Name: Pradyumn R Pai
Roll No: 50
Class: CS7A

PROGRAM CODE

dfa_ds.c:

```
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include "dsu.c"

struct DFA {
    int stateNum;
    bool* finalState;
    char * inputAlphabet;
    int ** transitionTable;
};

void freeDFA(struct DFA* dfa){
    if (!dfa) return;
    if (dfa->finalState){
        free(dfa->finalState);
    }
    if (dfa->transitionTable){
        int n = dfa->stateNum;
        for (int i=0;i<n;++i){
            if (dfa->transitionTable[i]){
                free(dfa->transitionTable[i]);
            }
        }
        free(dfa->transitionTable);
    }
    free(dfa);
```

```
}

struct DFA* init_DFA(int n, char* inputAlphabet){
    struct DFA* out = malloc(sizeof(struct DFA));
    if (!out){
        return NULL; //failed allocation
    }

    out->stateNum = n;
    out->inputAlphabet = inputAlphabet;
    int m = strlen(inputAlphabet);

    out->transitionTable = malloc(sizeof(int*)*n);
    if (!out->transitionTable){
        freeDFA(out);
        return NULL;
    }

    out->finalState = malloc(sizeof(bool)*n);
    if (!out->finalState){
        freeDFA(out);
    }
    for (int i=0;i<n;++i){
        out->finalState[i] = false;
    }

    for (int i=0;i<n;++i){
        out->transitionTable[i] = malloc(sizeof(int)*m);
        if (!out->transitionTable[i]){
            freeDFA(out);
            return NULL;
```

```
        }
        for (int j=0;j<m;++j){
            out->transitionTable[i][j] = i;
        }
    }

    return out;
}

int inputIndexDFA(struct DFA* dfa, char c){
    int m = strlen(dfa->inputAlphabet);
    for (int i=0;i<m;++i){
        if (c==dfa->inputAlphabet[i]){
            return i;
        }
    }
    return -1;
}

void addTransitionDFA(struct DFA* dfa, int s, int t, char c){
    int i = inputIndexDFA(dfa,c);
    if (i!=-1){
        dfa->transitionTable[s][i] = t;
    }
}



void printDFA(struct DFA* dfa){
    printf("The transition table is as follows:\n");
    int n = dfa->stateNum;
    int m = strlen(dfa->inputAlphabet);
```

```
    printf("\t");
    for (int i=0;i<m;++i){
        printf("%c\t",dfa->inputAlphabet[i]);
    }
    printf("\n");
    for (int i=0;i<n;++i){
        if (i==0){
            printf("->");
        }
        if (dfa->finalState[i]){
            printf("*");
        }
        printf("q%d\t",i);
        for (int j=0;j<m;++j){
            printf("q%d\t",dfa->transitionTable[i][j]);
        }
        printf("\n");
    }
}

struct DFA* readDFA() {
    // read input
    int n, f, m;
    scanf("%d%d%d", &n, &f, &m);
    if (f<0 || f>n){
        printf("Invalid number of final states\n");
        return NULL;
    }

    int finalStates[f];
    for (int i=0;i<f;++i){
```

```
    scanf("%d",finalStates+i);
    if (finalStates[i]<0 || finalStates[i]>=n){
        printf("Invalid final state %d\n",finalStates[i]);
        return NULL;
    }
}


char* inputChars = malloc(sizeof(char)*(m+1));
if (!inputChars) {
    printf("Failed to allocate memory for input characters\n");
    return NULL;
}

scanf("%s\n", inputChars);
if (strlen(inputChars) != m) {
    free(inputChars);
    printf("Input characters length mismatch\n");
    return NULL;
}

struct DFA *dfa = init_DFA(n,inputChars);
if (!dfa) {
    free(inputChars);
    printf("Failed to initialize DFA\n");
    return NULL;
}

for (int i=0;i<f;++i){
    dfa->finalState[finalStates[i]] = true;
}
```

```
    for (int i=0;i<n;++i){
        for (int j=0;j<m;++j) {
            scanf("%d",dfa->transitionTable[i]+j);
        }
    }

    return dfa;
}
```

dfa_minimization.c:

```
#include <stdio.h>
#include "dfa_ds.c"

int main(){
    struct DFA* dfa = readDFA();
    if (!dfa){
        printf("DFA initialization failed\n");
        return 1;
    }
    printDFA(dfa);

    struct DFA* minimizeddfa = dfsMinimization(dfa);
    if (!minimizeddfa){
        printf("DFA minimization failed\n");
        return 1;
    }
    printf("\n\nThe minimized dfa is:\n");
    printDFA(minimizeddfa);

    freeDFA(dfa);
    freeDFA(minimizeddfa);
```

```
}
```

OUTPUT:

input.txt:

6 3 2

1 2 4

01

3 1

2 5

2 5

0 4

2 5

5 5

output:

The transition table is as follows:

	0	1
->q0	q3	q1
*q1	q2	q5
*q2	q2	q5
q3	q0	q4
*q4	q2	q5
q5	q5	q5

The minimized dfa is:

The transition table is as follows:

	0	1
->q0	q0	q1
*q1	q1	q5
q2	q5	q5