Aditya Pradhan CSE CD Lab3 180905350

Q1
Code

```
/*
Design alexical analyzer which containsgetNextToken( ) for a simple C program to create a structure of token each time and return, which includes row number, column number and token type. The tokens to be identified are arithmetic operators,relational operators, logical operators,special symbols, keywords, numerical constants, string literalsand identifiers. Also, getNextToken() should ignore all the tokens when encountered inside single line or multiline comment or inside string literal. Preprocessor directive should also be stripped.
*/
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
#include <stdbool.h>

struct token
{
        unsigned int row, col;
        char type[10];
};

FILE *fa, *fb;

bool isValidDelimiter(char ch) {
  if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
  ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
  ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
  ch == '[' || ch == ']' || ch == '{' || ch == '}')
  return (true);
  return (false);
}
bool isValidOperator(char ch){
  if (ch == '+' || ch == '-' || ch == '*' ||
  ch == '/' || ch == '>' || ch == '<' ||
  ch == '=')
  return (true);
  return (false);
}
bool isvalidIdentifier(char* str){
  if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||
  str[0] == '3' || str[0] == '4' || str[0] == '5' ||
  str[0] == '6' || str[0] == '7' || str[0] == '8' ||
  str[0] == '9' || isValidDelimiter(str[0]) == true)
  return (false);
```

```
  return (true);
}
bool isValidKeyword(char* str) {
  if (!strcmp(str, "if") || !strcmp(str, "else") || !strcmp(str, "while") || !strcmp(str, "do") ||    !strcmp(str,
"break") || !strcmp(str, "continue") || !strcmp(str, "int")
  || !strcmp(str, "double") || !strcmp(str, "float") || !strcmp(str, "return") || !strcmp(str,    "char") || !
strcmp(str, "case") || !strcmp(str, "char")
  || !strcmp(str, "sizeof") || !strcmp(str, "long") || !strcmp(str, "short") || !strcmp(str, "typedef") || !
strcmp(str, "switch") || !strcmp(str, "unsigned")
  || !strcmp(str, "void") || !strcmp(str, "static") || !strcmp(str, "struct") || !strcmp(str, "goto"))
  return (true);
  return (false);
}
bool isValidInteger(char* str) {
  int i, len = strlen(str);
  if (len == 0)
  return (false);
  for (i = 0; i < len; i++) {
    if (str[i] != '0' && str[i] != '1' && str[i] != '2'&& str[i] != '3' && str[i] != '4' && str[i] != '5'
    && str[i] != '6' && str[i] != '7' && str[i] != '8' && str[i] != '9' || (str[i] == '-' && i > 0))
    return (false);
  }
  return (true);
}
bool isRealNumber(char* str) {
  int i, len = strlen(str);
  bool hasDecimal = false;
  if (len == 0)
  return (false);
  for (i = 0; i < len; i++) {
    if (str[i] != '0' && str[i] != '1' && str[i] != '2' && str[i] != '3' && str[i] != '4' && str[i]       != '5' &&
str[i] != '6' && str[i] != '7' && str[i] != '8'
    && str[i] != '9' && str[i] != '.' || (str[i] == '-' && i > 0))
    return (false);
      if (str[i] == '.')
    hasDecimal = true;
  }
  return (hasDecimal);
}

bool isComment(char c)
{
char ca = getc(fa), cb;

          while(ca != EOF)
          {
                    if(ca == ' ')
```

```
		{
			while(ca == ' ')
			{
				ca = getc(fa);
			}
		}

		if(ca == '/')
		{
			cb = getc(fa);
			if(cb == '/') // single line comment
			{
				while(ca != '\n')
				{
					ca = getc(fa);
				}
				return true;
			}
			else if(cb == '*') // for multiline comment
			{
				do
				{
					while(ca != '*')
					{
						ca = getc(fa);
					}
					ca = getc(fa);
				}while(ca != '/');
				return true;
			}

		}
		ca = getc(fa);
	}
}

int main()
{

	char filename[100];
	printf("enter the input file name: \n");
	scanf("%s", filename);
	fa = fopen(filename, "r");

	if(fa == NULL)
	{
		printf("file not found \n");
```

```
		exit(0);
	}

	char c = getc(fa);
	int row = 1, col = 1;
	while(c != EOF)
	{
		if(c == '\n') // check for end of line
		{
			col = 1;
			row++;
			c = getc(fa);
			printf("\n");
			continue;
		}

		if(c == '#')	// check for preprocessor directive
		{
			//ca = getc(fptr1);
			while(c != EOF && c != '\n')
			{
				c = getc(fa);
				//printf("inside loop 1\n");
			}
			c = getc(fa);
			//printf("after loop 1\n");
			if(c == EOF)
			{
				break;
			}
			continue;
		}

		col++;
		char ans[10];
		int i = 0;
		if(isalpha(c)) // check for keyword and identifiers
		{
			ans[i++] = c;
			c = getc(fa);
			col++;
			while(i < 10 && isalpha(c))
			{
				ans[i++] = c;
				col++;
				//printf("here1 \n");
			}
```

```
                ans[i] = '\0';

                if(isvalidIdentifier(ans) || isValidKeyword(ans))
                {
                        printf("<id,%d,%d,> ",row, col);
                        c = getc(fa);
                }
        }
        else if(isValidOperator(c) || isValidDelimiter(c))      // check for operators
        {
                printf("<%c,%d,%d,> ", c, row, col);
                c = getc(fa);
        }
        else if(isdigit(c))         // check for digits
        {
                char ans[10];
                int i = 0;
                ans[i++] = c;
                c = getc(fa);
                col++;
                while(isdigit(c))
                {
                        ans[i++] = c;
                        col++;
                }
                ans[i] = '\0';

                if(isValidInteger(ans))
                {
                        printf("<%s,%d,%d, >", ans, row, col);
                }
        }
        else
        {
                c = getc(fa);
        }


    }
    printf("\n");
    return 0;
}
```

Aditya Pradhan CSE CD Lab3 180905350

Output