Oreofe Solarin
Email: ons8@case.edu
Course: CSDS 337 - Compiler Design
Instructor: Dr. Vipin Chaudhary

Problem Set - 2
ID: 3637932
Term: Spring 2024
Due Date: 14^{th} February, 2024

Number of hours delay for this Problem Set: 2
Cumulative number of hours delay so far: 3

I discussed this homework with:

SUBMISSION GUIDELINES: Submit a zip file that includes the written answers and the flex file for Problem 4.

Problem 1 - 5 points
Describe the language denoted by the following regular expression?
$(aa|bb)^*((ab|ba)(aa|bb)^*(ab|ba)(aa|bb)^*)^*$

Solution:

1. $(aa|bb)^*$: This part matches any number (including zero) of occurrences of the strings "aa" or "bb" in the given sequence.
2. $(ab|ba)$: This part matches either "ab" or "ba".
3. $(aa|bb)^*$: Similar to the first part, this matches any number of occurrences of "aa" or "bb" again.
4. $(ab|ba)$: Again, matches either "ab" or "ba".
5. $(aa|bb)^*$: Once more, matches any number of occurrences of "aa" or "bb".

The entire expression is encapsulated within $(...)^*$, as talked about above can repeat zero or more times. Overall, the language described by this regular expression consists of strings that start with any number of repetitions of "aa" or "bb", followed by alternating occurrences of "ab" or "ba" separated by any number of repetitions of "aa" or "bb".

Problem 2 - 35 points
Write regular definitions for the following languages:

a All strings of lowercase letters that contain the five vowels in reverse order.

b Binary strings that has at least 3 characters, and the third character is 0.

c Binary strings that has number of 0s which is a multiple of 3

d Binary strings that starts and ends with the same character

e Binary strings that has odd length

f Binary strings that starts with 0 and has odd length, or starts with 1 and has even length

g Binary strings whose length is at least 1 and at most 3

Solution:

a $[a-z]^*[o][e][u][i][a-z]^*$

b (0|1)(0|1)0(0|1)*

c (1*01*01*01*0)*1*0*

d (0|1)(0|1)*(0|1)

e (0|1)(0|1)*

f (0(00)*1|1(00)*0)(0|1)*

g (0|1|00|01|10|11|000|001|010|011|100|101|110|111)

Problem 3 - 10 points
Provide transition diagram as an NFA to recognize the language represented by $a|abb|a^*b^+$. Convert this NFA to a DFA and show all steps.

Solution:

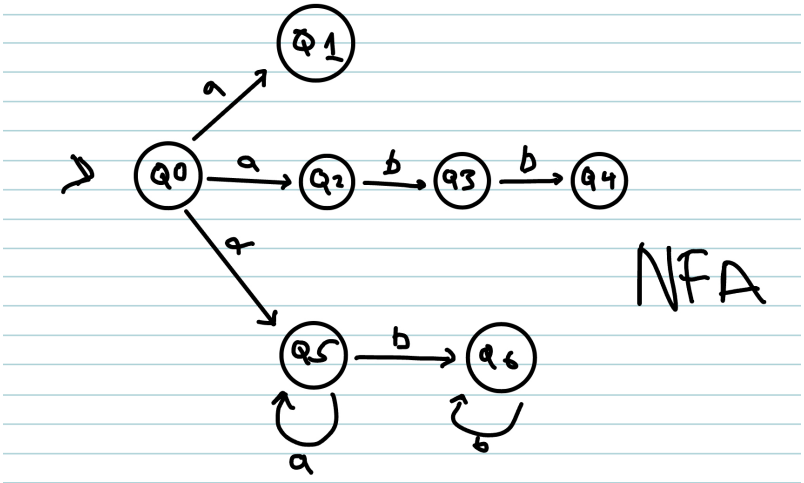

Figure 1: NFA

Table 1: State Transition Table

State	a	b
→ q0,5	q1,2,5	q6
* q6	-	q6
* q1,2,5	q5	q3,6
q5	q5	q6
* q3,6	-	q4,6
* q4,6	-	q6

Problem 4 - 50 points
Write a flex program which does the following:

- reads multiple input files
- for each file:
  - it prints the number of characters, number words and number of lines

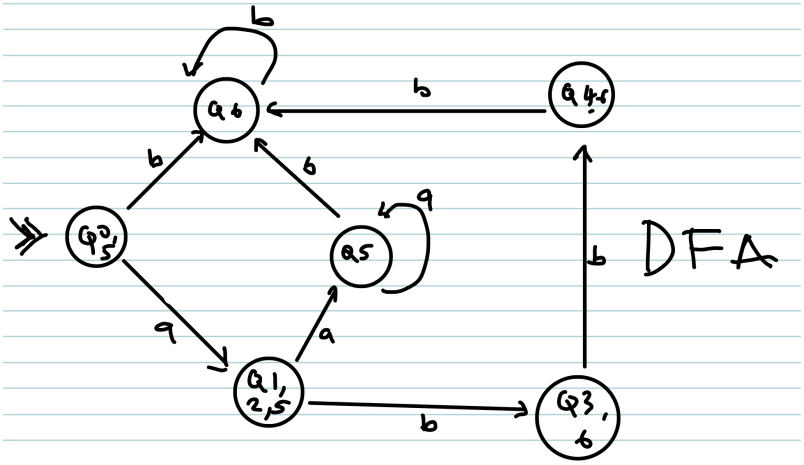

Figure 2: DFA

  - it replaces more than one contiguous space by a single space
  - it prints the number of single line C comments
  - it prints the number of multiple line C comments
  - it prints the number of occurrences of each of these keywords: *for*, *do*, and *while*
- all the above counts are printed for each file in order and a cumulative number for all the files is also printed at the end
- the entire output is printed to a file named "problem4output"
- the output should clearly indicate what each of the count indicates
- the flex file should be named "problem4lex.l"

Solution:

```
%{
#include <stdio.h>
int char_count = 0;
int word_count = 0;
int line_count = 0;
int single_line_comment_count = 0;
int multi_line_comment_count = 0;
int for_count = 0;
int do_count = 0;
int while_count = 0;

// Disable yywrap requirement
#define YY_NO_INPUT
%}

%x COMMENT

%%


"/*"            { multi_line_comment_count++; BEGIN(COMMENT); }
"//"            { single_line_comment_count++; }
```

```
<COMMENT>"*/"   { multi_line_comment_count++; BEGIN(INITIAL); }
<COMMENT>.|\n   { /* Ignoring characters inside multi-line comments */ }
[ \t]+          { /* Replace multiple spaces with a single space */ }
\n              { line_count++; }

for             { for_count++; }
do              { do_count++; }
while           { while_count++; }

[a-zA-Z]+       { word_count++;  char_count += yyleng;  }



%%

int yywrap() {
    return 1;
}

int main(int argc, char* argv[]) {
    FILE *input_file;
    int i;

    if (argc < 2) {
        printf("Usage: %s file1 file2 ... fileN\n", argv[0]);
        return 1;
    }

    FILE *output_file = fopen("problem4output", "w");
    if (output_file == NULL) {
        perror("Error opening output file");
        return 1;
    }

    for (i = 1; i < argc; ++i) {
        input_file = fopen(argv[i], "r");
        if (input_file == NULL) {
            perror("Error opening input file");
            return 1;
        }

        char_count = word_count = line_count = 0;
        single_line_comment_count = multi_line_comment_count = 0;
        for_count = do_count = while_count = 0;

        yyin = input_file;
        yylex();

        line_count++; // Count the last line

        fprintf(output_file, "File: %s\n", argv[i]);
        fprintf(output_file, "Number of characters: %d\n", char_count);
        fprintf(output_file, "Number of words: %d\n", word_count);
```

```
        fprintf(output_file, "Number of lines: %d\n", line_count);
        fprintf(output_file, "Number of single line C comments: %d\n", single_line_comment_count
        fprintf(output_file, "Number of multiple line C comments: %d\n", multi_line_comment_coun
        fprintf(output_file, "Number of 'for' occurrences: %d\n", for_count);
        fprintf(output_file, "Number of 'do' occurrences: %d\n", do_count);
        fprintf(output_file, "Number of 'while' occurrences: %d\n", while_count);
        fprintf(output_file, "\n");

        fclose(input_file);
    }

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