AASHNA NITIN KUNKOLIENKER
190905304, CSE-D-44
CD lab 4, 29-10-21

Question 1)
1. Using getNextToken( ) implemented in Lab No 3,design a Lexical Analyser to implement local and global symbol table to store tokens for identifiers using array of structure

// worked over the previous lab’s code, modifications are highlighted.

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<ctype.h>
struct token
{


int index;
unsigned int row,col;//Line numbers
char token_name[ ];

};

struct symboltable{
    char name[20];
    char type[20];
    int size;
}st[20];
int ctr=0;
int i,row,col;
char ltype[20]; // buffer to save type names of every identifier before updating st
char spsymbols[] = {'?', ';', ':', ',', '.'};
char *keywords[] = {"int", "return", "for", "while", "if", "else","printf", "case", "break",
"float", "const", "bool"};
char arithsymbols[] = {'*', '%'};  /* + - not included because of the possibility of ++ and --
*/
int search(char* name) //to search for an entry in the Symbol Table
{
for(int i=0;i<ctr;i++)
{
if(strcmp(name,st[i].name) == 0)
{
return 1;
}
}
return 0;
}
```

```
void enterinSt(char* name , char* type) //inserting a row in the symbol table
{
    if(search(name) == 0) //to avoid repetition of identifiers
    {
                strcpy(st[ctr].name,name);
        strcpy(st[ctr].type,type);
        printf("%s",st[ctr].type);
        if(strcmp(type,"func") == 0)
        {
            st[ctr].size = -999; //if function don't set any size
        }
        else
        {
        st[ctr].size = 4;

        }
        ctr++;
    }
}

int iskeyword(char *key)
{
    for(i=0;i<12;i++)
    {
    if (strcmp(key, keywords[i]) == 0)
    {
    return 1;
    }

    }
    return 0;
}


int isarithsymbol(char ch)
{
    for(i=0;i<4;i++)
    {
    if(arithsymbols[i]==ch)
    {
    return 1;
    }
    }
    return 0;
}

void newLine()
{
    row++;
    col = 1;
}
```

```
int isspsymbol(char ch)
{
    for(i=0;i<5;i++)
    {
    if(spsymbols[i]==ch)
    {
    return 1;
    }
    }
    return 0;
}

void tokeninput(struct token *tok, char c, int row, int col)
{
    tok->token_name[0] = c;

    tok->index =0;
    tok->row = row;
    tok->col = col;
}


struct token getNextToken(FILE *f,char *name)
{
    char c;
    struct token tkn;
    tkn.row=-1;


    int gotToken = 0;
    while (!gotToken && (c = fgetc(f)) != EOF)
    {
        if (isspsymbol(c)||isarithsymbol(c))
        {
            tokeninput(&tkn, c, row, col);
            gotToken = 1;
            ++col;
        }

        else if (c == '('||c==')'||c=='}'||c=='{'||c==']'||c=='[')
        {
            tokeninput(&tkn, c, row, col);
            gotToken = 1;

            ++col;
        }

        else if (c == '+')
        {
            int next = fgetc(f);
            if (next != '+')
            {
                tokeninput(&tkn, c, row, col);
                gotToken = 1;
                ++col;
                fseek(f, -1, SEEK_CUR); //shift file pointer back by 1
            }
            else
            {
                tokeninput(&tkn, c, row, col);
                strcpy(tkn.token_name, "++");
                gotToken = 1;
                col += 2; /*skip 2 columns cuz after reading c we read another character too
*/
            }
        }
            else if (c == '-')
        {
            int next = fgetc(f);
            if (next != '-')
            {
                tokeninput(&tkn, c, row, col);
                gotToken = 1;
                ++col;
                fseek(f, -1, SEEK_CUR); //shift file pointer back by 1
            }
            else
            {
                tokeninput(&tkn, c, row, col);
                strcpy(tkn.token_name, "--");
                gotToken = 1;
                col += 2; /*skip 2 columns cuz after reading c we read another character too
*/
            }
        }

            else if (c == '=')
        {
            int next = fgetc(f);
            if (next != '=')
            {
                tokeninput(&tkn, c, row, col);
                gotToken = 1;
                ++col;
                fseek(f, -1, SEEK_CUR); //shift file pointer back by 1
            }
            else
            {
                tokeninput(&tkn, c, row, col);
                strcpy(tkn.token_name, "==");
                gotToken = 1;
                col += 2; /*skip 2 columns cuz after reading c we read another character too
*/
```

```
    }
}

else if (isdigit(c))
{
    tkn.row = row;
    tkn.col = col++;
    strcpy(tkn.token_name,"num");

    while ((c = fgetc(f)) != EOF && isdigit(c))
    {
        col++;
    }
    gotToken = 1;
    fseek(f, -1, SEEK_CUR);//shift file pointer back by 1
}

 else if (c == '#')
{
    while ((c = fgetc(f)) != EOF && c != '\n')
    {
    continue;
    }

    newLine();
}
else if (c == '\n')
{
    newLine();
    c = fgetc(f);
    if (c == '#')
    {
        while ((c = fgetc(f)) != EOF && c != '\n')
            ;
        newLine();
    }
    else if (c != EOF)
        fseek(f, -1, SEEK_CUR);
}

else if (isspace(c))
    ++col;

else if (isalpha(c) || c == '_')
{
    tkn.row = row;
    tkn.col = col++;
    tkn.token_name[0]=c;
   int k = 1;
    while ((c = fgetc(f)) != EOF && isalnum(c))
    {
        tkn.token_name[k++] = c;
```

```
                ++col;
            }
            tkn.token_name[k] = '\0';
            int flag=1;
            for(int i = 0 ; i < 12 ; i++){
                if(strcmp(tkn.token_name,keywords[i]) == 0)
                {
                    if((strcmp(tkn.token_name,"int")==0)||strcmp(tkn.token_name,"float")==0)
                    {

                        strcpy(ltype,tkn.token_name);
                    }
                   flag=0;
                    break;
                }
            }
```

/* if it is a keyword, let the token name be the keyword itself.. else token name should be "id" for an identifier */

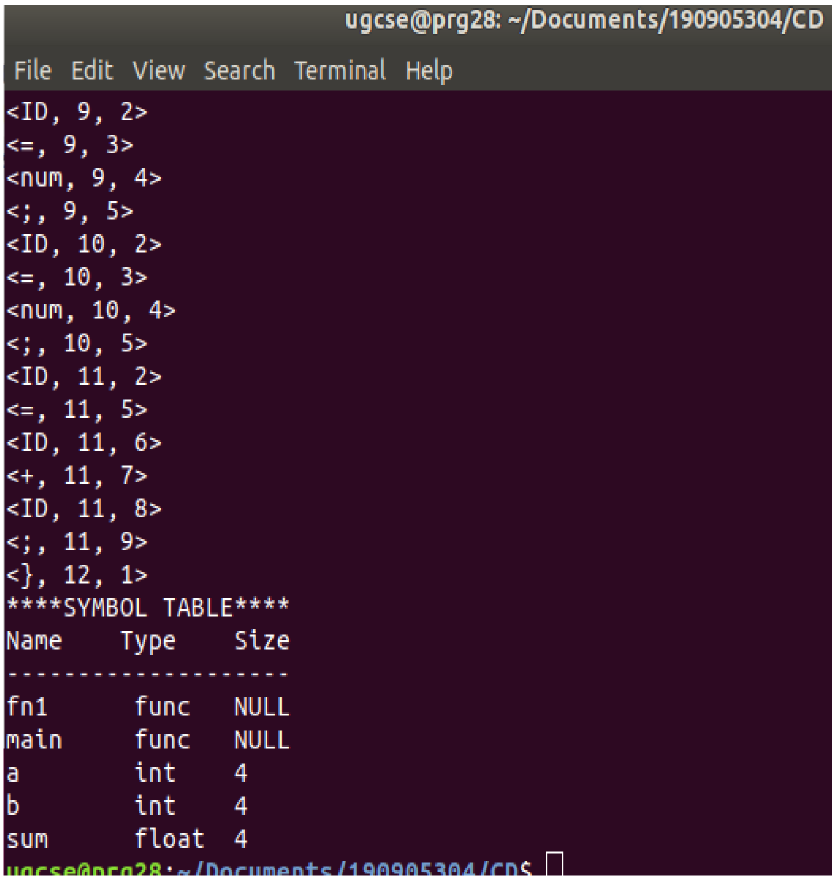
```
               if(flag==1)
               {
                if(c=='(')
                {
                    printf("found a function %s",tkn.token_name);
                    char nm[10];
                    strcpy(nm,"func");
                    enterinSt(tkn.token_name,nm);


                }
                else
                {
                    enterinSt(tkn.token_name,ltype);

                 }
                 strcpy(tkn.token_name,"ID");

                }


         gotToken = 1;
         fseek(f, -1, SEEK_CUR);
        }
        //comment removal code from first lab

        else if (c == '/')
        {
            int d = fgetc(f);
            ++col;
            if (d == '/')
            {
```

```
        while ((c = fgetc(f)) != EOF && c != '\n')
            ++col;
        if (c == '\n')
            newLine();
    }
    else if (d == '*')
    {
        do
        {
            if (d == '\n')
                newLine();
            while ((c == fgetc(f)) != EOF && c != '*')
            {
                ++col;
                if (c == '\n')
                    newLine();
            }
            ++col;
        } while ((d == fgetc(f)) != EOF && d != '/' && (++col));
        ++col;
    }
    else
    {
       tokeninput(&tkn, c, row, --col);
        gotToken = 1;
        fseek(f, -1, SEEK_CUR);
    }
}

else if (c == '"')
{
    tkn.row = row;
    tkn.col = col;
    int k = 1;
    tkn.token_name[0]= '"';
    while ((c = fgetc(f)) != EOF && c != '"')
    {
        tkn.token_name[k++] = c;
        ++col;
    }
    tkn.token_name[k] ='"';
    gotToken = 1;
}

//code of solved exercise modified:

else if (c == '<' || c == '>' || c == '!')
{
    tokeninput(&tkn, c, row, col);
    ++col;
    int c = fgetc(f);
    if (c == '=')
```

```
            {
                ++col;
                strcat(tkn.token_name, "=");
            }
            else
            {
                fseek(f, -1, SEEK_CUR);
            }
            gotToken = 1;
        }

         else if (c == '&' || c == '|')
        {
            int d = fgetc(f);
            if (c == d) //handling && and ||
            {
                tkn.token_name[0] = tkn.token_name[1] = c;
                tkn.token_name[2] = '\0';
                tkn.row = row;
                tkn.col = col;
                ++col;
                gotToken = 1;
            }
            else // just & or |
            {
                tkn.token_name[0] = c;
                tkn.token_name[1] = '\0';
                tkn.row = row;
                tkn.col = col;
                ++col;
                gotToken = 1;
                fseek(f, -1, SEEK_CUR); //shift file pointer back by 1
            }
            ++col;
        }
        else
            ++col;
    }
    strcpy(name,tkn.token_name);
   // printf("%s ",name);
    return tkn;
}

int main()
{

    char c, buf[10];

    printf("Enter file name: ");
    char input[256];
    scanf("%s", input);
    FILE *fp=fopen(input,"r");
    if (fp == NULL)
    {
        printf("Cannot open file\n");
        exit(0);
    }

    struct token tok;
    char nm[100];
    while ((tok = getNextToken(fp,nm)).row != -1)
    {
        printf("<%s, %d, %d>\n",nm, tok.row, tok.col);
    }
    printf("****SYMBOL TABLE****");
    printf("\nName\tType\tSize\n");
    printf("--------------------\n");
    for(int j=0;j<ctr;j++)
    {
        printf("%s \t %s \t",st[j].name,st[j].type);
        if(st[j].size==-999)
        {
            printf("NULL\n");
        }
        else

        {
            printf("%d \n",st[i].size);
        }
    }
    fclose(fp);
    return 0;
}
```

SAMPLE PROGRAM USED :

```
#include<stdio.h>
fn1()
{
        printf("Hi");
}
main()
{
        int a,b;
        float sum;
        a=1;
        b=1;
        sum=a+b;
}
```


ugcse@prg28: ~/Documents/190905304/CD
File Edit View Search Terminal Help

```
<ID, 9, 2>
<=, 9, 3>
<num, 9, 4>
<;, 9, 5>
<ID, 10, 2>
<=, 10, 3>
<num, 10, 4>
<;, 10, 5>
<ID, 11, 2>
<=, 11, 5>
<ID, 11, 6>
<+, 11, 7>
<ID, 11, 8>
<;, 11, 9>
<}, 12, 1>
****SYMBOL TABLE****
Name    Type    Size
--------------------
fn1      func   NULL
main     func   NULL
a        int    4
b        int    4
sum      float  4
ugcse@prg28:~/Documents/190905304/CD$
```