

# CS 358 LAB 1

Arnav Jain - 220002018

## Question 1

Code:

```
C question1.c x
C question1.c > isValidIdentifier(char *)
You, 39 minutes ago | 1 author (You)
1  #include <stdio.h>
2  #include <ctype.h>
3  #include <string.h>
4
5  int isValidIdentifier(char *identifier)
6  {
7      if (!isalpha(identifier[0]) && identifier[0] != '_')
8      {
9          return 0;
10     }
11     for (int i = 1; i < strlen(identifier); i++)
12     {
13         if (!isalnum(identifier[i]) && identifier[i] != '_')
14         {
15             return 0;
16         }
17     }
18     return 1;
19 }
20
21 int main()
22 {
23     char identifier[100];
24     printf("Enter an identifier: ");
25     scanf("%s", identifier);
26     if (isValidIdentifier(identifier))
27     {
28         printf("%s' is a valid identifier.\n", identifier);
29     }
30     else
31     {
32         printf("%s' is not a valid identifier.\n", identifier);
33     }
34     return 0;
35 }
36
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POLYGLOT NOTEBOOK  GITLENS  SPELL CHECKER

• arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$ ./question1
Enter an identifier: hello
'hello' is a valid identifier.
• arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$ ./question1
Enter an identifier: 123
'123' is not a valid identifier.
• arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$ ./question1
Enter an identifier: _hello
'_hello' is a valid identifier.
• arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$ ./question1
Enter an identifier: _123
'_123' is a valid identifier.
• arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$ ./question1
Enter an identifier: #123
'#123' is not a valid identifier.
○ arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$
```

## Question 2

Code:

```
C question2.c x
C question2.c > analyzeTokens(FILE *)
You, 5 minutes ago | 1 author (You)
1  #include <stdio.h>
2  #include <ctype.h>
3  #include <string.h>
4
5  #define MAX_LENGTH 100
6
7  int total_tokens = 0;
8
9  // Function to check if a string is a keyword
10 int isKeyword(char *word)
11 {
12     char *keywords[] = {"int", "float", "char", "if", "else", "while", "for", "return", "void", NULL};
13     for (int i = 0; keywords[i] != NULL; i++)
14     {
15         if (strcmp(word, keywords[i]) == 0)
16         {
17             total_tokens++;
18             return 1;
19         }
20     }
21     return 0;
22 }
23
24 // Function to check if a character is an operator
25 int isOperator(char c)
26 {
27     char operators[] = "+-*/%=<>(),;.{#[]";
28     for (int i = 0; operators[i] != '\0'; i++)
29     {
30         if (c == operators[i])
31         {
32             total_tokens++;
33             return 1;
34         }
35     }
36     return 0;
37 }
```

```

C question2.c x
C question2.c > analyzeTokens(FILE *)
38
39 // Function to check if a string is an identifier (variable/function name)
40 int isIdentifier(char *word)
41 {
42     if (!isalpha(word[0]) && word[0] != '_')
43     {
44         return 0;
45     }
46     for (int i = 1; i < strlen(word); i++)
47     {
48         if (!isalnum(word[i]) && word[i] != '_')
49         {
50             return 0;
51         }
52     }
53     total_tokens++;
54     return 1;
55 }
56
57 // Function to check if a string is a number (integer or float)
58 int isNumber(char *word)
59 {
60     int i = 0;
61     int hasDecimal = 0;
62
63     // Check for negative sign at the start
64     if (word[0] == '-')
65     {
66         i = 1;
67     }
68     for (; i < strlen(word); i++)
69     {
70         if (word[i] == '.')
71         {
72             if (hasDecimal) // More than one decimal point is not valid
73                 return 0;
74             hasDecimal = 1;
75         }
76         else if (!isdigit(word[i]))
77         {
78             return 0;
79         }
80     }
81     total_tokens++;
82     return 1;
83 }
84

```

```

C question2.c x
C question2.c > analyzeTokens(FILE *)
58 int isNumber(char *word)
59 {
60     return word[0] != '0' && word[strlen(word) - 1] != '0';
61 }
62
63 // Function to check if a string is a string literal
64 int isStringLiteral(char *word)
65 {
66     return word[0] == '"' && word[strlen(word) - 1] == '"';
67 }
68
69 // Function to skip comments (single-line and multi-line)
70 void skipComments(FILE *file)
71 {
72     char c, next;
73
74     while ((c = fgetc(file)) != EOF)
75     {
76         if (c == '/' && (next = fgetc(file)) == '/') // Single-line comment
77         {
78             while ((c = fgetc(file)) != EOF && c != '\n'); // Skip until end of line
79         }
80         else if (c == '/' && next == '*') // Multi-line comment
81         {
82             while ((c = fgetc(file)) != EOF)
83             {
84                 if (c == '*' && (next = fgetc(file)) == '/') // End of multi-line comment
85                 {
86                     break;
87                 }
88             }
89         }
90         else
91         {
92             ungetc(c, file); // Push the character back for processing
93             break; // Exit the loop once we've reached a non-comment character
94         }
95     }
96 }

```

```
C question2.c x
C question2.c > analyzeTokens(FILE *)
119
120 // Function to analyze and print tokens
121 void analyzeTokens(FILE *file)
122 {
123     char word[MAX_LENGTH];
124     char c;
125     int i = 0;
126
127     while ((c = fgetc(file)) != EOF)
128     {
129         skipComments(file); // Skip comments
130         char d;
131         if (isalpha(c) || c == '_' ) // Start of a possible identifier or keyword
132         {
133             word[i++] = c;
134         }
135         else if (c == '"') // Start of a string literal
136         {
137             word[i++] = c;
138             while ((c = fgetc(file)) != EOF && c != '"') // Read string literal until closing quote
139             {
140                 word[i++] = c;
141             }
142             word[i++] = '\0';
143             word[i] = '\0'; // Null-terminate string literal
144             printf("String Literal: %s\n", word);
145             total_tokens++; // Increment token count for string literals
146             i = 0;
147         }
148         else if (isdigit(c) || (c == '-' && ( isdigit(d = fgetc(file))))) // Start of a number
149         {
150             word[i++] = c;
151             ungetc(d, file); // Push the character back for number processing
152             while ((c = fgetc(file)) != EOF && (isdigit(c) || c == '.'))
153             {
154                 word[i++] = c;
155             }
156             word[i] = '\0'; // Null-terminate the number
157             if (isNumber(word)) // Check if it's a valid number
158             {
159                 printf("Number      : %s\n", word); // Print the number token
160             }
161             i = 0; // Reset for the next token
162         }
163     }
}
```

```

C question2.c x
C question2.c > analyzeTokens(FILE *)
121 void analyzeTokens(FILE *file)
127 while ((c = fgetc(file)) != EOF)
148     else if (isdigit(c) || (c == '-' && (isdigit(d = fgetc(file))))) // Start of a number
162 }
163 else
164 {
165     if (i > 0) // End of an identifier or keyword
166     {
167         word[i] = '\0';
168         if (isKeyword(word)) // Check for keyword
169         {
170             printf("Keyword : %s\n", word);
171         }
172         else if (isIdentifier(word)) // Check for identifier
173         {
174             printf("Identifier: %s\n", word);
175         }
176         i = 0; // Reset for the next word
177     }
178
179     if (isOperator(c)) // Check for operator
180     {
181         printf("Operator : %c\n", c);
182     }
183 }
184 }
185 You, 41 minutes ago • LAB 1 initial code
186 printf("Total token count: %d\n", total_tokens);
187 }
188
189 int main(int argc, char *argv[])
190 {
191     if (argc != 2)
192     {
193         printf("Usage: %s <filename>\n", argv[0]);
194         return 1;
195     }
196
197     FILE *file = fopen(argv[1], "r");
198     if (file == NULL)
199     {
200         printf("Error opening file %s.\n", argv[1]);
201         return 1;
202     }
203
204     analyzeTokens(file);
205     fclose(file);

```

```
C question2.c x
C question2.c > analyzeTokens(FILE *)
121 void analyzeTokens(FILE *file)
122     while ((c = fgetc(file)) != EOF)
123         else
124             if (i > 0) // End of an identifier or keyword
125                 else if (isIdentifier(word)) // Check for identifier
126                     printf("Identifier: %s\n", word);
127             }
128             i = 0; // Reset for the next word
129         }
130     }
131     if (isOperator(c)) // Check for operator
132     {
133         printf("Operator : %c\n", c);
134     }
135 }
136
137 You, 41 minutes ago • LAB 1 initial code
138 printf("Total token count: %d\n", total_tokens);
139 }
140
141 int main(int argc, char *argv[])
142 {
143     if (argc != 2)
144     {
145         printf("Usage: %s <filename>\n", argv[0]);
146         return 1;
147     }
148
149     FILE *file = fopen(argv[1], "r");
150     if (file == NULL)
151     {
152         printf("Error opening file %s.\n", argv[1]);
153         return 1;
154     }
155
156     analyzeTokens(file);
157     fclose(file);
158     return 0;
159 }
```

Examples on which this code was tested:

```
C example1.c x
C example1.c > ...
You, 1 second ago | 1 author (You)
1 #include <stdio.h> You, 44 minutes ago • LAB 1 initial code
2 #include <ctype.h>
3 #include <string.h>
4 int main()
5 {
6     // comment added so that the compiler can ignore this
7     char s[] = "Hello World!";
8     printf("%s\n", s);
9     return 0;
10 }
11
```

```
C example2.c X
C example2.c > ...
You, 1 second ago | 1 author (You)
1 #include <stdio.h>
2 #include <ctype.h>
3 #include <string.h>
4 // this is a comment
5 int main()
6 {
7     int x = 1;
8     char s[] = "Hello World!";
9     printf("%s\n", s);
10    return 0;
11 }
12
```

Result:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POLYGLOT NOTEBOOK GITLENS SPELL CHECKER
• arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$ ./question2 example1.c
Operator : #
Identifier: include
Operator : <
Identifier: stdio
Operator : .
Identifier: h
Operator : >
Operator : #
Identifier: include
Operator : <
Identifier: ctype
Operator : .
Identifier: h
Operator : >
Operator : #
Identifier: include
Operator : <
Identifier: string
Operator : .
Identifier: h
Operator : >
Keyword : int
Identifier: main
Operator : (
Operator : )
Operator : {
Keyword : char
Identifier: s
Operator : =
String Literal: "Hello World!"
Operator : ;
Identifier: printf
Operator : (
String Literal: "%s\n"
Operator : ,
Identifier: s
Operator : )
Operator : ;
Keyword : return
Number : 0
Operator : ;
Operator : }
Total token count: 42
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POLYGLOT NOTEBOOK GITLENS SPELL CHECKER
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$ ./question2 example2.c
Operator : #
Identifier: include
Operator : <
Identifier: stdio
Operator : .
Identifier: h
Operator : >
Operator : #
Identifier: include
Operator : <
Identifier: ctype
Operator : .
Identifier: h
Operator : >
Operator : #
Identifier: include
Operator : <
Identifier: string
Operator : .
Identifier: h
Operator : >
Keyword : int
Identifier: main
Operator : (
Operator : )
Operator : {
Keyword : int
Identifier: x
Operator : =
Number : 1
Operator : ;
Keyword : char
Identifier: s
Operator : =
String Literal: "Hello World!"
Operator : ;
Identifier: printf
Operator : (
String Literal: "%s\n"
Operator : ,
Identifier: s
Operator : )
Operator : ;
Keyword : return
Number : 0
Operator : ;
Operator : }
Total token count: 47
```

## Question 3

Code:

```
C question3.c x
C question3.c > main(int, char *[])
You, 49 minutes ago | 1 author (You)
#include <stdio.h>
1
2
3 int main(int argc, char *argv[])
4 {
5     if (argc != 2)
6     {
7         printf("Usage: %s <filename>\n", argv[0]);
8         return 1;
9     }
10    FILE *file = fopen(argv[1], "r");
11    if (file == NULL)
12    {
13        printf("Error opening file %s.\n", argv[1]);
14        return 1;
15    }
16    char c;
17    int spaces = 0, lines = 0, characters = 0;
18    while ((c = fgetc(file)) != EOF)
19    {
20        characters++;
21    }
22    if (c == ' ')
23    {
24        spaces++;
25    }
26    if (c == '\n')
27    {
28        lines++;
29    }
30    }
31    fclose(file);
32    printf("Total characters: %d\n", characters);
33    printf("Total spaces: %d\n", spaces);
34    printf("Total lines ( if a line is defined as identifying new line special character): %d\n", lines);
35    printf("Total lines ( if a line is defined as per english language): %d\n", lines+1);
36    return 0;
37 }
38 }
```

Examples on which this code was tested on:

```
example1.txt x
example1.txt
1 hello im Arnav Jain
2 I belong to B tech department CSE IIT Indore.
```

```
example2.txt x
example2.txt
You, 50 minutes ago | 1 author (You)
1 hello
2
```

Result:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POLYGLOT NOTEBOOK GITLENS SPELL CHECKER 1
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$ ./question3 example1.txt
Total characters: 65
Total spaces: 11
Total lines ( if a line is defined as identifying new line special character): 1
Total lines ( if a line is defined as per english language): 2
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$ ./question3 example2.txt
Total characters: 6
Total spaces: 0
Total lines ( if a line is defined as identifying new line special character): 1
Total lines ( if a line is defined as per english language): 2
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiler/LAB 1$
```

To view the code files , refer github

<https://github.com/arnavjain2710/Compiler-Techniques/tree/main/LAB%201>