Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Course No. : CSE4130
Course Name : Formal Languages and Compilers Lab

Assignment No. : 04

Submitted By:

Name : Shahriar Hasan Chowdhury
ID No. : 17 01 04 030
Session : Spring - 2020
Section : A (A2)

Assignment Question:

Suppose, a given C source program has been scanned, filtered, lexically analyzed and tokenized as that were done in earlier sessions. In addition, line numbers have been assigned to the source code lines for generating proper error messages. As the first step to Syntax Analysis, we now perform detection of simple syntax errors like duplication of tokens except parentheses or braces, unbalanced braces or parentheses problem, unmatched ‘else’ problem, etc. Duplicate identifier declarations must also be detected with the help of the Symbol Table.

Code:

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

int max_word = 10000;
int each_word_len = 20;

// ========================================== assignment 1 starts =======
========================================
FILE *p1,*p2, *p3;
void add_line()
{
    FILE *f1 = fopen("without_comment.txt", "r");
    FILE *f2 = fopen("with_line.txt", "w");
    int line_num = 1;
    char line[20];

    itoa(line_num, line, 10);
    fputs("line ", f2);
    //fputs(line, f2);
    //fputc(' ', f2);
    line_num++;
    char c;
    if(!f1)
    {
        printf("\nFile not found");
    }
    else
```

```
    {
        while((c = fgetc(f1)) != EOF)
        {
            if((c == '\n'))
            {

                itoa(line_num, line, 10);
                fputc('\n', f2);
                fputs("line ", f2);
                //fputs(line, f2);
                //fputc(' ', f2);

                line_num++;
            }
            else
            {
                fputc(c,f2);
            }

        }
    }
    fclose(f1);
    fclose(f2);

}

void single_line_comment()
{
    char temp1;
    while((temp1=fgetc(p1))!=EOF)
    {
        if(temp1 == '\n')
            return;
    }
}
```

```
void multi_line_comment()
{
    char temp1, temp2;
    while((temp1=fgetc(p1))!=EOF)
    {
        if(temp1 == '*')
        {
            temp2 = fgetc(p1);
            if(temp2=='/')
                return;
        }
    }
}
void remove_comments()
{
    char c;
    p1 = fopen("input.txt", "r");
    p2 = fopen("without_comment.txt", "w");

    if(!p1)
    {
        printf("\nFile can't be opened");
    }
    else
    {
        while((c=fgetc(p1)) != EOF)
        {
            if((c =='/'))
            {
                if((c=fgetc(p1)) == '/')
                {
                    single_line_comment();
                }
                else if(c == '*')
                {
                    multi_line_comment();
```

```
                }
            }
            else
            {
                fputc(c,p2);
            }
        }
    }
    fclose(p1);
    fclose(p2);
}

void remove_space_newline()
{
//    p1 = fopen("without_comment.txt", "r");
    p1 = fopen("with_line.txt", "r");
    p2 = fopen("output_assignment_1.txt", "w");

    if(!p1)
    {
        printf("\nFile can't be opened");
    }
    else
    {
        char a,b;
        while((a=fgetc(p1)) != EOF)
        {
            if((a==' '))
            {
                char temp = a;
                while((a=fgetc(p1))!=EOF)
                {
                    if(a!= ' ' && (a!='\n')&& (a!='\t'))
                    {
                        fputc(temp,p2);
```

```
                        break;
                    }
                }
            }
            if((a!=' ') && (a != '\n') && (a!='\t'))
            {
                fputc(a,p2);
            }
        }
    }
    fclose(p1);
    fclose(p2);
}

void print_output()
{
    printf("\n\n Assignment 1 Output\n\n");
    char c;
    p2 = fopen("output_assignment_1.txt", "r");
    while((c=fgetc(p2))!=EOF)
    {
        printf("%c",c);
    }
    fclose(p2);
//    remove("without_comment.txt");
}

void call_assignment1()
{
    remove_comments();
    add_line();
    remove_space_newline();
    ///print_output();
}
// ------------------------------------------------- Assignment 1 Ends --
----------------------------------
```

```
// ================================================== Assignment 2 Starts
====================================
int LINE_NUM = 1;

int isSeperator(char c)
{
    if(c == ';' || c == '\''  || c =='\"' || c == ',')
        return 1;
    return 0;
}

int isOperator(char c)
{
    if(c == '+' || c == '-' || c == '*' || c == '/' ||
        c == '=' || c == '>' || c == '<' || c == '!' ||
         c == '%' || c == '&' || c == '^' || c == '~')
         return 1;
    return 0;
}

int isDoubleOperator(char c1, char c2)
{
    if((c1 == '+' && c2 == '+') || (c1 == '-' && c2 == '-
') || (c1 == '+' && c2 == '=') ||
        (c1 == '-
' && c2 == '=') || (c1 == '=' && c2 == '=') || (c1 == '>' && c2 == '=') ||
        (c1 == '<' && c2 == '=') || (c1 == '&' && c2 == '&') || (c1 == '|'
&& c2 == '|') ||
        (c1 == '>' && c2 == '>') || (c1 == '<' && c2 == '<') || (c1 == '?'
&& c2 == ':'))
            return 1;
    return 0;
}

int isParenthesis(char c)
```

```
{
    if(c == '(' || c == ')' || c == '{' || c == '}' || c == '[' || c == ']'
)
        return 1;
    return 0;
}

char keywords[32][32] =
{
    "auto", "const", "double", "float", "int", "short", "struct", "unsigned
",
    "break", "continue", "else", "for", "long", "signed", "switch", "void",
    "case", "default", "enum", "goto", "register", "sizeof", "typedef", "vo
latile",
    "char", "do", "extern", "if", "return", "static", "union", "while"
};



int isKeyword(char arr[])
{
    for(int i=0; i<32; i++)
    {
        if(strcmp(keywords[i], arr) == 0)
            return 1;
    }
    return 0;
}

int isIdentifier(char arr[])
{
    if (!(isalpha(arr[0]) || arr[0]== '_'))
       return 0;

    for (int i = 1; i < strlen(arr); i++)
    {
```

```
        if (!(isalpha(arr[i]) || arr[i] == '_' || isdigit(arr[i])))
            return 0;
    }
    return 1;
}


int isRealNumber(char arr[])
{
    int NumOfPoint = 0;
    int digit = 1;
    for(int i = 0; i < strlen(arr); i++)
    {
        if(isdigit(arr[i]))
            digit = 1;
        else if(arr[i] == '.')
            NumOfPoint++;
        else
        {
            digit = 0;
            break;
        }
    }

    if(arr[strlen(arr)-1] == '.')
        return 0;

    if(digit == 1 && NumOfPoint <= 1)
        return 1;
    return 0;
}


void separateLexemes(char input[])
{
    FILE *f = fopen("step2input.txt", "w");
```

```
int l = strlen(input);
for(int i=0;i<l;i++)
{
    if(isSeperator(input[i]))
    {
        fputc(' ', f);
        fputc(input[i], f);
        fputc(' ', f);
    }
    else if(isParenthesis(input[i]))
    {

        fputc(' ', f);
        fputc(input[i], f);
        fputc(' ', f);

    }
    else if(isDoubleOperator(input[i], input[i+1]))
    {
        fputc(' ', f);
        fputc(input[i], f);
        i++;
        fputc(input[i], f);
        fputc(' ', f);
    }
    else if(isOperator(input[i]))
    {
        fputc(' ', f);
        fputc(input[i], f);
        fputc(' ', f);
    }
    else{
        fputc(input[i], f);
    }
}
fclose(f);
```

```
}


void categorizeLexemes(char input2[])
{
    FILE *f2 = fopen("output_assignment_2.txt", "w");
    int endIndex = 0;
    char id[100];
    for(int i=0;i<strlen(input2);i++)
    {
        if(isParenthesis(input2[i]))
        {
            //printf("[par %c] ", input2[i]);
            fputs("[par ", f2);
            fputc(input2[i], f2);
            fputs("] ", f2);
        }
        else if(isSeperator(input2[i]))
        {
            //printf("[sep %c] ",input2[i]);
            fputs("[sep ", f2);
            fputc(input2[i], f2);
            fputs("] ", f2);
        }
        else if(isOperator(input2[i]))
        {
            if(isOperator(input2[i+1]))
            {
//                printf("[op %c%c] ", input2[i++], input2[i]);
                //printf("[op %c",input2[i]);
                fputs("[op ", f2);
                fputc(input2[i], f2);
                i++;
                //printf("%c] ",input2[i]);
                fputc(input2[i], f2);
                fputs("] ", f2);
```

```
        }
        else{
            //rintf("[op %c] ",input2[i]);
            fputs("[op ", f2);
            fputc(input2[i], f2);
            fputs("] ", f2);
        }

    }
    else
    {
         if(input2[i] != ' ')
         {
             id[endIndex] = input2[i];
             endIndex++;
         }
         else
         {
             id[endIndex] = '\0';
             endIndex = 0;
             if(strlen(id) >= 1)
             {
                 if(isKeyword(id)){
                    //printf("[kw %s] ",id);
                    fputs("[kw ", f2);
                    fputs(id, f2);
                    fputs("] ", f2);
                 }
                 else if(isLine(id)){
                    char line[20];
                    itoa(LINE_NUM, line, 10);

                    //printf("[line %d]", LINE_NUM);
                    fputs("[line ", f2);
                    fputs(line, f2);
                    fputs("] ", f2);
```

```
                        LINE_NUM++;

                    }
                    else if(isIdentifier(id)){
                        //printf("[id %s] ",id);
                        fputs("[id ", f2);
                        fputs(id, f2);
                        fputs("] ", f2);
                    }
                    else if(isRealNumber(id)){
                        //printf("[num %s] ",id);
                        fputs("[num ", f2);
                        fputs(id, f2);
                        fputs("] ", f2);
                    }
                    else{
                        //printf("[unkn %s] ",id);
                        fputs("[unkn ", f2);
                        fputs(id, f2);
                        fputs("] ", f2);
                    }

                }
            }
        }
    }
    //printf("\n\n");
    fclose(f2);
}

void call_assignment2()
{
    char input[100000];
    char input2[100000];

    FILE *f1 = fopen("output_assignment_1.txt", "r");
```

```
    if(f1) fgets(input, 100000, f1);
    else printf("File Not Found");
    fclose(f1);

    separateLexemes(input);

    FILE *f2 = fopen("step2input.txt", "r");
    if(f2) fgets(input2, 100000, f2);
    else printf("File Not Found");
    fclose(f2);

    categorizeLexemes(input2);
}
/// ------------------------------------------------- Assignment 2 Ends -
-----------------------------------


char line[10][10] = {"line"};

int isLine(char arr[])
{
    if(strcmp(line[0], arr) == 0)
        return 1;
    return 0;
}

void check_unbalanced_parenthesis(char word[max_word][each_word_len], int l
en_word){
    /// check if unbalanced parenthesis

    int pr1_start = 0;
    int pr1_end = 0;
    int pr2_start = 0;
    int pr2_end = 0;
    int is_found_missmatch = 0;
    int line_no = 0;
```

```
for(int i=0;i<len_word;i+=1)
{
    if(strcmp(word[i], "line") == 0)
    {
        line_no = word[i+1];
    }

    if(strcmp(word[i], "(") == 0){
        pr1_start++;
    }
    else if(strcmp(word[i], ")") == 0){
        pr1_end++;
    }
    else if(strcmp(word[i], "{") == 0){
        pr2_start++;
    }
    else if(strcmp(word[i], "}") == 0){
        pr2_end++;
    }

    if(pr1_end > pr1_start){
        printf("Misplaced ')' at line: %s\n", line_no);
        is_found_missmatch = 1;
        break;
    }
    if(pr2_end > pr2_start){
        printf("Misplaced '}' at line: %s\n", line_no);
        is_found_missmatch = 1;
        break;
    }


}

if(pr1_end != pr1_start && is_found_missmatch == 0){
    if(pr1_start > pr1_end)
```

```
            printf("Misplaced ')' at line: %s\n", line_no);
        else
            printf("Misplaced '(' at line: %s\n", line_no);
    }
    if(pr2_end != pr2_start && is_found_missmatch == 0){
        if(pr2_start > pr2_end)
            printf("Misplaced '{' at line: %s\n", line_no);
        else
            printf("Misplaced '}' at line: %s\n", line_no);
    }

    /// end checking unbalanced parenthesis

}

void check_unbalanced_else(char word[max_word][each_word_len], int len_word
)
{
    /// check unbalanced else
    int count_if = 0;
    int count_else = 0;
    int is_found_missmatch_else = 0;
    int line_no = 0;
    for(int i=0;i<len_word;i+=1)
    {
        if(strcmp(word[i], "line") == 0)
        {
            line_no = word[i+1];
        }

        if(strcmp(word[i], "if") == 0){
            count_if++;
        }
        else if(strcmp(word[i], "else") == 0){
            count_else++;
        }
```

```
        if(count_else > count_if){
            printf("Unmatched 'else' at line: %s\n", line_no);
            break;
        }
    }

    /// end checking unbalanced else

}

void check_duplicate_token(char word[max_word][each_word_len], int len_word
){
    char crw[20];
    int crwi = 0;
    char nrw[20];
    int nrwi  = 0;
    int line_no = 0;
    int err_code = 1;
    int previous_error_line = 0;


    /// check if duplicate token
    for(int i=0;i<len_word;i+=1)
    {
        if(strcmp(word[i], "line") == 0)
        {
            line_no = word[i+1];
        }

        strcpy(crw, word[i]);
        crwi = i;
        strcpy(nrw, word[i+2]);
        nrwi = i+2;

//        printf("==========\n");
```

```
//         printf("crw is : %s\n", crw);
//         printf("nrw is : %s\n", nrw);

        if(!(strcmp(crw, "line") == 0) &&
           !(strcmp(crw, "par") == 0) &&
           !(strcmp(crw, "{") == 0) &&
           !(strcmp(crw, "}") == 0) &&
           !(strcmp(crw, "(") == 0) &&
           !(strcmp(crw, ")") == 0))
        {
            if(strcmp(crw, nrw) == 0){
                if((strcmp(crw, "kw") == 0) && (strcmp(nrw, "kw") == 0)){
                    if((strcmp(word[crwi+1], "else") == 0) && (strcmp(word[
nrwi+1], "return") == 0)){
                        continue;
                    }
                }
                if(previous_error_line != line_no){
                    previous_error_line = line_no;
                    printf("duplicate token at line %s\n", line_no);
                }

            }

        }

    }
}

void call_assignment4()
{
    char str[100000];
    char output_step1[100000];
    char output_with_space[100000];
    FILE *f1 = fopen("output_assignment_2.txt", "r");
    if(f1)
```

```
    fgets(str, 100000, f1);
else
    printf("File Not Found");
fclose(f1);

//puts("\n\n");
//puts(str);

/// keep word with space
int j = 0;
for(int i=0; i<strlen(str); i++)
{
    if(str[i] == '[')
        continue;
    else if(str[i] == ']')
        continue;
    else
    {
        output_with_space[j] = str[i];
        j++;
    }
}

/// split tokens into words
char word[max_word][each_word_len];
int k=0;
char* piece = strtok(output_with_space, " ");
int len_word=0;
while(piece != NULL)
{
    strcpy(word[k], piece);
    k++;
    piece = strtok(NULL, " ");
    len_word++;
}
strcpy(word[k], "0--end--0");
```

```
    /// main work

    check_duplicate_token(word, len_word);
    check_unbalanced_parenthesis(word, len_word);
    check_unbalanced_else(word, len_word);

}

int main(void)
{
    call_assignment1();
    call_assignment2();
    call_assignment4();
    return 0;
}
```