Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

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

Final Project

Submitted By:

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

Assignment Question:

Write a C program to read a source code in C from the console and print a report on the given source code. The report will contain the following:

1. The source code with line number
2. Any kind of error covered in Session 1 to Session 5
3. Header Files
4. Identifiers
5. Keywords
6. Functions

Sample Input	Sample Output

#include<stdio.h>
/*Program Starts*/
float func(int x))
{
int y;;
for(y; x,){} //for loop
return 1.5;
}
int main()
{
int a=1;
if(c)
a=2;
else
a=0;
return 0;
}
	Source code with line number:
	1. #include<stdio.h>
	2. /*Program Starts*/
	3. float func(int x))
	4. {
	5. int y;;
	6. for(y; x,){} //for loop
	7. return 1.5;
	8. ....
	9. ....
	Errors:
	At line number 3: Misplaced Parenthesis
	At line number 5: ....
	At line number 6: ....
	....
	Header Files: stdio.h
	Identifiers: func, x, y, ...
	Keywords: float, int, ...
	Functions: func, main

Code:

```
/*****************************************************************/
/* CSE 4130 : Spring 2020 Final Project ID: 170104030 */
/*****************************************************************/


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

int max_word = 10000;
int each_word_len = 20;
void categorizeLexemes(char input2[]);
int isLine(char arr[]);
```

```c
// ============================================ assignment 1 starts =
===============================================
FILE *p1,*p2, *p3,*output_file;

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);
    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);

                line_num++;
            }
            else
            {
                fputc(c,f2);
            }
        }
    }
    fclose(f1);
    fclose(f2);
}

void print_source_code()
{
    output_file = fopen("report_170104030.txt", "a");
    FILE *f1 = fopen("input.c", "r");
    int line_num = 1;
    char line[20];

    itoa(line_num, line, 10);

    fprintf(output_file, "%-3s. ", line);
    printf("%-3s. ", line);
    line_num++;
    char c;
    if(!f1)
    {
        printf("\ninput file File not found");
    }
    else
    {
        while((c = fgetc(f1)) != EOF)
        {
            if((c == '\n'))
            {
                fprintf(output_file, "\n");
                printf("\n");
                itoa(line_num, line, 10);


                fprintf(output_file, "%-3s. ", line);
                printf("%-3s. ", line);
                line_num++;
            }
            else
            {
                fprintf(output_file, "%c", c);
                printf("%c", c);
            }

        }
    }
    fclose(f1);
    fclose(output_file);
}

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 take_input()
{
    char c;
    printf("Paste a C code to check: \n\n");
    p1 = fopen("input.c", "w");

    while(scanf("%c", &c) != EOF){
        fputc(c,p1);
    }
    fclose(p1);
}

void remove_comments()
{
    char c;
    p1 = fopen("input.c", "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("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 call_assignment1()
{
    output_file = fopen("report_170104030.txt", "w");
    fprintf(output_file, "Source code with line number:\n\n");
    printf("\n\n===================================\n\n");
    printf("Source code with line number:\n\n");
    fclose(output_file);
    print_source_code();

    printf("\n");
    remove_comments();
    add_line();
    remove_space_newline();


}
// --------------------------------------------------
- Assignment 1 Ends -----------------------------------

// =================================================== Assignment 2 S
tarts =====================================
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", "un
signed",
    "break", "continue", "else", "for", "long", "signed", "switch", "
void",
    "case", "default", "enum", "goto", "register", "sizeof", "typedef
", "volatile",
    "char", "do", "extern", "if", "else", "return", "static", "union"
, "while"
};



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

```c
    }
    return 0;
}

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

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

int isHeaderInclude(char arr[])
{
    if(strcmp(arr, "#include") == 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]))
        {
            fputs("[par ", f2);
            fputc(input2[i], f2);
            fputs("] ", f2);
        }
        else if(isSeperator(input2[i]))
        {
            fputs("[sep ", f2);
            fputc(input2[i], f2);
            fputs("] ", f2);
        }
        else if(isOperator(input2[i]))
        {
            if(isOperator(input2[i+1]))
            {
                fputs("[op ", f2);
                fputc(input2[i], f2);
                i++;
                fputc(input2[i], f2);
                fputs("] ", f2);
            }
            else{
                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)){
                        fputs("[kw ", f2);
                        fputs(id, f2);
                        fputs("] ", f2);
                     }
                     else if(isLine(id)){
                        char line[20];
                        itoa(LINE_NUM, line, 10);

                        fputs("[line ", f2);
                        fputs(line, f2);
                        fputs("] ", f2);
                        LINE_NUM++;

                     }
                     else if(isHeaderInclude(id)){
                        fputs("[header ", f2);
                        fputs(id, f2);
                        fputs("] ", f2);

                     }

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

                }
            }
        }
    }
    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 -------------------------------------



/// Making of symble Table ###########################################
###########



int sl = 1;
int totalSymbleInTable = 0;
char maybeAssignmentOP = '!';
int number_of_value_in_symble_table;
struct symble_table
```

```c
{
    int sl_no;
    char name[20];
    char id_type[10];
    char data_type[20];
    char scope[10];
    char value[100];
};

struct symble_table st[100];

void keep_identifier(char input_step1[],char output_step1[])
{
    int j = 0;
    for(int i=0; i<strlen(input_step1); i++)
    {
        if(input_step1[i]=='k' && input_step1[i+1]=='w')
        {
            i+=3;
        }
        else if(input_step1[i]=='o' && input_step1[i+1]=='p')
        {
            i+=3;
        }
        else if(input_step1[i]=='n' && input_step1[i+1]=='u' && input
_step1[i+2]=='m')
        {
            i+=4;
        }
        else if(input_step1[i]=='s' && input_step1[i+1]=='e' && input
_step1[i+2]=='p')
        {
            i+=4;
        }
        else if(input_step1[i]=='b' && input_step1[i+1]=='r' && input
_step1[i+2]=='c')
        {
            i+=4;
        }
        else if(input_step1[i]=='p' && input_step1[i+1]=='a' && input
_step1[i+2]=='r')
        {
            i+=4;
        }
        output_step1[j++]=input_step1[i];
    }
```

```
}


int isFunc(char arr[])
{
    if(strcmp(arr, "(") == 0)
        return 1;
    return 0;
}
int k = 1;
void update(int sl_no, char value[], char isAssign)
{
    if(isAssign == '='){
        strcpy(st[sl_no].value, value);
    }

}
int checkValidDataTye(char data_type[])
{
    if((strcmp(data_type, "int") == 0) || (strcmp(data_type, "float")
 == 0) || (strcmp(data_type, "double") == 0) || (strcmp(data_type, "l
ong") == 0) || (strcmp(data_type, "short") == 0))
        return 1;
    else
        return 0;
}

int search(char name[], char id_type[], char scope[])
{
    for(int i=0;i<=totalSymbleInTable;i++)
    {
        if((strcmp(st[i].name, name) == 0) && (strcmp(st[i].scope, sc
ope) == 0)){
            return st[i].sl_no;
        }
    }
    return -1;
}

void insert(int sl_no, char name[], char id_type[], char data_type[],
 char scope[], char value[], char isAssign)
{
    int q = search(name, id_type, scope);
    if(q != -1){
        update(q, value, isAssign);
    }
    else {

        if(checkValidDataTye(data_type)){
            st[k].sl_no = k;
            strcpy(st[k].name, name);
            strcpy(st[k].id_type, id_type);
            strcpy(st[k].data_type, data_type);
            strcpy(st[k].scope, scope);
            strcpy(st[k].value, value);

            totalSymbleInTable++;
            k++;
        }
    }
    number_of_value_in_symble_table = k;
}

void display()
{
    printf("\t|-------|----------|----------|------------|----------
|--------|\n");
    printf("\t|%-6s | %-8s | %-8s | %-10s | %-8s | %-
6s |\n", "Sl.NO", "Name","Id Type","Data Type","Scope", "Value");
    printf("\t|=======|==========|==========|============|==========|
========|\n");
    for(int i=1; i<k; i++)
    {
        printf("\t|%-6d | %-8s | %-8s | %-10s | %-8s | %-
6s |\n",st[i].sl_no,st[i].name,st[i].id_type,st[i].data_type,st[i].sc
ope,st[i].value);
        printf("\t|-------|----------|----------|------------|-------
---|--------|\n");
    }
}

int make_symble_table()
{
    char input_step1[100000];
    char output_step1[100000];
    char output_with_space[100000];

//    FILE *f1 = fopen("input_Assignment3.txt", "r");
    FILE *f1 = fopen("output_assignment_2.txt", "r");
    if(f1)
```

```
        fgets(input_step1, 100000, f1);
    else
        printf("File Not Found");
    fclose(f1);


    keep_identifier(input_step1,output_step1);

    int j = 0;
    for(int i=0; i<strlen(output_step1); i++)
    {
        if(output_step1[i] == '[')
            continue;
        else if(output_step1[i] == ']')
            continue;
        else
        {
            output_with_space[j] = output_step1[i];
            j++;
        }
    }



    char word[10000][20];
    int k=0;
    char* piece = strtok(output_with_space, " ");
    while(piece != NULL)
    {
        strcpy(word[k], piece);
        k++;
        piece = strtok(NULL, " ");
    }
    strcpy(word[k], "0--end--0");

    int vi = 0;
    int si = 0;
    char scope[1000][20];
    char value[100][20];
    char funcOrVar[100][20];
    int isScopeGlobal = 1;
    strcpy(scope[si],"Global");


    // printf("\t===================== Step 2 =====================\n
\n");
```

```
    for(int i=0; i<10000 - 1; i++)
    {
        if(strcmp("0--end--0", word[i]) == 0)
            break;
        if(strcmp("id", word[i]) == 0)
        {
            if(isFunc(word[i+2]))
            {
                strcpy(funcOrVar[0],"Func");
                if(checkValidDataTye(word[i-1])){
                    insert(sl,word[i+1],funcOrVar[0],word[i-
1],scope[si],"\0",maybeAssignmentOP);
                }
            }
            else
            {
                strcpy(funcOrVar[0],"Var");
                if(strcmp("=", word[i+2]) == 0)
                {
                    strcpy(value[vi],word[i+3]);
                    maybeAssignmentOP = '=';
                }
                insert(sl,word[i+1],funcOrVar[0],word[i-
1],scope[si],value[vi],maybeAssignmentOP);
                vi++;
                maybeAssignmentOP = '!';
            }
            if(strcmp("(", word[i+2]) == 0)
            {
                isScopeGlobal = 0;
                strcpy(scope[si],word[i+1]);
            }
            sl++;
        }
        if(strcmp("}", word[i]) == 0)
        {
            isScopeGlobal = 1;
            strcpy(scope[si],"Global");
        }

    }


    /// display symble table
//     display();
    return 1;
```

```
}


/// $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$



void check_unbalanced_parenthesis(char word[max_word][each_word_len],
 int len_word){
    /// check if unbalanced parenthesis
    output_file = fopen("report_170104030.txt", "a");
    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){
            fprintf(output_file, "At line Number %-
3s: Misplaced Parenthesis\n", line_no);
            printf("At line Number %-
3s: Misplaced Parenthesis\n", line_no);
            is_found_missmatch = 1;
            break;
        }
        if(pr2_end > pr2_start){
            fprintf(output_file, "At line Number %-
3s: Misplaced Curly Brace\n", line_no);
            printf("At line Number %-
3s: Misplaced Curly Brace\n", line_no);
            is_found_missmatch = 1;
            break;
        }


    }

    if(pr1_end != pr1_start && is_found_missmatch == 0){
        if(pr1_start > pr1_end){
            fprintf(output_file, "At line Number %-
3s: Misplaced Parenthesis\n", line_no);
            printf("At line Number %-
3s: Misplaced Parenthesis\n", line_no);
        }
        else{
            fprintf(output_file, "At line Number %-
3s: Misplaced Parenthesis\n", line_no);
            printf("At line Number %-
3s: Misplaced Parenthesis\n", line_no);
        }

    }
    if(pr2_end != pr2_start && is_found_missmatch == 0){
        if(pr2_start > pr2_end){
            fprintf(output_file, "At line Number %-
3s: Misplaced Curly Brace\n", line_no);
            printf("At line Number %-
3s: Misplaced Curly Brace\n", line_no);
        }
        else{
            fprintf(output_file, "At line Number %-
3s: Misplaced Curly Brace\n", line_no);
            printf("At line Number %-
3s: Misplaced Curly Brace\n", line_no);
        }

    }
    fclose(output_file);
    /// end checking unbalanced parenthesis

}
```

```
void check_unbalanced_else(char word[max_word][each_word_len], int le
n_word)
{
    /// check unbalanced else
    output_file = fopen("report_170104030.txt", "a");
    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){
            fprintf(output_file, "At line Number %-
3s: Mismatch Else\n", line_no);
            printf("At line Number %-3s: Mismatch Else\n", line_no);
            break;
        }
    }

    fclose(output_file);
    /// end checking unbalanced else
}

void check_wrong_for_loop(char word[max_word][each_word_len], int len
_word){

    output_file = fopen("report_170104030.txt", "a");
    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], "for") == 0){
            while(strcmp(word[i], ")") != 0)
            {
                if(strcmp(word[i], ",") == 0){
                    fprintf(output_file, "At line Number %-
3s: Expected ';' before ')'\n", line_no);
                    printf("At line Number %-
3s: Expected ';' before ')'\n", line_no);
                }
                i++;
            }
        }
    }

    fclose(output_file);
}
void check_undeclared_id(char word[max_word][each_word_len], int len_
word){

    output_file = fopen("report_170104030.txt", "a");
    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], "id") == 0){
            int temp = number_of_value_in_symble_table;
            int found = 0;
            int j = 0;
            for(j=0;j<number_of_value_in_symble_table;j++){
                if(strcmp(st[j].name, word[i+1]) == 0){
                    found = 1;
                    break;
                }
            }

            if(!found){
                fprintf(output_file, "At line Number %-
3s: Undeclared ID '%s'\n", line_no,word[i+1]);
                printf("At line Number %-
3s: Undeclared ID '%s' \n", line_no, word[i+1]);
            }
        }
    }
    fclose(output_file);
}

void check_duplicate_token(char word[max_word][each_word_len], int le
n_word){

    output_file = fopen("report_170104030.txt", "a");
    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;


        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;
                    fprintf(output_file, "At line Number %-
3s: Duplicate Token\n", line_no);
                    printf("At line Number %-
3s: Duplicate Token\n", line_no);
                    break;
                }

            }

        }

    }

    fclose(output_file);
}

void check_error()
{
    output_file = fopen("report_170104030.txt", "a");
    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);


    /// 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++;
        }
    }
```

```c
/// 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




fprintf(output_file, "\n\nErrors:\n\n");
printf("\n\nErrors:\n\n");
fclose(output_file);
check_unbalanced_parenthesis(word, len_word);
check_unbalanced_else(word, len_word);
check_duplicate_token(word, len_word);
check_wrong_for_loop(word, len_word);
check_undeclared_id(word, len_word);

output_file = fopen("report_170104030.txt", "a");

fprintf(output_file, "\nHeader Files: \t");
printf("\nHeader Files: \t");
for(int i=0;i<len_word;i++)
{
    if(strcmp(word[i], "header") == 0){
        fprintf(output_file,"%s ", word[i+5]);
        printf("%s ", word[i+5]);
    }
}

fprintf(output_file,"\nIdentifiers: \t");
printf("\nIdentifiers: \t");
for(int i=0;i<number_of_value_in_symble_table;i++)
{
    fprintf(output_file, "%s ",st[i].name);
```

```c
        printf("%s ",st[i].name);
    }

    fprintf(output_file,"\nKeyWords: \t");
    printf("\nKeyWords: \t");
    for(int i=0;i<len_word;i++)
    {
        if(strcmp(word[i], "kw") == 0){
            fprintf(output_file, "%s ", word[i+1]);
            printf("%s ", word[++i]);
        }
    }

    fprintf(output_file,"\nFunctions: \t");
    printf("\nFunctions: \t");
    for(int i=0;i<number_of_value_in_symble_table;i++)
    {
        if(strcmp(st[i].id_type, "Func") == 0){
            fprintf(output_file, "%s ",st[i].name);
            printf("%s ",st[i].name);
        }

    }

    fprintf(output_file, "\n");
    printf("\n");
    fclose(output_file);

}

void remove_unnecessary_files()
{
    remove("output_assignment_1.txt");
    remove("output_assignment_2.txt");
    remove("step2input.txt");
    remove("with_line.txt");
    remove("without_comment.txt");
}

int main(void)
{
    take_input();
    call_assignment1();
    call_assignment2();
    make_symble_table();
    check_error();
    remove_unnecessary_files();
    return 0;
}
```