                    leftRe.put(str,beeta_);
```

```
                leftRe.put((char)p + "" ,alpha_ ) ;
                p++;
            }
            break;
        }

    }
    for (String de:del.keySet()) {
        if (map.containsKey(de))
        {
            map.remove(de);
        }
    }
    for (String sr : leftRe.keySet()) {
        map.put(sr , leftRe.get(sr));
    }
    if (del.size()==0)
    {
        System.out.println("No Left Recursion in the production ");
        return true;
    }
    else
    {
        return false ;
    }

  }


  private static void hrkhan() {
      System.out.println("\nName :- Habiburrahman \nRollno 17BCS071
\nB-Tech Computer Engineering \n");
  }

}
```

Output:-

```
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » more cfg
A-->A+A/A*A/D
D-->1/2/3
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » javac checkLL1.java
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » java checkLL1

Name :- Habiburrahman
Rollno 17BCS071
B-Tech Computer Engineering

 Left Recursion found in the production A-->[A+A, A*A, D]
No Factoring (Non - Deterministic Grammar) in the production After Left Recursion removing
CFG is Not  valid for LL1
the CFG suitable for LL parsing
P --> +AP / *AP / # /
A --> DP /
D --> 1 / 2 / 3 /
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » more cfg
E-->TR
R-->+TR/#
T-->FY
Y-->*FY/#
F-->(E)/i
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » javac checkLL1.java
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » java checkLL1

Name :- Habiburrahman
Rollno 17BCS071
B-Tech Computer Engineering

No Left Recursion in the production
No Factoring (Non - Deterministic Grammar) in the production After Left Recursion removing
CFG is valid for LL1
R --> +TR / # /
T --> FY /
E --> TR /
F --> (E) / i /
Y --> *FY / # /
```

**Question 3 ) Write a program in C to generate SLR parse table from CFG grammar.**

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

int i2,j2,i,j,k,m2=0,n=0,o,p,ns=0,tn=0,rr=0,ch=0;
char
read[15][10],gl[15],gr[15][10],temp,templ[15],tempr[15][10],*ptr,temp2[5]
,dfa[15][15];
char variables[15]={'\0'}, terminals[15]={'\0'};
char slr[15][15][10]={'\0'};
char foll[10];
int len_var, len_ter;
struct states
{
	char lhs[15],rhs[15][10];
	int n;
}I[15];

void follow(char c);
void first(char c);

int compstruct(struct states s1,struct states s2)
{
	int t;
	if(s1.n!=s2.n)
		return 0;
	if( strcmp(s1.lhs,s2.lhs)!=0 )
		return 0;
	for(t=0;t<s1.n;t++)
		if( strcmp(s1.rhs[t],s2.rhs[t])!=0 )
			return 0;
	return 1;
}

void hrkhan()
{
	printf("Name: Habiburrahman\n");
	printf("Rollno : 17BCS071\n");
	printf("B-Tech Computer Enigeering \n");
}
void moreprod()
{
	int r,s,t,l1=0,rr1=0;
	char *ptr1,read1[15][10];

	for(r=0;r<I[ns].n;r++)
	{
```

```
                ptr1=strchr(l[ns].rhs[l1],'.');
                t=ptr1-l[ns].rhs[l1];
                if( t+1==strlen(l[ns].rhs[l1]) )
                {
                        l1++;
                        continue;
                }
                temp=l[ns].rhs[l1][t+1];
                l1++;
                for(s=0;s<rr1;s++)
                        if( temp==read1[s][0] )
                                break;
                if(s==rr1)
                {
                        read1[rr1][0]=temp;
                        rr1++;
                }
                else
                        continue;

                for(s=0;s<n;s++)
                {
                        if(gl[s]==temp)
                        {
                                l[ns].rhs[l[ns].n][0]='.';
                                l[ns].rhs[l[ns].n][1]='\0';
                                strcat(l[ns].rhs[l[ns].n],gr[s]);
                                l[ns].lhs[l[ns].n]=gl[s];
                                l[ns].lhs[l[ns].n+1]='\0';
                                l[ns].n++;
                        }
                }
        }
}

void canonical(int l)
{
        int t1;
        char read1[15][10],rr1=0,*ptr1;
        for(i=0;i<l[l].n;i++)
        {
                temp2[0]='.';
                ptr1=strchr(l[l].rhs[i],'.');
                t1=ptr1-l[l].rhs[i];
                if( t1+1==strlen(l[l].rhs[i]) )
                        continue;

                temp2[1]=l[l].rhs[i][t1+1];
                temp2[2]='\0';

                for(j=0;j<rr1;j++)
```

```
        if( strcmp(temp2,read1[j])==0 )
                break;
if(j==rr1)
{
        strcpy(read1[rr1],temp2);
        read1[rr1][2]='\0';
        rr1++;
}
else
        continue;

for(j=0;j<I[0].n;j++)
{
        ptr=strstr(I[l].rhs[j],temp2);
        if( ptr )
        {
                templ[tn]=I[l].lhs[j];
                templ[tn+1]='\0';
                strcpy(tempr[tn],I[l].rhs[j]);
                tn++;
        }
}

for(j=0;j<tn;j++)
{
        ptr=strchr(tempr[j],'.');
        p=ptr-tempr[j];
        tempr[j][p]=tempr[j][p+1];
        tempr[j][p+1]='.';
        I[ns].lhs[I[ns].n]=templ[j];
        I[ns].lhs[I[ns].n+1]='\0';
        strcpy(I[ns].rhs[I[ns].n],tempr[j]);
        I[ns].n++;
}

moreprod();
for(j=0;j<ns;j++)
{
        if( compstruct(I[ns],I[j])==1 )
        {
                I[ns].lhs[0]='\0';
                for(k=0;k<I[ns].n;k++)
                        I[ns].rhs[k][0]='\0';
                I[ns].n=0;
                dfa[l][j]=temp2[1];
                break;
        }
}
if(j<ns)
{
        tn=0;
```

```
                for(j=0;j<15;j++)
                {
                        templ[j]='\0';
                        tempr[j][0]='\0';
                }
                continue;
        }

        dfa[l][j]=temp2[1];
        printf("\n\nI%d :",ns);
        for(j=0;j<l[ns].n;j++)
                printf("\n\t%c -> %s",l[ns].lhs[j],l[ns].rhs[j]);

        //getch();
        ns++;
        tn=0;
        for(j=0;j<15;j++)
        {
                templ[j]='\0';
                tempr[j][0]='\0';
        }
    }
}

void extract_var_char(){
    for (i=0;i<n;i++){
        if (gl[i] >= 'A' && gl[i] <= 'Z'){
            if (!strchr(variables,gl[i]))
                sprintf(variables,"%s%c",variables,gl[i]);
        }

        for (j=0;j<strlen(gr[i]);j++){
            if (gr[i][j] < 'A' || gr[i][j] > 'Z'){
                if (!strchr(terminals,gr[i][j]))
                    sprintf(terminals,"%s%c",terminals,gr[i][j]);
            }
        }
    }
    strcat(terminals,"$");
    printf ("Variables : %s\nTerminals : %s\n",variables,terminals);
}

void display_slr(){
    printf("\n\t\tSLR(1) Table...\n\n\t");
    for (i=0;i<len_ter;i++)
        printf("%c\t",terminals[i]);
    for (i=0;i<len_var;i++)
        printf("%c\t",variables[i]);
    printf("\n");
    for (i=0;i<ns;i++){
        printf("I%d\t",i);
```

```
                for (j=0;j<len_var+len_ter;j++)
                        printf("%s\t",slr[i][j]);
                printf("\n");
        }
}

int  main()
{
        hrkhan();
        FILE *f;
        int l;
        //clrscr();

        for(i=0;i<15;i++)
        {
                l[i].n=0;
                l[i].lhs[0]='\0';
                l[i].rhs[0][0]='\0';
                dfa[i][0]='\0';
        }

        f=fopen("grammar.txt","r");
        while(!feof(f))
        {
                fscanf(f,"%c",&gl[n]);
                fscanf(f,"%s\n",gr[n]);
                n++;
        }

        printf("\tGiven Grammar...\n");
        for(i=0;i<n;i++)
                printf("\t\t%c -> %s\n",gl[i],gr[i]);

        l[0].lhs[0]='Z';
        strcpy(l[0].rhs[0],".S");
        l[0].n++;
        l=0;
        for(i=0;i<n;i++)
        {
                temp=l[0].rhs[l][1];
                l++;
                for(j=0;j<rr;j++)
                        if( temp==read[j][0] )
                                break;
                if(j==rr)
                {
                        read[rr][0]=temp;
                        rr++;
                }
                else
                        continue;
```

```
		for(j=0;j<n;j++)
		{
			if(gl[j]==temp)
			{
				I[0].rhs[I[0].n][0]='.';
				strcat(I[0].rhs[I[0].n],gr[j]);
				I[0].lhs[I[0].n]=gl[j];
				I[0].n++;
			}
		}
	}
	ns++;

	printf("\n\tCanonicals...\n");
	printf("\nI%d :\n",ns-1);
	for(i=0;i<I[0].n;i++)
		printf("\t%c -> %s\n",I[0].lhs[i],I[0].rhs[i]);

	for(l=0;l<ns;l++){
		canonical(l);
	}

	//////////////Construction of SLR(1) Table/////////////////
	int t,tempo;

	extract_var_char();
	len_var = strlen(variables);
	len_ter = strlen(terminals);
	int columns = len_var + len_ter;

	for (i=0;i<ns;i++){

		for (j=0;j<ns;j++){
			if (dfa[i][j] != '\0'){
				if (strchr(terminals,dfa[i][j])){
					sprintf(slr[i][strchr(terminals,dfa[i][j])-
terminals],"s%d",j);
				}
				else if (strchr(variables,dfa[i][j])){

	sprintf(slr[i][len_ter+strchr(variables,dfa[i][j])-variables],"%d",j);
				}
			}
		}

		for (j=0;j<I[i].n;j++){
			int temp_ind = strlen(I[i].rhs[j])-1; // to make the 1st
state that end with "S." accept.

			if (I[i].rhs[j][temp_ind] == '.'){
				if (I[i].rhs[j][temp_ind-1] == 'S' && i==1)
```

```
                                        sprintf(slr[i][strchr(terminals,'$')-
terminals],"accept");
                                else{
                                        follow(I[i].lhs[j]);

                                        int ha,prod_num,tempo_store;



                                        for (ha=0;gl[ha]!='\0';ha++){
                                                if
(strncmp(gr[ha],I[i].rhs[j],temp_ind)==0)
                                                        prod_num = ha + 1;
                                        }

                                        for (ha=0; ha<m2 ; ha++){
                                                tempo_store =
strchr(terminals,foll[ha])-terminals;
                                                if (strlen(slr[i][tempo_store]) > 0){
                                                        if (slr[i][tempo_store][0] == 's')
                                                                printf("\nSR-Conflict
occurred....\n");
                                                        else
                                                                printf("\nRR-Conflict
occurred....\n");
                                                }
                                                else

        sprintf(slr[i][tempo_store],"r%d",prod_num);
                                        }
                                        m2=0;
                                }
                        }
                }
        }

        display_slr();
 return 0 ;
}

void follow(char c){
 if(gl[0]==c)
  foll[m2++]='$';
 for(i2=0;i2<n;i2++){
    for(j2=0;j2<strlen(gr[i2]);j2++){
      if(gr[i2][j2]==c){
        if(gr[i2][j2+1]!='\0')
          first(gr[i2][j2+1]);

        if(gr[i2][j2+1]=='\0'&&c!=gl[i2])
          follow(gl[i2]);
```

```
      }
    }
  }
}

void first(char c){
  int k;
  if(!(isupper(c)))
    foll[m2++]=c;
  for(k=0;k<n;k++){
    if(gl[k]==c){
      if(gr[k][0]=='$')
        follow(gl[i2]);
      else if(islower(gr[k][0]))
        foll[m2++]=gr[k][0];
      else
        first(gr[k][0]);
    }
  }
}
```

Output:-

```
[Habiburrahman-khans-MacBook-Pro :: ~/desktop/slr » gcc slrtable.c
[Habiburrahman-khans-MacBook-Pro :: ~/desktop/slr » ./a.out
Name: Habiburrahman
Rollno : 17BCS071
B-Tech Computer Enigeering
        Given Grammar...
                S -> AA
                A -> aA
                A -> b

        Canonicals...

I0 :
        Z -> .S
        S -> .AA
        A -> .aA
        A -> .b


I1 :
        Z -> S.

I2 :
        S -> A.A
        A -> .aA
        A -> .b

I3 :
        A -> a.A
        A -> .aA
        A -> .b

I4 :
        A -> b.

I5 :
        S -> AA.

I6 :
        A -> aA.Variables : SA
Terminals : ab$

                SLR(1) Table...

        a       b       $       S       A
I0      s3      s4              1       2
I1                      accept
I2      s3      s4                      5
I3      s3      s4                      6
I4      r3      r3      r3
I5                      r1
I6      r2      r2      r2
Habiburrahman-khans-MacBook-Pro :: ~/desktop/slr »
```

**Question 4 ) Write a program to find the Leaders and Basic Blocks for a Three Address Code given through a file.**

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class leaderBasicBlocks {
   public static void main(String[] args) throws FileNotFoundException {
      File file = new File("leaderbasicblockinput") ;
      hrkhan();
      Scanner s = new Scanner(file);
      HashMap<Integer , String>  input = new HashMap<>();
      HashMap<Integer , String>  leader = new HashMap<>();
      HashMap<Integer , ArrayList<String>>  basic = new HashMap<>();
      int i  =1 ;
      while (s.hasNextLine())
      {
         String str = s.nextLine() ;
         input.put(i++ , str);
      }
      leader_func(input, leader);
      basicblocksfunc(input , leader , basic);
   }

   private static void basicblocksfunc(HashMap<Integer, String> input,
HashMap<Integer, String> leader, HashMap<Integer, ArrayList<String>>
basic) {
            Set<Integer> linenos = leader.keySet() ;
            int[]   lineno = new int[linenos.size()];
            int j = 0 ;
      for (int i : linenos) {
         lineno[j] = i ;
         j++;
      }
      Arrays.sort(lineno);
      j = 1;
      for (int i = 0; i <lineno.length-1; i++) {
         ArrayList<String>  basicstr = new ArrayList<>();
         for (int k = lineno[i]; k <lineno[i+1]; k++) {
            basicstr.add(input.get(k));
         }
         basic.put(j++ , basicstr);
```

```
        }
        ArrayList<String>  basicstr = new ArrayList<>();
        for (int k = lineno[lineno.length -1 ]; k < input.size()+1; k++) {
            basicstr.add(input.get(k));
        }
        basic.put(j++ , basicstr);
        for (int i : basic.keySet()) {
            ArrayList<String>  showoutput =  basic.get(i) ;
            System.out.println("Basic blocks  no " + i + "    ");
            for (String st:
                 showoutput) {
                System.out.println(st);
            }
        }
    }


    private static void leader_func(HashMap<Integer, String> input,
HashMap<Integer, String> leader) {
        leader.put(1 , input.get(1)) ;
        for (int i = 1; i <=input.size(); i++) {
            if (input.get(i).contains("goto"))
            {
                int m  = input.get(i).indexOf("(");
                int n   = input.get(i).indexOf(")");
                int line_no = Integer.parseInt(input.get(i).substring(m+1 , n));
                leader.put(line_no , input.get(line_no));
                if (input.containsKey(i+1) )
                {
                    leader.put(i+1, input.get(i+1));
                }
            }
        }
        for (int i : leader.keySet()) {
            System.out.println("leaders line no " + i + "    " + leader.get(i));
        }
    }
    private static void hrkhan() {
        System.out.println("\nName :- Habiburrahman \nRollno 17BCS071
\nB-Tech Computer Engineering \n");
    }
}
```

Output :-

```
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » more leaderbasicblockinput
i=1
j=1
t1 = 10 * i
t2 = t1 + j
t3 = 8 * t2
t4 = t3 - 88
a[t4] = 0.0
j = j + 1
if j <= goto (3)
i = i + 1
if i <= 10 goto (2)
i = 1
t5 = i - 1
t6 = 88 * t5
a[t6] = 1.0
i = i + 1
if i <= 10 goto (13)
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » javac leaderBasicBlocks.java
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› » java leaderBasicBlocks

Name :- Habiburrahman
Rollno 17BCS071
B-Tech Computer Engineering

leaders line no 1    i=1
leaders line no 2    j=1
leaders line no 3    t1 = 10 * i
leaders line no 10    i = i + 1
leaders line no 12    i = 1
leaders line no 13    t5 = i - 1
Basic blocks  no 1
i=1
Basic blocks  no 2
j=1
Basic blocks  no 3
t1 = 10 * i
t2 = t1 + j
t3 = 8 * t2
t4 = t3 - 88
a[t4] = 0.0
j = j + 1
if j <= goto (3)
Basic blocks  no 4
i = i + 1
if i <= 10 goto (2)
Basic blocks  no 5
i = 1
Basic blocks  no 6
t5 = i - 1
t6 = 88 * t5
a[t6] = 1.0
i = i + 1
if i <= 10 goto (13)
Habiburrahman-khans-MacBook-Pro :: hrkhan/sem6/compilerDesign ‹master*› »
```