****

**Ahsanullah University of Science & Technology**

**Department of Computer Science & Engineering**

**Course No. :** CSE 4130

**Course Name :** Formal Languages and Compilers Lab

**Assignment No. :** Final Online Assessment

**Submitted By:**

Name : Emdadul Haque

ID No. : 17-01-04-028

Session : Spring - 2020

Section : A (A2)

**Q1.**

Write a program to read a C program as input and find out the user defined functions in the program along with the line number. You must write the output as [Function Name: Line Number] in a file and display the output on console reading from the file.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| #include<stdio.h>  void func()  {  }  int main(void)  {  int a, b;  printf(“”);  func();  scanf(“”);  return 0;  } | func: Line No 2  main: Line No 5 |

**Answer:**

/\*

Name: Emdadul Haque

ID:17 01 04 028

Lab Group: A2

Course No: CSE4130 Final Online Assessment, Spring 2020

\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

#define NUMBER\_OF\_KEYWORDS 32

char keywords[NUMBER\_OF\_KEYWORDS][100] =

{

"auto","break","case","char","const","continue","default",

"do","double","else","enum","extern","float","for","goto",

"if","int","long","register","return","short","signed",

"sizeof","static","struct","switch","typedef","union",

"unsigned","void","volatile","while"

};

char input[1000000] ;

char all\_code\_in\_line[1000][1000] ;

char allVal\_token[1000][1000] ;

char allVal\_token\_type[1000][1000] ;

int lineCount = 0;

int allVal\_token\_size = 0;

int token\_map\_line[1000];

void fileRead(char \*fileName){

FILE \*fptr;

char c;

int i = 0;

if ((fptr = fopen(fileName, "r")) == NULL) {

printf("Error! opening file");

}else{

while((c = fgetc(fptr)) != EOF){

input[i] = c;

i++;

}

}

fclose(fptr);

}

void printOutput(char str[]){

printf("%s\n",str);

}

int isParen(char ch)

{

int flag = 0;

if (ch == '(' || ch == ')' || ch == '{' || ch == '}' || ch == '[' || ch == ']' ){

flag = 1;

}

return flag;

}

int isOperator(char ch)

{

int flag = 0;

if (ch == '+' || ch == '-' || ch == '\*' ||ch == '/' || ch == '>' || ch == '<' ||ch == '=' ||ch == '!' || ch == '&' || ch == '|'){

flag = 1;

}

return flag;

}

int isValidIdentifier(char str[])

{

if (!((str[0] >= 'a' && str[0] <= 'z') || (str[0] >= 'A' && str[1] <= 'Z')|| str[0] == '\_'))

return 0;

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

if (!((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' && str[i] <= 'Z') || (str[i] >= '0' && str[i] <= '9') || str[i] == '\_'))

return 0;

}

return 1;

}

int isSeperator(char ch)

{

int flag = 0;

if (ch == ';' || ch == ',' || ch == '\'' ){

flag = 1;

}

return flag;

}

void funLine(){

for(int i = 0 ; i < strlen(input) ; i++){

}

}

void convertString(){

char output[1000000];

int j = 0;

for(int i = 0; i < strlen(input) ;i++){

if(isSeperator(input[i]) || isOperator(input[i]) || isParen(input[i])){

output[j++] = ' ';

output[j++] = input[i];

output[j++] = ' ';

}

else{

output[j++] = input[i];

}

}

output[j] = '\0';

strcpy(input , output);

}

void trim(char str[]){

char output[10000];

int length = strlen(str);

if(length <= 0)

return ;

int j = 0;

int start = 0;

for(int i = 0; str[i] == ' ' ;i++)

start++;

for(int i = start ; i < strlen(str) ; i++ ){

if(str[i] == ' ' && str[i+1] == ' '){

int space = 1;

while(space){

i++;

if(str[i+1] != ' '){

space = 0;

output[j] = ' ';

j++;

}

}

}else{

output[j] = str[i];

j++;

}

}

output[j] = '\0';

if(output[strlen(output)-1] == ' ')

output[strlen(output)-1] = '\0';

int i = 0;

for(; i<strlen(output) ;i++){

str[i] = output[i];

}

str[i] = '\0';

}

void convertInLine(){

char buffer[500];

for(int i = 0,j=0 ; i < strlen(input) ;i++){

buffer[j++] = input[i];

if(input[i] == '\n'){

buffer[j] = '\0';

trim(buffer);

strcpy(all\_code\_in\_line[lineCount] , buffer );

lineCount++;

j = 0;

}

}

}

int isKeyword(char str[]){

int flag = 0;

for(int i = 0; i < NUMBER\_OF\_KEYWORDS; i++){

if(strcmp(keywords[i], str) == 0){

flag = 1;

break;

}

}

return flag;

}

char\* check\_type(char str[]){

if(isSeperator(str[0])){

return "sp";

}

else if(isParen(str[0])){

return "par";

}

else if(isOperator(str[0])){

return "op";

}

else{

trim(str);

if(isKeyword(str)){

return "kw";

}

else if(isValidIdentifier(str)){

return "id";

}

}

return "unknown";

}

void parse\_token()

{

for(int i = 0 ; i <= lineCount; i++){

char buffer[500];

for(int j = 0,k = 0; j < strlen(all\_code\_in\_line[i]) ;j++){

buffer[k++] = all\_code\_in\_line[i][j];

if(all\_code\_in\_line[i][j] == ' '){

buffer[k] = '\0';

char \*ty = check\_type(buffer);

trim(buffer);

strcpy(allVal\_token[allVal\_token\_size],buffer);

strcpy(allVal\_token\_type[allVal\_token\_size],ty);

token\_map\_line[allVal\_token\_size] = i;

allVal\_token\_size++;

k = 0;

}

}

}

}

int main(){

fileRead("input.c");

printf("\n");

convertString();

convertInLine();

parse\_token();

char Functions[10000] ;

for(int i = 0 ; i <= allVal\_token\_size; i++){

if( strcmp(allVal\_token\_type[i], "id") == 0 && allVal\_token[i+1][0] == '('){

if(!strcmp(allVal\_token[i], "printf") || !strcmp(allVal\_token[i], "scanf")){

}else{

if(strstr(Functions, allVal\_token[i]) == 0 ){

strcat(Functions, allVal\_token[i]);

strcat(Functions, ", ");

}

printf("%s ,Line No : %d\n",allVal\_token[i], token\_map\_line[i]+1);

}

}

}

printf("So, function is : %s \n",Functions);

return 0;

}

**Q2.**

**Design a recursive-descent parser for the following grammar and mention some strings (at least one from each production rule) from the language generated by the grammar.**

**E → aA | bAB**

**A → b | bA**

**B → a |**

**Answer:**

/\*

Name: Emdadul Haque

ID: 17 01 04 028

Lab Group:A2

Course No: CSE4130 Final Online Assessment, Spring 2020

\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

int n,i;

char s[100];

int B()

{

if(s[i]=='a')

{

i++;

return 1;

}

return 1;

}

int A()

{

if(s[i]=='b')

{

i++;

if(A())

return 1;

return 1;

}

return 0;

}

int E()

{

if(s[i]=='a')

{

i++;

if(A())

return 1;

return 0;

}

if(s[i]=='b')

{

i++;

if(A())

{

if(B())

{

return 1;

}

return 0;

}

return 0;

}

return 0;

}

int main()

{

printf("Enter input string: ");

scanf("%s",&s);

n = strlen(s);

i = 0;

int x = E();

printf("Output: ");

if( i == n && x )

printf("Accepted\n");

else

printf("Rejected\n");

return 0;

}