Name : Tushar Pathak
Registration Number : 230905396
Roll Number : CSE6B-48
Week Num : 3

LAB 3 : CONSTRUCTION OF TOKEN GENERATOR

Q1) Write functions to identify the following tokens.
a. Arithmetic, relational and logical operators.
b. Special symbols, keywords, numerical constants, string literals and identifiers.

Code:

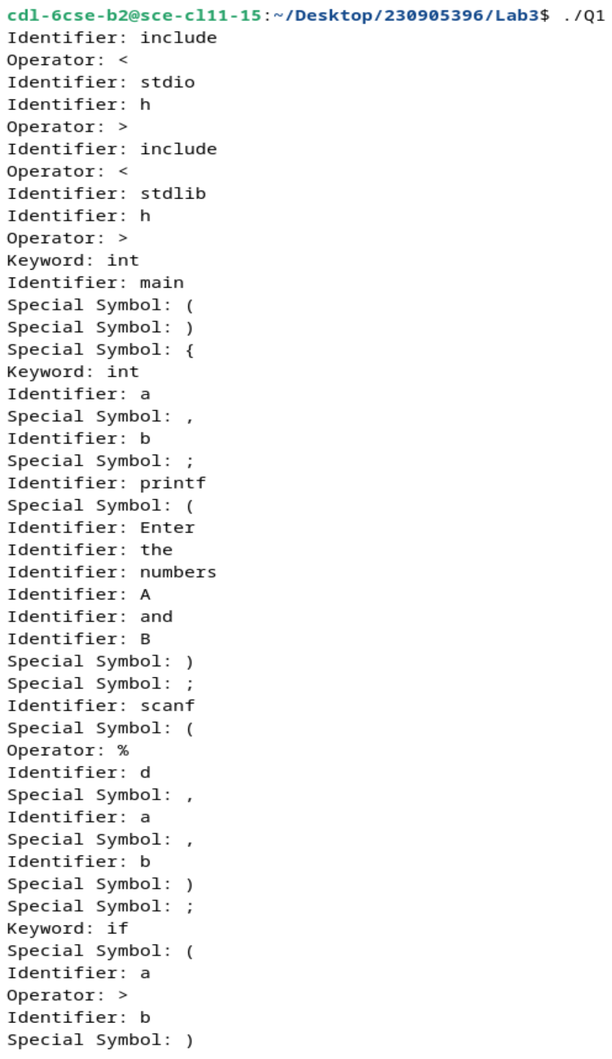
```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

char keywords[][10] = {
        "int","char","if","else","while","for","return","void"
};

int isKeyword(char str[]) {
        for (int i = 0; i < 8; i++) {
                if (strcmp(keywords[i], str) == 0)
                        return 1;
        }
        return 0;
}

int main() {
FILE *fp;
char ch, buf[20];
int i;

fp = fopen("Q1Input.c", "r");

while ((ch = fgetc(fp)) != EOF) {

        /* Identifier and keyword */
        if (isalpha(ch)) {
                i = 0;
                buf[i++] = ch;

                while (isalnum(ch = fgetc(fp)))
                        buf[i++] = ch;

                buf[i] = '\0';
                ungetc(ch, fp);
```

```
			if (isKeyword(buf))
				printf("Keyword: %s\n", buf);
			else
				printf("Identifier: %s\n", buf);
		}

		/* Numeric constant */
		else if (isdigit(ch)) {
			i = 0;
			buf[i++] = ch;

			while (isdigit(ch = fgetc(fp)))
				buf[i++] = ch;

			buf[i] = '\0';
			ungetc(ch, fp);

			printf("Number: %s\n", buf);
		}

		/* Operators */
		else if (strchr("+-*/%=<>!", ch)) {
			printf("Operator: %c\n", ch);
		}

		/*special symbols*/
		else if (strchr(";,(){}", ch)) {
			printf("Special Symbol: %c\n", ch);
		}
	}
	fclose(fp);
}
```

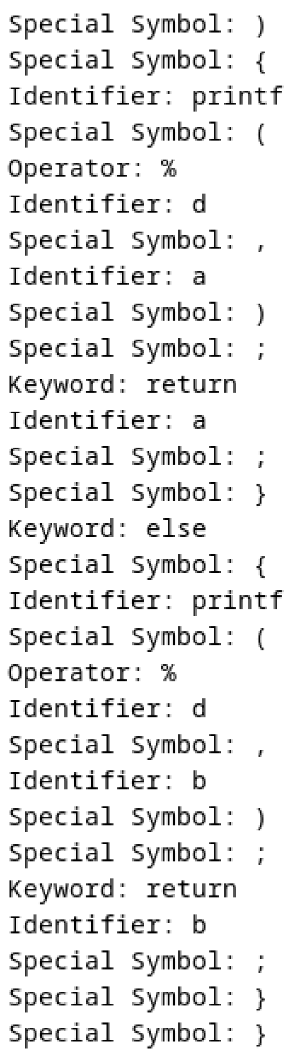
Q1Input.c

```
#include<stdio.h>
#include<stdlib.h>
int main(){
	int a, b;
	printf("Enter the numbers A and B...");
	scanf("%d",&a,&b);
	if(a>b){
		printf("%d",a);
		return a;
	}else{
		printf("%d",b);
		return b;
	}
}
```

Output:



```
cdl-6cse-b2@sce-cl11-15:~/Desktop/230905396/Lab3$ ./Q1
Identifier: include
Operator: <
Identifier: stdio
Identifier: h
Operator: >
Identifier: include
Operator: <
Identifier: stdlib
Identifier: h
Operator: >
Keyword: int
Identifier: main
Special Symbol: (
Special Symbol: )
Special Symbol: {
Keyword: int
Identifier: a
Special Symbol: ,
Identifier: b
Special Symbol: ;
Identifier: printf
Special Symbol: (
Identifier: Enter
Identifier: the
Identifier: numbers
Identifier: A
Identifier: and
Identifier: B
Special Symbol: )
Special Symbol: ;
Identifier: scanf
Special Symbol: (
Operator: %
Identifier: d
Special Symbol: ,
Identifier: a
Special Symbol: ,
Identifier: b
Special Symbol: )
Special Symbol: ;
Keyword: if
Special Symbol: (
Identifier: a
Operator: >
Identifier: b
Special Symbol: )
```



```
Special Symbol: )
Special Symbol: {
Identifier: printf
Special Symbol: (
Operator: %
Identifier: d
Special Symbol: ,
Identifier: a
Special Symbol: )
Special Symbol: ;
Keyword: return
Identifier: a
Special Symbol: ;
Special Symbol: }
Keyword: else
Special Symbol: {
Identifier: printf
Special Symbol: (
Operator: %
Identifier: d
Special Symbol: ,
Identifier: b
Special Symbol: )
Special Symbol: ;
Keyword: return
Identifier: b
Special Symbol: ;
Special Symbol: }
Special Symbol: }
```

Q2) Design a lexical analyzer that includes a getNextToken() function for processing a simple C program. The analyzer should construct a token structure containing the row number, column number, and token type for each identified token. The getNextToken() function must ignore tokens located within singleline or multi-line comments, as well as those found inside string literals. Additionally, it should strip out preprocessor directives.

Code:

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

typedef struct {
   char lexeme[30];
   char type[30];
   int row, col;
} Token;

char *keywords[] = {
   "int","char","float","if","else","while","return",NULL
```

```
};

int isKeyword(char *str) {
   for (int i = 0; keywords[i] != NULL; i++) {
      if (strcmp(keywords[i], str) == 0)
         return 1;
   }
   return 0;
}

Token getNextToken(FILE *fp) {
   static int row = 1, col = 0;
   Token t;
   char c;
   int i = 0;

   while ((c = fgetc(fp)) != EOF) {
      col++;

      if (c == '\n') {
         row++;
         col = 0;
         continue;
      }

      /* Ignore preprocessor */
      if (c == '#') {
         while (c != '\n')
            c = fgetc(fp);
         row++;
         col = 0;
         continue;
      }

      /* Ignore comments */
      if (c == '/') {
         char next = fgetc(fp);
         if (next == '/') {
            while (c != '\n')
               c = fgetc(fp);
            row++;
            continue;
         } else if (next == '*') {
            while (!(c == '*' && fgetc(fp) == '/'))
               c = fgetc(fp);
            continue;
         } else {
            ungetc(next, fp);
         }
      }

      /* Ignore string literals */
```

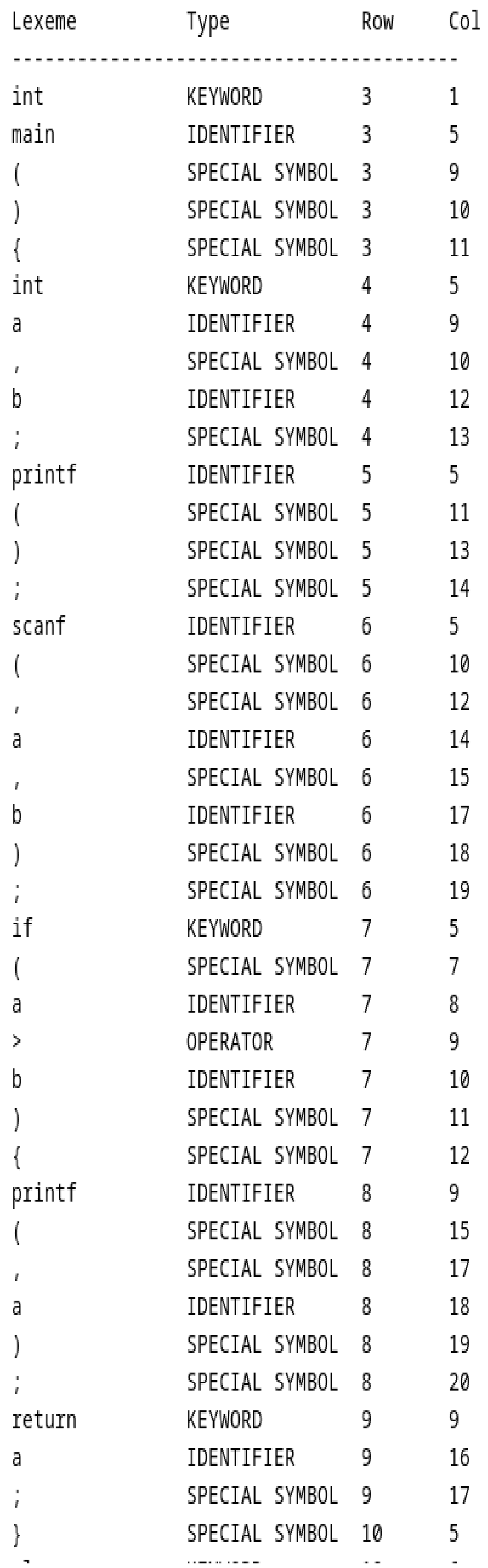
```
if (c == '"') {
    while ((c = fgetc(fp)) != '"' && c != EOF);
    continue;
}

/* Identifier and keyword */
if (isalpha(c)) {
    i = 0;
    t.lexeme[i++] = c;
    while (isalnum(c = fgetc(fp))) {
        t.lexeme[i++] = c;
        col++;
    }
    t.lexeme[i] = '\0';
    ungetc(c, fp);

    strcpy(t.type, isKeyword(t.lexeme) ? "KEYWORD" : "IDENTIFIER");
    t.row = row;
    t.col = col - strlen(t.lexeme) + 1;
    return t;
}

/* Number */
if (isdigit(c)) {
    i = 0;
    t.lexeme[i++] = c;
    while (isdigit(c = fgetc(fp))) {
        t.lexeme[i++] = c;
        col++;
    }
    t.lexeme[i] = '\0';
    ungetc(c, fp);

    strcpy(t.type, "NUMBER");
    t.row = row;
    t.col = col - strlen(t.lexeme) + 1;
    return t;
}

/* Operators */
if (strchr("+-*/<>=!", c)) {
    t.lexeme[0] = c;
    t.lexeme[1] = '\0';
    strcpy(t.type, "OPERATOR");
    t.row = row;
    t.col = col;
    return t;
}

/* Special Symbols */
if (strchr(";(){}[],", c)) {
    t.lexeme[0] = c;
```

```
            t.lexeme[1] = '\0';
            strcpy(t.type, "SPECIAL SYMBOL");
            t.row = row;
            t.col = col;
            return t;
        }
    }

    strcpy(t.type, "EOF");
    return t;
}

int main() {
    FILE *fp = fopen("Q1Input.c", "r");
    Token t;

    while (strcmp((t = getNextToken(fp)).type, "EOF") != 0)
        printf("%s  %s  %d  %d\n", t.lexeme, t.type, t.row, t.col);

    fclose(fp);
    return 0;
}
```

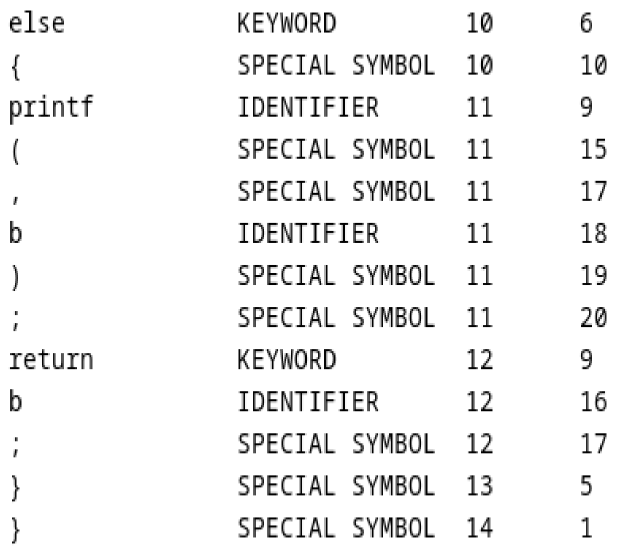
<u>Q1Input.c</u>

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int a, b;
    printf("Enter the numbers A and B...");
    scanf("%d",&a,&b);
    if(a>b){
        printf("%d",a);
        return a;
    }else{
        printf("%d",b);
        return b;
    }
}
```

Output:



```
Lexeme          Type            Row     Col
-----------------------------------------
int             KEYWORD         3       1
main            IDENTIFIER      3       5
(               SPECIAL SYMBOL  3       9
)               SPECIAL SYMBOL  3       10
{               SPECIAL SYMBOL  3       11
int             KEYWORD         4       5
a               IDENTIFIER      4       9
,               SPECIAL SYMBOL  4       10
b               IDENTIFIER      4       12
;               SPECIAL SYMBOL  4       13
printf          IDENTIFIER      5       5
(               SPECIAL SYMBOL  5       11
)               SPECIAL SYMBOL  5       13
;               SPECIAL SYMBOL  5       14
scanf           IDENTIFIER      6       5
(               SPECIAL SYMBOL  6       10
,               SPECIAL SYMBOL  6       12
a               IDENTIFIER      6       14
,               SPECIAL SYMBOL  6       15
b               IDENTIFIER      6       17
)               SPECIAL SYMBOL  6       18
;               SPECIAL SYMBOL  6       19
if              KEYWORD         7       5
(               SPECIAL SYMBOL  7       7
a               IDENTIFIER      7       8
>               OPERATOR        7       9
b               IDENTIFIER      7       10
)               SPECIAL SYMBOL  7       11
{               SPECIAL SYMBOL  7       12
printf          IDENTIFIER      8       9
(               SPECIAL SYMBOL  8       15
,               SPECIAL SYMBOL  8       17
a               IDENTIFIER      8       18
)               SPECIAL SYMBOL  8       19
;               SPECIAL SYMBOL  8       20
return          KEYWORD         9       9
a               IDENTIFIER      9       16
;               SPECIAL SYMBOL  9       17
}               SPECIAL SYMBOL  10      5
```



```
else            KEYWORD         10      6
{               SPECIAL SYMBOL  10      10
printf          IDENTIFIER      11      9
(               SPECIAL SYMBOL  11      15
,               SPECIAL SYMBOL  11      17
b               IDENTIFIER      11      18
)               SPECIAL SYMBOL  11      19
;               SPECIAL SYMBOL  11      20
return          KEYWORD         12      9
b               IDENTIFIER      12      16
;               SPECIAL SYMBOL  12      17
}               SPECIAL SYMBOL  13      5
}               SPECIAL SYMBOL  14      1
```