Dr BR Ambedkar National Institute of Technology
Jalandhar-144011, Punjab, India.

JULY-DEC, 2023

Department of Computer Science & Engineering

System and Compiler Design Laboratory

(CSPC-421)

Submitted to:
Dr. Somesula Manoj Kumar
(Assistant Professor)
Department of Computer
Science & Engineering

Submitted by:
Divyansh Siddharth
20103054
A (G2)
CSE-7th Sem

Question 2nd solution

```c
#include <stdio.h>
#include <string.h>
#define TSIZE 128

int table[100][TSIZE];

char terminal[TSIZE];

char nonterminal[26];

struct product
{
    char str[100];
    int len;
} pro[20];

int no_pro;
char first[26][TSIZE];
char follow[26][TSIZE];

char first_rhs[100][TSIZE];

int isNT(char c)
{
    return c >= 'A' && c <= 'Z';
}

void readFromFile()
{
    FILE *fptr;
    fptr = fopen("text.txt", "r");
    char buffer[255];
    int i;
    int j;
    while (fgets(buffer, sizeof(buffer), fptr))
    {
        printf("%s", buffer);
        j = 0;
        nonterminal[buffer[0] - 'A'] = 1;
        for (i = 0; i < strlen(buffer) - 1; ++i)
        {
            if (buffer[i] == '|')
            {
                ++no_pro;
                pro[no_pro - 1].str[j] = '\0';
                pro[no_pro - 1].len = j;
```

```
                pro[no_pro].str[0] = pro[no_pro - 1].str[0];
                pro[no_pro].str[1] = pro[no_pro - 1].str[1];
                pro[no_pro].str[2] = pro[no_pro - 1].str[2];
                j = 3;
            }
            else
            {
                pro[no_pro].str[j] = buffer[i];
                ++j;
                if (!isNT(buffer[i]) && buffer[i] != '-' && buffer[i] != '>')
                {
                    terminal[buffer[i]] = 1;
                }
            }
        }
        pro[no_pro].len = j;
        ++no_pro;
    }
}
void add_FIRST_A_to_FOLLOW_B(char A, char B)
{
    int i;
    for (i = 0; i < TSIZE; ++i)
    {
        if (i != '^')
            follow[B - 'A'][i] = follow[B - 'A'][i] || first[A - 'A'][i];
    }
}
void add_FOLLOW_A_to_FOLLOW_B(char A, char B)
{
    int i;
    for (i = 0; i < TSIZE; ++i)
    {
        if (i != '^')
            follow[B - 'A'][i] = follow[B - 'A'][i] || follow[A - 'A'][i];
    }
}
void FOLLOW()
{
    int t = 0;
    int i, j, k, x;
    while (t++ < no_pro)
    {
        for (k = 0; k < 26; ++k)
        {
            if (!nonterminal[k])
                continue;
            char nt = k + 'A';
```

```
            for (i = 0; i < no_pro; ++i)
            {
                for (j = 3; j < pro[i].len; ++j)
                {
                    if (nt == pro[i].str[j])
                    {
                        for (x = j + 1; x < pro[i].len; ++x)
                        {
                            char sc = pro[i].str[x];
                            if (isNT(sc))
                            {
                                add_FIRST_A_to_FOLLOW_B(sc, nt);
                                if (first[sc - 'A']['^'])
                                    continue;
                            }
                            else
                            {
                                follow[nt - 'A'][sc] = 1;
                            }
                            break;
                        }
                        if (x == pro[i].len)
                            add_FOLLOW_A_to_FOLLOW_B(pro[i].str[0], nt);
                    }
                }
            }
        }
    }
}
void add_FIRST_A_to_FIRST_B(char A, char B)
{
    int i;
    for (i = 0; i < TSIZE; ++i)
    {
        if (i != '^')
        {
            first[B - 'A'][i] = first[A - 'A'][i] || first[B - 'A'][i];
        }
    }
}
void FIRST()
{
    int i, j;
    int t = 0;
    while (t < no_pro)
    {
        for (i = 0; i < no_pro; ++i)
        {
```

```
            for (j = 3; j < pro[i].len; ++j)
            {
                char sc = pro[i].str[j];
                if (isNT(sc))
                {
                    add_FIRST_A_to_FIRST_B(sc, pro[i].str[0]);
                    if (first[sc - 'A']['^'])
                        continue;
                }
                else
                {
                    first[pro[i].str[0] - 'A'][sc] = 1;
                }
                break;
            }
            if (j == pro[i].len)
                first[pro[i].str[0] - 'A']['^'] = 1;
        }
        ++t;
    }
}
void add_FIRST_A_to_FIRST_RHS__B(char A, int B)
{
    int i;
    for (i = 0; i < TSIZE; ++i)
    {
        if (i != '^')
            first_rhs[B][i] = first[A - 'A'][i] || first_rhs[B][i];
    }
}

void FIRST_RHS()
{
    int i, j;
    int t = 0;
    while (t < no_pro)
    {
        for (i = 0; i < no_pro; ++i)
        {
            for (j = 3; j < pro[i].len; ++j)
            {
                char sc = pro[i].str[j];
                if (isNT(sc))
                {
                    add_FIRST_A_to_FIRST_RHS__B(sc, i);
                    if (first[sc - 'A']['^'])
                        continue;
                }
```

```
                else
                {
                    first_rhs[i][sc] = 1;
                }
                break;
            }
            if (j == pro[i].len)
                first_rhs[i]['^'] = 1;
        }
        ++t;
    }
}

void parseString(char str[])
{
    int stack[100], top = -1;
    stack[++top] = 0;
    int i = 0;

    printf("\nParsing Steps:\n");

    while (1)
    {
        int currentState = stack[top];
        char currentSymbol = str[i];

        if (!terminal[currentSymbol])
        {
            printf("Error: Invalid input symbol '%c'\n", currentSymbol);
            return;
        }

        int productionIndex = table[currentState][currentSymbol];
        if (productionIndex > 0)
        {
            printf("Applied %s\n", pro[productionIndex - 1].str);

            for (int j = pro[productionIndex - 1].len - 1; j > 2; j--)
            {
                if (pro[productionIndex - 1].str[j] != '^')
                {
                    top--;
                }
            }

            stack[++top] = table[stack[top - 1]][pro[productionIndex -
1].str[0]];
        }
```

```
        else
        {
            printf("Error: No valid production for state %d and symbol
'%c'\n", currentState, currentSymbol);
            return;
        }

        if (currentSymbol == '\0')
        {
            printf("String Accepted!\n");
            return;
        }

        i++;
    }
}

int main()
{
    readFromFile();
    follow[pro[0].str[0] - 'A']['$'] = 1;
    FIRST();
    FOLLOW();
    FIRST_RHS();
    int i, j, k;

    printf("\n");
    for (i = 0; i < no_pro; ++i)
    {
        if (i == 0 || (pro[i - 1].str[0] != pro[i].str[0]))
        {
            char c = pro[i].str[0];
            printf("FIRST OF %c: ", c);
            for (j = 0; j < TSIZE; ++j)
            {
                if (first[c - 'A'][j])
                {
                    printf("%c ", j);
                }
            }
            printf("\n");
        }
    }

    printf("\n");
    for (i = 0; i < no_pro; ++i)
    {
        if (i == 0 || (pro[i - 1].str[0] != pro[i].str[0]))
```

```
        {
            char c = pro[i].str[0];
            printf("FOLLOW OF %c: ", c);
            for (j = 0; j < TSIZE; ++j)
            {
                if (follow[c - 'A'][j])
                {
                    printf("%c ", j);
                }
            }
            printf("\n");
        }
    }

    printf("\n");
    for (i = 0; i < no_pro; ++i)
    {
        printf("FIRST OF %s: ", pro[i].str);
        for (j = 0; j < TSIZE; ++j)
        {
            if (first_rhs[i][j])
            {
                printf("%c ", j);
            }
        }
        printf("\n");
    }

    terminal['$'] = 1;

    terminal['^'] = 0;

    printf("\n");
    printf("\n\t**************** LL(1) PARSING TABLE ******************\n");
    printf("\t-------------------------------------------------------\n");
    printf("%-10s", "");
    for (i = 0; i < TSIZE; ++i)
    {
        if (terminal[i])
            printf("%-10c", i);
    }
    printf("\n");
    int p = 0;
    for (i = 0; i < no_pro; ++i)
    {
        if (i != 0 && (pro[i].str[0] != pro[i - 1].str[0]))
            p = p + 1;
        for (j = 0; j < TSIZE; ++j)
```

```
        {
            if (first_rhs[i][j] && j != '^')
            {
                table[p][j] = i + 1;
            }
            else if (first_rhs[i]['^'])
            {
                for (k = 0; k < TSIZE; ++k)
                {
                    if (follow[pro[i].str[0] - 'A'][k])
                    {
                        table[p][k] = i + 1;
                    }
                }
            }
        }
    }
    k = 0;
    for (i = 0; i < no_pro; ++i)
    {
        if (i == 0 || (pro[i - 1].str[0] != pro[i].str[0]))
        {
            printf("%-10c", pro[i].str[0]);
            for (j = 0; j < TSIZE; ++j)
            {
                if (table[k][j])
                {
                    printf("%-10s", pro[table[k][j] - 1].str);
                }
                else if (terminal[j])
                {
                    printf("%-10s", "");
                }
            }
            ++k;
            printf("\n");
        }
    }

      char input[100];

    printf("Enter the input string: ");
    scanf("%s", input);

    parseString(input);

    return 0;
}
```

3rd solution

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

void convertToWords(const char *num);

%}

%option noyywrap

%%

[1-9][0-9]*    { convertToWords(yytext); }
0              { printf("zero\n"); }
.              { /* Ignore other characters */ }

%%

void convertToWords(const char *num) {
    // Define arrays for words representing numbers
    const char *units[] = {"", "one", "two", "three", "four", "five", "six",
"seven", "eight", "nine"};
    const char *teens[] = {"", "eleven", "twelve", "thirteen", "fourteen",
"fifteen", "sixteen", "seventeen", "eighteen", "nineteen"};
    const char *tens[] = {"", "ten", "twenty", "thirty", "forty", "fifty",
"sixty", "seventy", "eighty", "ninety"};

    int t[4];

    int numvalue = atoi(num);

    t[1]=numvalue%1000;

    t[2]=numvalue%100;

    t[3]=numvalue%10;

    t[0]=numvalue-t[1];
    t[1]=t[1]-t[2];
    t[2]=t[2]-t[3];

    if(t[0]/1000!=0 && t[1]/100!=0)
     {cout<<units[t[0]/1000]<<"thousand"<<units[t[1]/100]<<"hundred"<<" ";
        if(t[2]+t[3]<20)
```

```
            cout<<teens[t[2]+t[3]-10];
        else
            cout<<tens[t[2]/10]<<units[t[3]];}
    else  if(t[0]/1000==0 && t[1]/100!=0)
     {cout<<units[t[1]/100]<<"hundred"<<" ";
        if(t[2]+t[3]<20)
            cout<<teens[t[2]+t[3]-10];
        else
            cout<<tens[t[2]/10]<<units[t[3]];}
            else  if(t[0]/1000==0 && t[1]/100==0)
     {
        if(t[2]+t[3]<20)
            cout<<teens[t[2]+t[3]-10];
        else
            cout<<tens[t[2]/10]<<units[t[3]];}









}

int main() {
    yylex();
    return 0;
}
```

4th solution

```
%{
#include <stdio.h>
%}
```

```
%%
GO_TO    { printf("GOTO"); }
.|\n    { printf("%s", yytext); }
%%

int main() {
    yylex();
    return 0;
}
```