CS 358 - COMPILER TECHNIQUES LAB

WEEKLY ASSIGNMENT REPORT

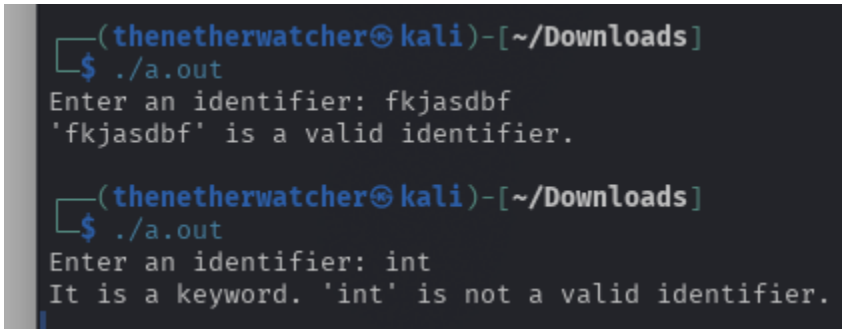
Problem 1:

```C/C++
#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdbool.h>



const char *keywords[] = {"auto", "break", "case", "char", "const", "continue",
"default", "do", "int", "long", "register", "return", "short", "signed",
"sizeof", "static", "struct", "switch", "typedef", "union", "unsigned", "void",
"volatile", "while", "for", "if", "else", "double", "else", "enum", "extern",
"float", "for", "goto", "if"};

int num_keywords = sizeof(keywords) / sizeof(keywords[0]);



int is_valid_identifier(const char *str) {

    bool flag=false;

    if (!(isalpha(str[0]) || str[0] == '_')) {

        return 0;

    }



    for (int i = 1; str[i] != '\0'; i++) {

        if (!(isalnum(str[i]) || str[i] == '_')) {
```

```
            return 0;
        }
    }

    for (int i = 0; i < num_keywords; i++) {
        if (strcmp(str, keywords[i]) == 0) {
            flag=true;
            printf("It is a keyword. ");
            return 0;
        }
    }

    return 1;
}

int main() {
    char identifier[100];

    printf("Enter an identifier: ");
    scanf("%s", identifier);

    if (is_valid_identifier(identifier)) {
        printf("'%s' is a valid identifier.\n", identifier);
```

```
    } else {
        printf("'%s' is not a valid identifier.\n", identifier);
    }
    return 0;
}
```

Output:



Problem 2:

```
C/C++
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

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);
}

// Returns 'true' if the string is a VALID IDENTIFIER.
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);
}


char* subString(char* str, int left, int right) {
    int i;
```

```
    char* subStr = (char*)malloc( sizeof(char) * (right - left + 2));
    for (i = left; i <= right; i++)
    subStr[i - left] = str[i];
    subStr[right - left + 1] = '\0';
    return (subStr);
}

void skipComment(char* str, int* left, int* right) {
    if (str[*left] == '/' && str[*left + 1] == '/') {
        // Skip single-line comment
        while (str[*right] != '\n' && str[*right] != '\0') {
            (*right)++;
        }
    } else if (str[*left] == '/' && str[*left + 1] == '*') {
        // Skip multi-line comment
        *right = *left + 2; // Skip '/*'
        while (!(str[*right] == '*' && str[*right + 1] == '/')) {
            (*right)++;
            if (str[*right] == '\0') break;
        }
        *right += 2; // Skip '*/'
    }
    *left = *right; // Move left pointer to the end of the comment
```

```
}


void detectTokens(char* str) {
   int left = 0, right = 0;
   int length = strlen(str);
   while (right <= length && left <= right) {
      if (str[left] == '/' && (str[left + 1] == '/' || str[left + 1] == '*')) {
         skipComment(str, &left, &right);  // Skip comment
         continue;
      }


      if (isValidDelimiter(str[right]) == false)
         right++;
      if (isValidDelimiter(str[right]) == true && left == right) {
         if (isValidOperator(str[right]) == true)
            printf("Valid operator : '%c'\n", str[right]);
         right++;
         left = right;
      } else if (isValidDelimiter(str[right]) == true && left != right || (right
== length && left != right)) {
         char* subStr = subString(str, left, right - 1);
         if (isValidKeyword(subStr) == true)
            printf("Valid keyword : '%s'\n", subStr);
```

```
        else if (isValidInteger(subStr) == true)
           printf("Valid Integer : '%s'\n", subStr);
        else if (isRealNumber(subStr) == true)
           printf("Real Number : '%s'\n", subStr);
        else if (isvalidIdentifier(subStr) == true
        && isValidDelimiter(str[right - 1]) == false)
           printf("Valid Identifier : '%s'\n", subStr);
        else if (isvalidIdentifier(subStr) == false
        && isValidDelimiter(str[right - 1]) == false)
           printf("Invalid Identifier : '%s'\n", subStr);
        left = right;
      }
   }
   return;
}

int main(){
   char str[100] = "/*float x = a + b; //abc*/";
   printf("The Program is : '%s' \n", str);
   printf("All Tokens are : \n");
   detectTokens(str);
   return (0);
}
```

Output:

```
┌──(thenetherwatcher㉿kali)-[~/Downloads]
└─$ gcc p2.c

┌──(thenetherwatcher㉿kali)-[~/Downloads]
└─$ ./a.out
The Program is : '/*float x = a + b; //abc*/'
All Tokens are :

┌──(thenetherwatcher㉿kali)-[~/Downloads]
└─$ gcc p2.c

┌──(thenetherwatcher㉿kali)-[~/Downloads]
└─$ ./a.out
The Program is : 'float x = a + b; //abc'
All Tokens are :
Valid keyword : 'float'
Valid Identifier : 'x'
Valid operator : '='
Valid Identifier : 'a'
Valid operator : '+'
Valid Identifier : 'b'
```

Problem 3:

```
C/C++

#include <stdio.h>

#include <ctype.h>



int main() {

    FILE *file;

    char filename[100];

    char ch;
```

```c
int space_count = 0;
int line_count = 0;
int char_count = 0;

printf("Enter the filename: ");
scanf("%s", filename);

file = fopen(filename, "r");
if (file == NULL) {
    printf("Error opening the file.\n");
    return 1;
}

while ((ch = fgetc(file)) != EOF) {
    char_count++;

    if (ch == ' ') {
        space_count++;
    }

    if (ch == '\n') {
        line_count++;
        char_count--;
```

```
        }
    }

    fclose(file);

    printf("Number of characters: %d\n", char_count);
    printf("Number of spaces: %d\n", space_count);
    printf("Number of lines: %d\n", line_count);

    return 0;
}
```

Output: