Name: Gokulnath M Prabhu
Class: CS7B
Roll No: 21

Lab Cycle 3 - Experiment 13

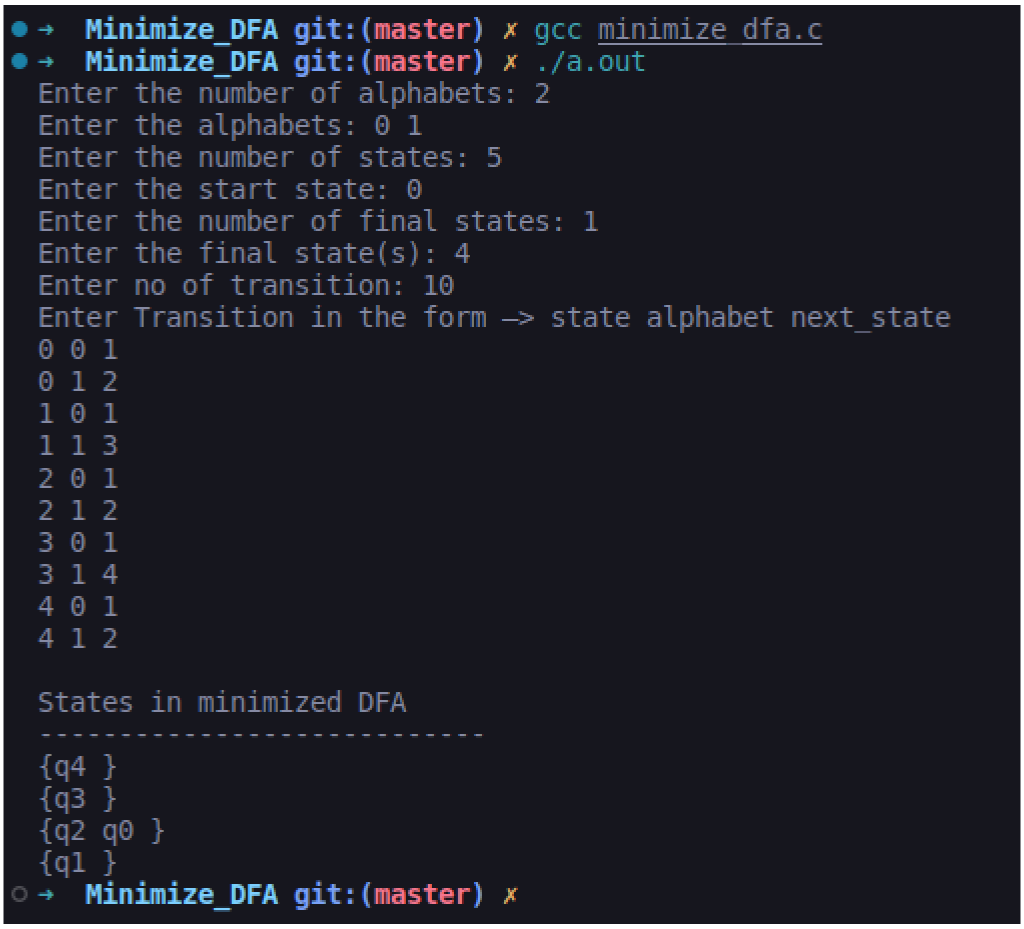
Write a program to minimize any given DFA.

Code:

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

static int nostate, noalpha, s, notransition, nofinal, start,
finalstate[20], r;
char alphabet[20];
int transition_map[30][30], table[30][30], nonfinalstate[20],
partition[20][20];

int findalpha(char a)
{
   int i;
   for (i = 0; i < noalpha; i++)
       if (alphabet[i] == a)
           return i;
   return (-1);
}

int main()
{
   int i, j, p[20], q[20], k;
   char a;
   for (i = 0; i < 30; i++)
   {
       for (j = 0; j < 30; j++)
           transition_map[i][j] = -1;
   }
   printf("Enter the number of alphabets: ");
   scanf("%d", &noalpha);
   getchar();
   printf("Enter the alphabets: ");
   for (i = 0; i < noalpha; i++)
   {
       alphabet[i] = getchar();
       getchar();
   }
   printf("Enter the number of states: ");
```

```
    scanf("%d", &nostate);
    printf("Enter the start state: ");
    scanf("%d", &start);
    printf("Enter the number of final states: ");
    scanf("%d", &nofinal);
    printf("Enter the final state(s): ");
    for (i = 0; i < nofinal; i++)
        scanf("%d", &finalstate[i]);
    printf("Enter no of transition: ");
    scanf("%d", &notransition);
    printf("Enter Transition in the form -> state alphabet
next_state\n");
    for (i = 0; i < notransition; i++)
    {
        scanf("%d %c %d", &r, &a, &s);
        j = findalpha(a);
        if (j == -1)
        {
            printf("\nerror\n");
            exit(1);
        }
        transition_map[r][j] = s;
    }
    for (i = 0; i < nostate; i++)
    {
        for (j = 0; j < i; j++)
        {
            table[i][j] = 0;
        }
    }
    int f = 0;
    k = 0;
    for (i = 0; i < nostate; i++)
    {
        f = 0;
        for (j = 0; j < nofinal; j++)
        {
            if (i == finalstate[j])
            {
                f = 1;
                break;
            }
        }
        if (f == 0)
        {
            nonfinalstate[k++] = i;
        }
    }
    for (i = 0; i < nofinal; i++)
    {
        for (j = 0; j < (nostate - nofinal); j++)
            if (nonfinalstate[j] > finalstate[i])
                table[nonfinalstate[j]][finalstate[i]] = 1;
            else
                table[finalstate[i]][nonfinalstate[j]] = 1;
    }
    int change = 1;
    while (change == 1)
    {
        change = 0;
        for (i = 0; i < nostate; i++)
        {
            for (j = 0; j < i; j++)
            {
                if (table[i][j] != 1)
                {
                    for (k = 0; k < noalpha; k++)
                        p[k] = transition_map[i][k];
                    for (k = 0; k < noalpha; k++)
                        q[k] = transition_map[j][k];
                    for (k = 0; k < noalpha; k++)
                    {
                        if (p[k] > q[k])
                        {
                            if (table[p[k]][q[k]] == 1)
                            {
                                change = 1;
                                table[i][j] = 1;
                                break;
                            }
                        }
                        else if (p[k] < q[k])
                        {
                            if (table[q[k]][p[k]] == 1)
                            {
                                change = 1;
                                table[i][j] = 1;
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
    k = 0;
    for (i = 0; i < nostate; i++)
    {
        k = 0;
        partition[i][k++] = i;
        for (j = 0; j < i; j++)
            if (table[i][j] == 0)
                partition[i][k++] = j;
        partition[i][k] = -1;
    }
    int newstate[20] = {0}, m;
    printf("\nStates in minimized DFA");
    printf("\n----------------------------\n");
    for (i = nostate - 1; i >= 0; i--)
    {
        k = 0;
        if (newstate[i] == 0)
        {
            printf("{");
            while (partition[i][k] != -1)
            {
                if (newstate[partition[i][k]] == 0)
                {
                    newstate[partition[i][k]] = 1;
                    printf("q%d ", partition[i][k]);
                }
                k++;
            }
            printf("}\n");
        }
    }
    return 0;
}
```

Output:



```
● ➜  Minimize_DFA git:(master) ✗ gcc minimize_dfa.c
● ➜  Minimize_DFA git:(master) ✗ ./a.out
  Enter the number of alphabets: 2
  Enter the alphabets: 0 1
  Enter the number of states: 5
  Enter the start state: 0
  Enter the number of final states: 1
  Enter the final state(s): 4
  Enter no of transition: 10
  Enter Transition in the form –> state alphabet next_state
  0 0 1
  0 1 2
  1 0 1
  1 1 3
  2 0 1
  2 1 2
  3 0 1
  3 1 4
  4 0 1
  4 1 2

  States in minimized DFA
  ----------------------------
  {q4 }
  {q3 }
  {q2 q0 }
  {q1 }
○ ➜  Minimize_DFA git:(master) ✗
```