DATE:30/07/2025

EXP:06

LEX PROGRAMS AND DFA CONSTRUCTION+VALIDATION

Q1:

Write a lex program to validate consecutive vowels and introduce '/' in between the vowels.

CODE:

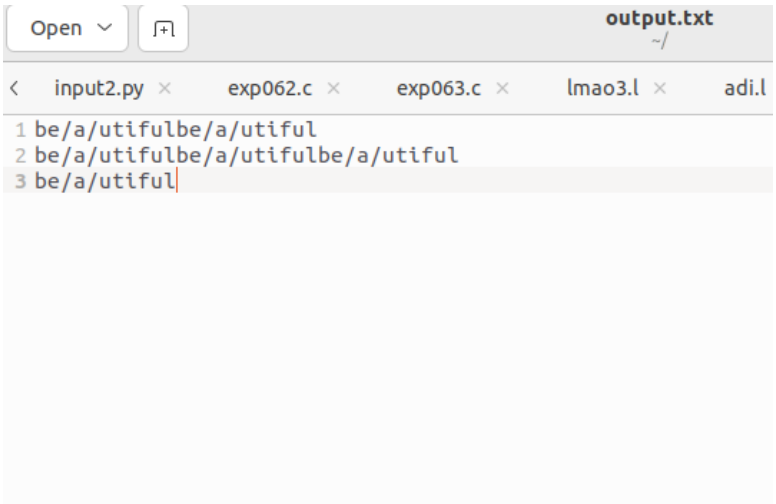
```
%option noyywrap
%{
#include <stdio.h>
FILE *out;
int is_vowel(char c) {
   return (c=='a'||c=='e'||c=='i'||c=='o'||c=='u'||c=='A'||c=='E'||c=='I'||c=='O'||c=='U');
}
%}

%%
.|\n {
   static int prev_vowel = 0;
   if (is_vowel(yytext[0])) {
      if (prev_vowel) {
         fprintf(out, "/");
      }
      prev_vowel = 1;
   } else {
      prev_vowel = 0;
   }
   fprintf(out, "%c", yytext[0]);
}
%%

int main() {
   out = fopen("output.txt", "a"); // open output file in append mode
   if (!out) {
      perror("output.txt");
      return 1;
   }
```

```
    yylex();
    fclose(out); // close the file after processing
    return 0;
```

OUTPUT:



2. Construct the DFA and validate the strings which contain ‘aab’.

CODE:

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

#define MAX_STATES 4
#define MAX_SYMBOLS 2 // 0 = 'a', 1 = 'b'

// Map input symbol to index
int getSymbol(char c) {
    if (c == 'a') return 0;
    if (c == 'b') return 1;
    return -1;
}

void printTransitionTable(int transitions[MAX_STATES][MAX_SYMBOLS]) {
    printf("\n--- Transition Table (for DFA: Contains 'aab') ---\n");
    printf("State\ta\tb\n");
    for (int i = 0; i < MAX_STATES; i++) {
        printf("q%d\t", i);
        for (int j = 0; j < MAX_SYMBOLS; j++) {
```

```
            if (transitions[i][j] == -1)
                printf("-\t");
            else
                printf("q%d\t", transitions[i][j]);
        }
        printf("\n");
    }
}

int main() {
    // DFA transitions
    int transitions[MAX_STATES][MAX_SYMBOLS] = {
        {1, 0}, // q0
        {2, 0}, // q1
        {2, 3}, // q2
        {3, 3}  // q3 (accept)
    };

    int acceptState = 3;
    int currentState = 0;
    char input[100];

    printf("Enter input string (only 'a' and 'b'): ");
    scanf("%s", input);

    printTransitionTable(transitions);
    printf("\nStep-by-step transitions:\n");

    for (int i = 0; i < strlen(input); i++) {
        int symbol = getSymbol(input[i]);
        if (symbol == -1) {
            printf("Invalid symbol '%c'\n", input[i]);
            return 1;
        }
        int nextState = transitions[currentState][symbol];
        printf("q%d --%c--> q%d\n", currentState, input[i], nextState);
        currentState = nextState;
    }

    if (currentState == acceptState)
        printf("\n Accepted: String contains 'aab'\n");
    else
        printf("\n Rejected: String does not contain 'aab'\n");

    return 0;
}
```

OUTPUT:

```
ubuntu@unix-Veriton-M200-H610:~$ gcc exp062.c -o adi
ubuntu@unix-Veriton-M200-H610:~$ ./adi
Enter input string (only 'a' and 'b'): ababaab

--- Transition Table (for DFA: Contains 'aab') ---
State   a       b
q0      q1      q0
q1      q2      q0
q2      q2      q3
q3      q3      q3

Step-by-step transitions:
q0 --a--> q1
q1 --b--> q0
q0 --a--> q1
q1 --b--> q0
q0 --a--> q1
q1 --a--> q2
q2 --b--> q3

✅ Accepted: String contains 'aab'
```

3. Construct DFA and validate the strings for the language $\{a^m b^n, m>=1 \ \& \ n\%3=1\}$.

CODE:

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

#define MAX_STATES 5
#define MAX_SYMBOLS 2 // 0 = 'a', 1 = 'b'

// DFA structure
typedef struct {
    int transitions[MAX_STATES][MAX_SYMBOLS];
    int acceptStates[MAX_STATES];
    int startState;
    int stateCount;
} DFA;
```

```
// Symbol mapping: 'a' = 0, 'b' = 1
int getSymbol(char c) {
    if (c == 'a') return 0;
    if (c == 'b') return 1;
    return -1;
}

// Construct DFA for the language a^m b^n where m ≥ 1 and n mod 3 = 1
void constructDFA(DFA *dfa) {
    dfa->stateCount = 5;
    dfa->startState = 0;

    // Initialize transitions to -1
    for (int i = 0; i < MAX_STATES; i++) {
        for (int j = 0; j < MAX_SYMBOLS; j++) {
            dfa->transitions[i][j] = -1;
        }
        dfa->acceptStates[i] = 0;
    }

    /*
        States:
        q0: Start (no input) → Rejects if b comes first
        q1: At least one a (a+)
        q2: a+ followed by b (n mod 3 = 1) → Accept
        q3: a+ followed by bb (n mod 3 = 2)
        q4: a+ followed by bbb (n mod 3 = 0)
    */

    dfa->transitions[0][0] = 1; // q0 --a--> q1
    dfa->transitions[0][1] = 0; // Invalid

    dfa->transitions[1][0] = 1; // q1 --a--> q1
    dfa->transitions[1][1] = 2; // q1 --b--> q2 (mod 1)

    dfa->transitions[2][0] = -1; // b followed by a → invalid
    dfa->transitions[2][1] = 3;  // q2 --b--> q3

    dfa->transitions[3][0] = -1;
    dfa->transitions[3][1] = 4; // q3 --b--> q4

    dfa->transitions[4][0] = -1;
    dfa->transitions[4][1] = 2; // q4 --b--> q2 (cycle)

    // Set accepting state
    dfa->acceptStates[2] = 1; // q2 is accepting
}
```

```
// Run input on DFA
int runDFA(DFA *dfa, const char *input) {
    int currentState = dfa->startState;

    for (int i = 0; input[i] != '\0'; i++) {
        int symbol = getSymbol(input[i]);
        if (symbol == -1) return 0; // Invalid symbol
        currentState = dfa->transitions[currentState][symbol];
        if (currentState == -1) return 0; // Invalid transition
    }

    return dfa->acceptStates[currentState];
}

// Print DFA transition table
void printTransitionTable(DFA *dfa) {
    printf("\n--- DFA Transition Table for L = { a^m b^n | m ≥ 1, n mod 3 = 1 } ---\n");
    printf("States: q0 to q4\n");
    printf("Start state: q0\n");
    printf("Accept state: q2\n\n");

    printf("State\ta\tb\n");
    for (int i = 0; i < dfa->stateCount; i++) {
        printf("q%d\t", i);
        for (int j = 0; j < MAX_SYMBOLS; j++) {
            int target = dfa->transitions[i][j];
            if (target == -1)
                printf("-\t");
            else
                printf("q%d\t", target);
        }
        printf("\n");
    }
}

int main() {
    DFA dfa;
    constructDFA(&dfa);
    printTransitionTable(&dfa);

    char input[100];
    printf("\nEnter a string (only a's followed by b's): ");
    scanf("%s", input);

    if (runDFA(&dfa, input))
        printf("Valid: String belongs to the language.\n");
```

```
    else
        printf(" Invalid: String does NOT belong to the language.\n");

    return 0;
}
```

OUTPUT:

```
ubuntu@unix-Veriton-M200-H610:~$ gcc exp063.c -o adi
ubuntu@unix-Veriton-M200-H610:~$ ./adi

--- DFA Transition Table for L = { a^m b^n | m ≥ 1, n mod 3 = 1 } ---
States: q0 to q4
Start state: q0
Accept state: q2

State   a       b
q0      q1      q0
q1      q1      q2
q2      -       q3
q3      -       q4
q4      -       q2

Enter a string (only a's followed by b's): aaabbbb
Valid: String belongs to the language.
ubuntu@unix-Veriton-M200-H610:~$ ./adi

--- DFA Transition Table for L = { a^m b^n | m ≥ 1, n mod 3 = 1 } ---
States: q0 to q4
Start state: q0
Accept state: q2

State   a       b
q0      q1      q0
q1      q1      q2
q2      -       q3
q3      -       q4
q4      -       q2

Enter a string (only a's followed by b's): abbb
 Invalid: String does NOT belong to the language.
```

RESULT:

The programs have been completed and the outputs have been verified.