Exercise 2
Lexical Analyser and Symbol Table using Lex

Nanda H Krishna
312217104093

January 20, 2020

1 Lex Program

```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int base = 1000, flag = 0, fg[20], cn = 0;
char symbols[20][20], values[20][20];
%}
keyword ("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")
function [a-zA-Z_][a-zA-Z0-9_]*[(].*[)]
identifier [a-zA-Z_][a-zA-Z0-9_]*
int_constant [0-9]+
float_constant [0-9]+.[0-9]+
%%
^#.* {
    printf("%s - preprocessor directive\n", yytext);
}
{keyword} {
    int i = 0;
    printf("%s - keyword\n", yytext);
    if(strcmp(yytext, "int") == 0) flag = 2;
    else if(strcmp(yytext,"float") == 0) flag = 4;
}
{function} {
    printf("%s - function call\n", yytext);
}
{identifier} {
    printf("%s - identifier\n", yytext);
```

```
    strcpy(symbols[cn],yytext);
}
{int_constant} {
    printf("%s - integer constant\n", yytext);
    strcpy(values[cn],yytext);
    fg[cn] = flag;
    cn++;
}
{float_constant} {
    printf("%s - float/double constant\n", yytext);
    strcpy(values[cn],yytext);
    fg[cn] = flag;
    cn++;
}
("<"|"<="|">"|">="|"=="|"!=") {
    printf("%s - relational operator\n", yytext);
}
= {
    printf("%s - assignment operator\n", yytext);
}
[{}(),;] {
    printf("%s - special character\n", yytext);
}
. { }
\n { }
%%
int main(int argc, char* argv[])
{
    yyin = fopen(argv[1], "r");
    yylex();
    printf("Symbol Table:\nType\tSymbol\tSize\tAddress\tValue\n");
    for (int i = 0; i < cn; i++)
    {
        printf("%s\t %s\t %d\t %d\t %s\n", (fg[i] == 2) ? "int": "float",
        symbols[i], fg[i], base, values[i]);
        base += fg[i];
    }
    return 0;
}
```

2 Input Program

```c
#include<stdio.h>
main()
{
    int a = 10, b = 20;
    float c = 9.25;
    if (a > b)
        printf("a is greater");
    else
        printf("b is greater");
}
```

3 Output

```
#include<stdio.h> - preprocessor directive
main() - function call
{ - special character
int - keyword
a - identifier
= - assignment operator
10 - integer constant
, - special character
b - identifier
= - assignment operator
20 - integer constant
; - special character
float - keyword
c - identifier
= - assignment operator
9.25 - float/double constant
; - special character
if - keyword
( - special character
a - identifier
> - relational operator
b - identifier
) - special character
printf("a is greater") - function call
; - special character
else - keyword
printf("b is greater") - function call
; - special character
} - special character
```

```
Symbol Table:
Type  Symbol Size  Address  Value
int    a      2      1000    10
int    b      2      1002    20
float  c      4      1004    9.25
```