CD Lab 3

Name:Rhea Adhikari
Reg No:190905156
Roll No:23

Q) Design a lexical analyzer which contains getNextToken( ) 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 literals and 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.

main.c

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

int i,row,col;
char arithsymbols[] = {'*', '%'};
char *keywords[] = {"int", "return", "for", "while", "if", "else","printf", "case", "break", "float",
"const", "bool"};
char spsymbols[] = {':', ',', '.','?', ';'};

struct token
{
        int index;
        unsigned int row,col;//Line numbers
        char token_name[20];
};

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;
}

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

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

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


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

struct token getNextToken(FILE *f,char *name)
{
	char c;
	struct token tkn;
	tkn.row=-1;
	int gotToken = 0;
	//if we find the token then out of loop
	//OR
	//if we reach eof then out of loop
	while (!gotToken && (c = fgetc(f)) != EOF)
	{
		if (isSpecialSymbol(c)||isArithmeticSymbol(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);
		}
		else
		{
			tokenInput(&tkn, c, row, col);
			strcpy(tkn.token_name, "++");
			gotToken = 1;
			col += 2;
		}
	}
	else if (c == '-')
	{
		int next = fgetc(f);
		if (next != '-')
		{
			tokenInput(&tkn, c, row, col);
			gotToken = 1;
			++col;
			fseek(f, -1, SEEK_CUR);
		}
		else
		{
			tokenInput(&tkn, c, row, col);strcpy(tkn.token_name, "--");
			gotToken = 1;
			col += 2;
		}
	}
	else if (c == '=')
	{
		int next = fgetc(f);
		if (next != '=')
		{
			tokenInput(&tkn, c, row, col);
			gotToken = 1;
			++col;
			fseek(f, -1, SEEK_CUR);
		}
		else
		{
			tokenInput(&tkn, c, row, col);
			strcpy(tkn.token_name, "==");
```

```
                gotToken = 1;
                col += 2;
        }
}
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);
}
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++;
        int k = 1;
        while ((c = fgetc(f)) != EOF && isalnum(c))
        {
                tkn.token_name[k++] = c;
                ++col;
        }
        tkn.token_name[k] = '\0';
```

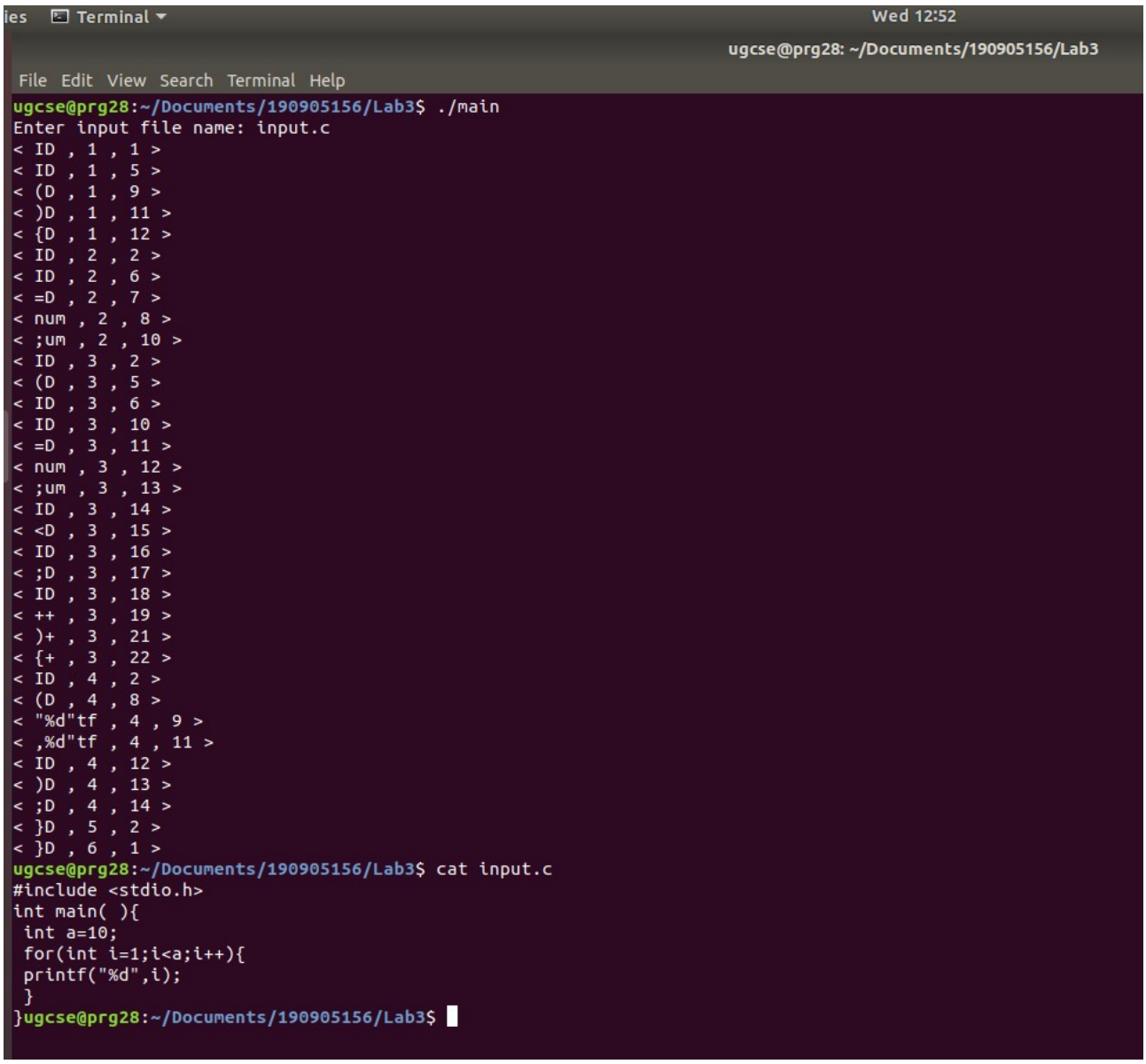
```
		int flag=1;
		for(int i = 0 ; i < 12 ; i++)
		{
			if(strcmp(tkn.token_name,keywords[i]) == 0)
			{
				flag=0;
				break;
			}
			if(flag==1)
			{
				strcpy(tkn.token_name,"ID");
			}
		}
		gotToken = 1;
		fseek(f, -1, SEEK_CUR);
	}
	else if (c == '/')
	{
		int d = fgetc(f);
		++col;
		if (d == '/')
		{
			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);
        return tkn;
}


int main()
{
        char c;

        char buf[10];
        char input[256];

        printf("Enter input file name: ");
        scanf("%s", input);

        FILE *fp1=fopen(input,"r");//read file

        if (fp1 == NULL)
        {
                printf("Cannot open the file\n");
                exit(0);
        }

        struct token tok;

        char namee[100];

        while ((tok = getNextToken(fp1,namee)).row != -1)
                printf("< %s , %d , %d >\n",namee, tok.row, tok.col);
        fclose(fp1);

        return 0;
}
```

input.c

```
#include <stdio.h>
int main( ){
 int a=10;
 for(int i=1;i<a;i++){
 printf("%d",i);
 }
}
```


ies Terminal
Wed 12:52
ugcse@prg28: ~/Documents/190905156/Lab3
File Edit View Search Terminal Help
ugcse@prg28:~/Documents/190905156/Lab3$ ./main
Enter input file name: input.c
< ID , 1 , 1 >
< ID , 1 , 5 >
< (D , 1 , 9 >
<)D , 1 , 11 >
< {D , 1 , 12 >
< ID , 2 , 2 >
< ID , 2 , 6 >
< =D , 2 , 7 >
< num , 2 , 8 >
< ;um , 2 , 10 >
< ID , 3 , 2 >
< (D , 3 , 5 >
< ID , 3 , 6 >
< ID , 3 , 10 >
< =D , 3 , 11 >
< num , 3 , 12 >
< ;um , 3 , 13 >
< ID , 3 , 14 >
< <D , 3 , 15 >
< ID , 3 , 16 >
< ;D , 3 , 17 >
< ID , 3 , 18 >
< ++ , 3 , 19 >
<)+ , 3 , 21 >
< {+ , 3 , 22 >
< ID , 4 , 2 >
< (D , 4 , 8 >
< "%d"tf , 4 , 9 >
< ,%d"tf , 4 , 11 >
< ID , 4 , 12 >
<)D , 4 , 13 >
< ;D , 4 , 14 >
< }D , 5 , 2 >
< }D , 6 , 1 >
ugcse@prg28:~/Documents/190905156/Lab3$ cat input.c
#include <stdio.h>
int main( ){
int a=10;
for(int i=1;i<a;i++){
printf("%d",i);
}
}ugcse@prg28:~/Documents/190905156/Lab3$