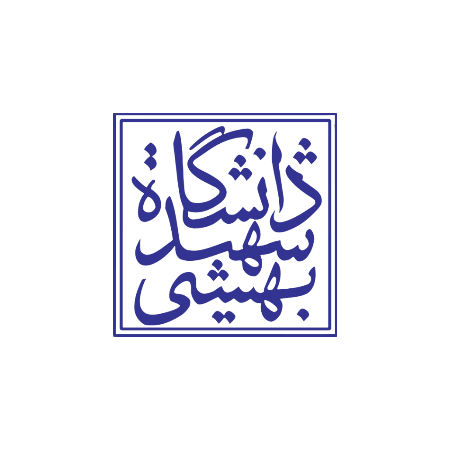
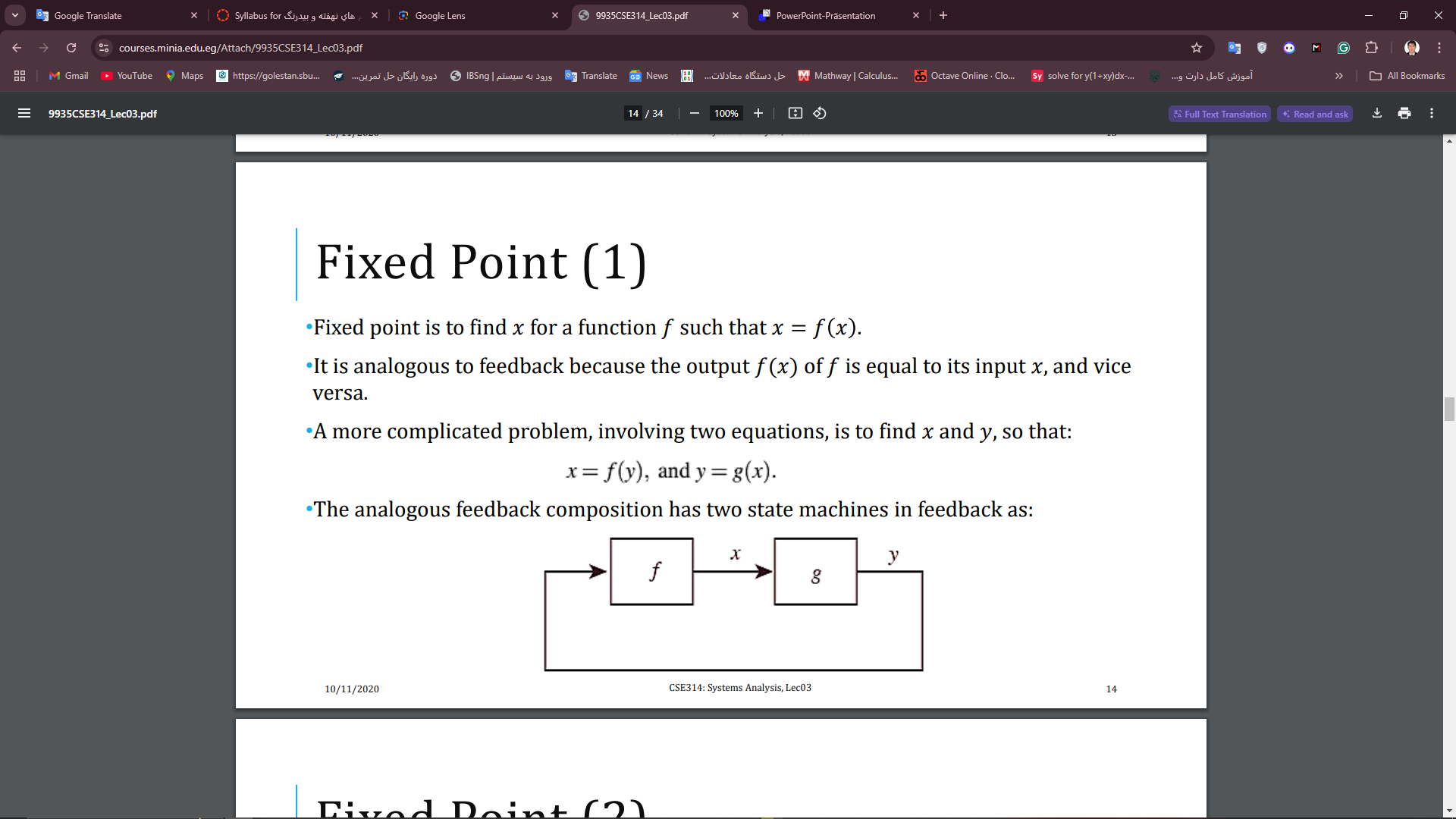
**گرازش تمرین 3**

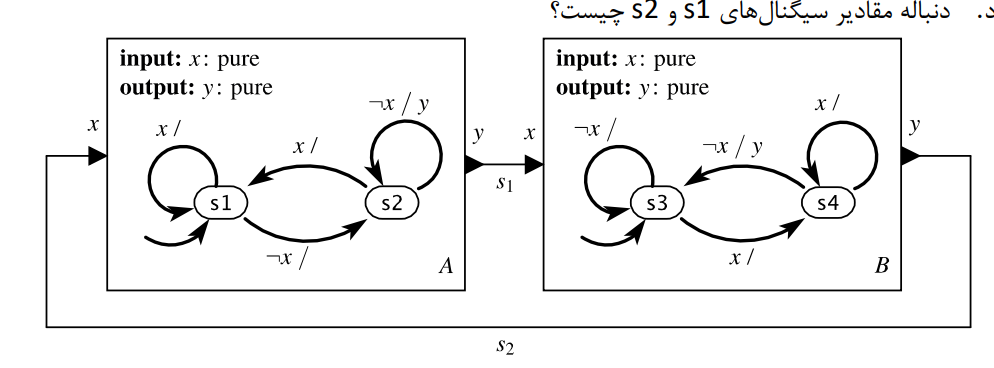
**بسم الله الرحمن الرحيم**

**درس مبانی سیستم نهفته و بیدرنگ**

**جواب سوال 1**  
 **الف)**

برای برسی خوش ساخت باید fixed point را برای تمام حالات برسی کنیم. اگر بتوانیم برای هر مدل fixed point پیدا کنیم میتواند گفت که مدل خوش ساخت است



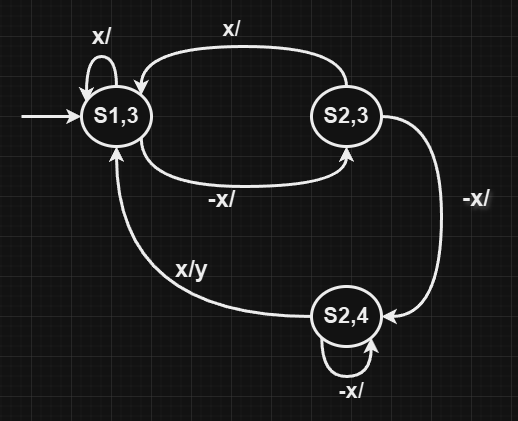
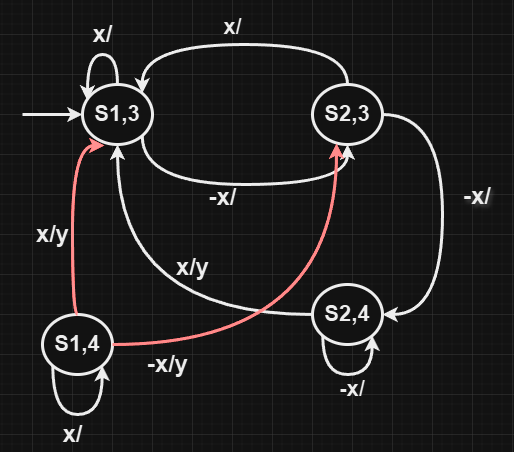


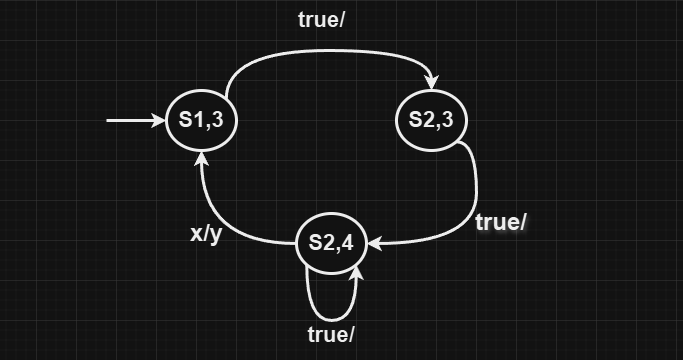
**برسی Fixed Point )**

حالت های **6 و 8** شرط یکتا بودن Fixed Point را برهم میزند و مدل **Ill**-**formed** ***ا***ست

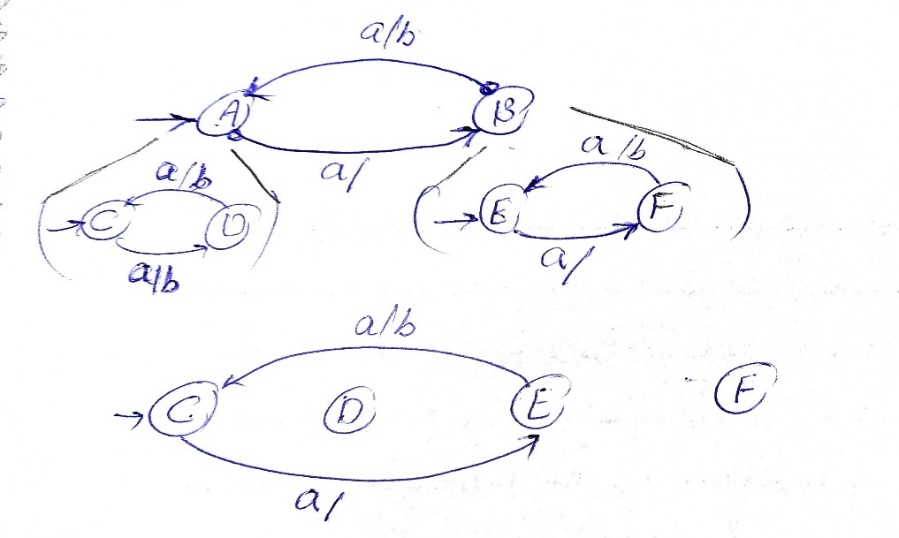
**ب)**چون مدل خوش ساخت نیست پس برساختنی هم نیست.

**ج)**State **S1,4** قابل درسترس نیست پس این اسیت را حذف می کنیم. پس داریم:





**د)**

**جواب سوال 2**

الف) یک FSM مسطح معادل رفتار آن ارائه کنید.

ب) این ماشین چه حالتهای دسترس پذیری دارد؟

چون ماشین حالت از نوع Preemptive می باشد هیچگاه وارد حالت های D و F نمی شود.

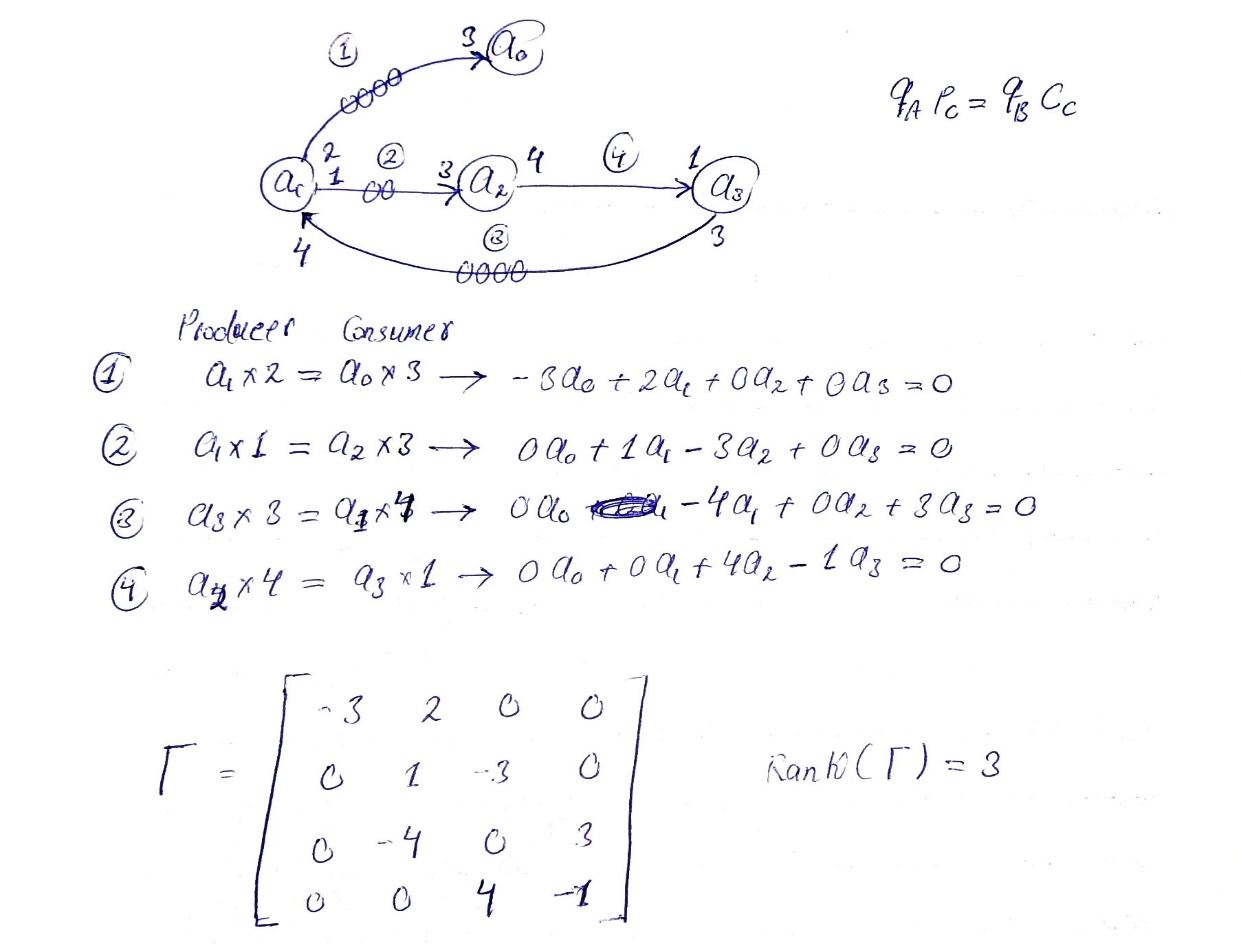
پس با هر بار ورودی به حالت های Eو C دسترسی پذیری دارد.

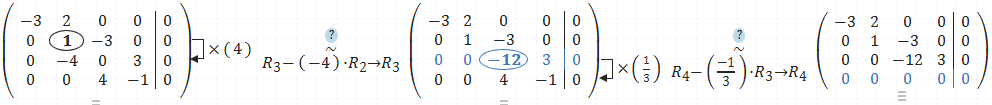
ج) رفتار ورودی/خروجی این ماشین را در یک جمله توصیف کنید.

این ماشین حالت با هر بار گرفتن ورودی a یک در میان b خروجی میدهد.

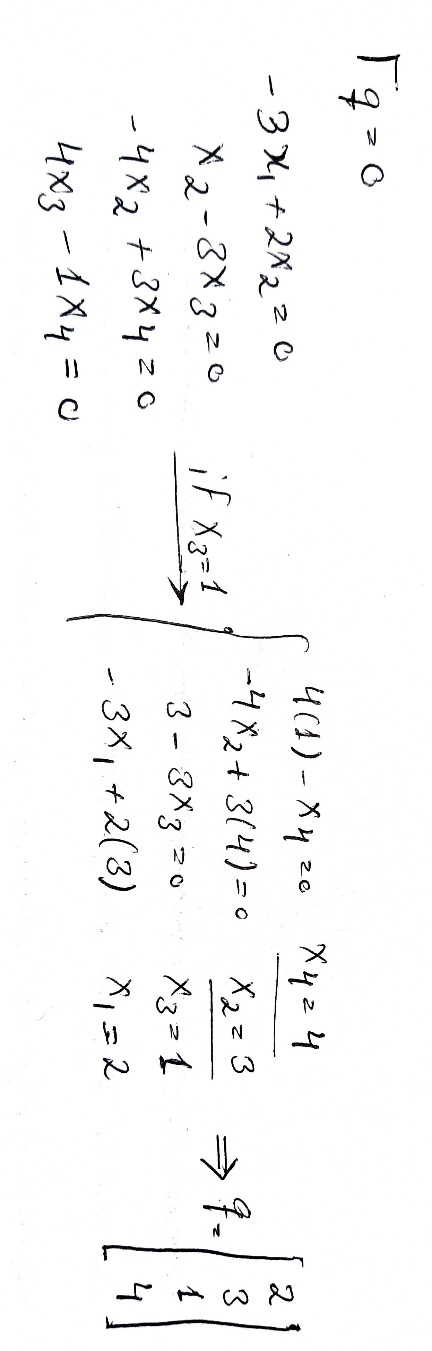
**جواب سوال 3**

الف) ماتریس وقوع (ماتریس توپولوژی) گراف Γ را استخراج کنید و مرتبه آن را مشخص نمایید.

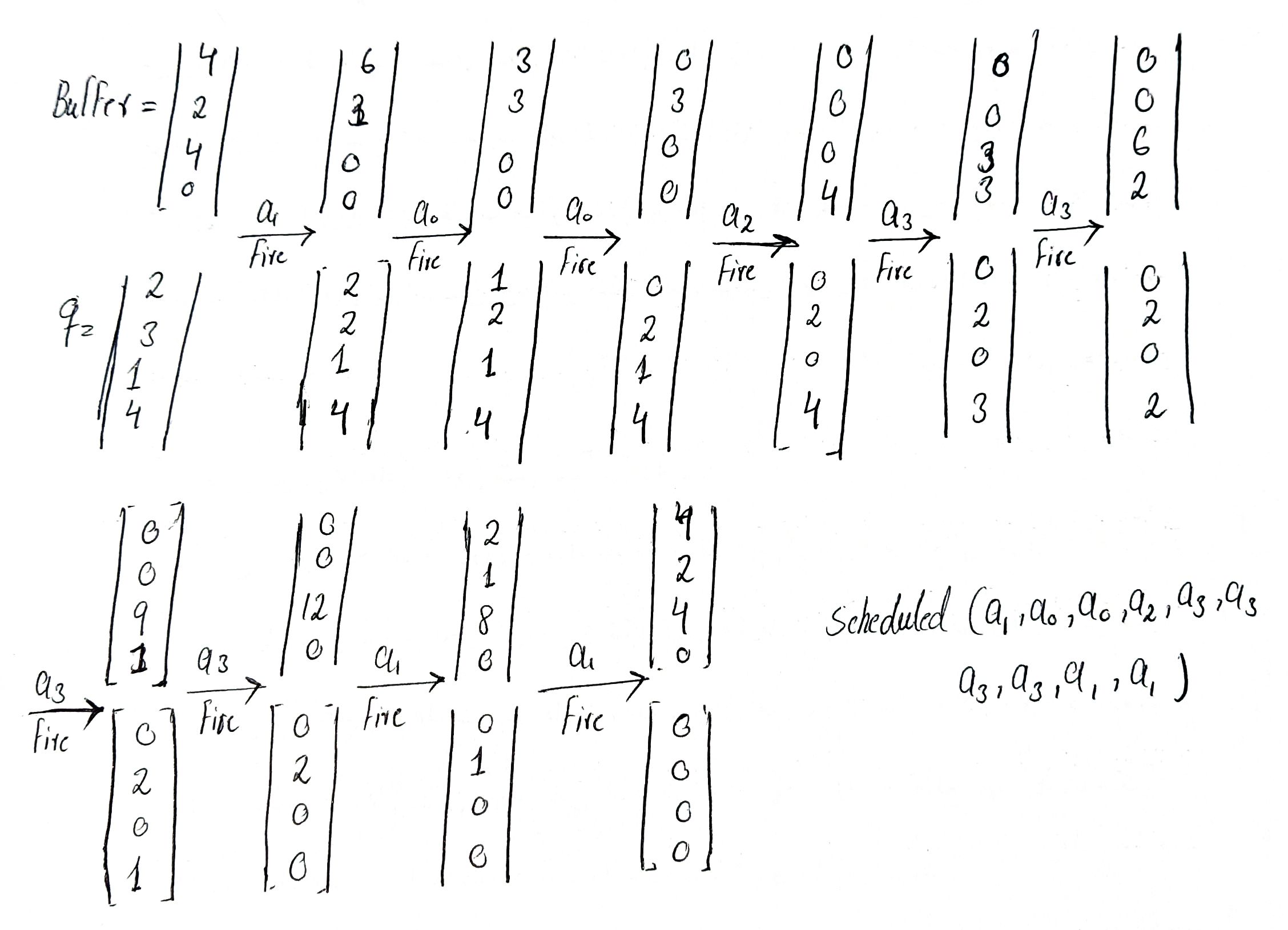




ب) مقدار q با کوچکترین مقادیر صحیح مثبت را به دست آورید به گونه ای که 0=Γq باشد.



ج) برای این گراف SDF یک زمانبندی تکرارشونده ترتیبی را به همراه شرایط اولیه بافرها مشخص کنید. حداقل مجموع بافرهای مورد نیاز برای اجرای زمانبندی به دست آمده چقدر است؟



The Schedule sequence is (a1, a0, a0, a2, a3, a3, a3, a3, a1, a1)

Minimum buffer for every edge is = | max (4, 6, 3, 0, 0, 0, 0, 0, 0, 2, 1) = 6 |

| max (2, 3, 3, 3, 0, 0, 0, 0, 0, 1, 2) = 3 |

| max (4, 0, 0, 0, 0, 3, 6, 9, 12, 8, 4) = 12 |

| max (0, 0, 0, 0, 4, 3, 2, 1, 0, 0, 0) = 4 |

در نتیجه به حد اقل بافر های به سایز 6, 3, 12, 4 نیاز هست.

د) با فرض موجود بودن توابع , a3(), a2(), a1()()4a، کد حلقه زمانبندی تکرار شونده گراف را به زبان C بنویسید. از آرایه ها برای پیاده سازی بافرها استفاده کنید.

**جواب:**

#include <stdio.h>

void a0();

void a1();

void a2();

void a3();

void printBuffer();

// declare buffers

int buffer1[6];     // a1 -> a0

int buffer2[3];     // a1 -> a2

int buffer3[12];    // a3 -> a1

int buffer4[4];     // a2 -> a3

// pointer for writer and reader on buffers

int index\_b1 = 4, index\_b2 = 2, index\_b3 = 4, index\_b4 = 0;

int main() {

// initialization buffers

    for(int i = 0; i < 4; i++){

        buffer1[i] = 1;

    }

    for(int i = 0; i < 2; i++){

        buffer2[i] = 1;

    }

    for(int i = 0; i < 4; i++){

        buffer4[i] = 1;

    }

    // print valuse all buffer initia

    printf("frist valuse: ");

    printBuffer();

    while(true){

        // these functions update all buffer, wb and rb

        a1();

            printf("fire a1: ");

            printBuffer();

        a0();

            printf("fire a0: ");

            printBuffer();

        a0();

            printf("fire a0: ");

            printBuffer();

        a2();

            printf("fire a2: ");

            printBuffer();

        a3();

            printf("fire a3: ");

            printBuffer();

        a3();

            printf("fire a3: ");

            printBuffer();

        a3();

            printf("fire a3: ");

            printBuffer();

        a3();

            printf("fire a3: ");

            printBuffer();

        a1();

            printf("fire a1: ");

            printBuffer();

        a1();

            printf("fire a3: ");

            printBuffer();

        printf("-------> End of Trun <-------");

        printf("\n------------------------------\n");

        i++;

    }

    return 0;

}

void a0(){

// counsume

    for (int i = 0; i < 3 ; i++)

    {

        index\_b1 = index\_b1-1;

        buffer1[index\_b1] = 0;

    }

}

void a1(){

// counsume

    for (int i = 0; i < 4 ; i++){

        index\_b3 = index\_b3-1;

        buffer3[index\_b3] = 0;

    }

// Produce a1 to a0

    for (int i = 0; i < 2 ; i++){

        buffer1[index\_b1] = 1;

        index\_b1 = index\_b1 + 1;

    }

// Produce a1 to a2

    for (int i = 0; i < 1 ; i++){

        buffer2[index\_b2] = 1;

        index\_b2 = index\_b2 + 1;

    }

}

void a2(){

// counsume

    for (int i = 0; i < 3 ; i++){

        index\_b2 = index\_b2-1;

        buffer2[index\_b2] = 0;

    }

// Produce a1 to a0

    for (int i = 0; i < 4 ; i++){

        buffer4[index\_b4] = 1;

        index\_b4 = index\_b4 + 1;

    }

}

void a3(){

// counsume

    index\_b4 = index\_b4-1;

    buffer4[index\_b4] = 0;

// Produce a3 to a1

    for (int i = 0; i < 3 ; i++){

        buffer3[index\_b3] = 1;

        index\_b3 = index\_b3 + 1;

    }

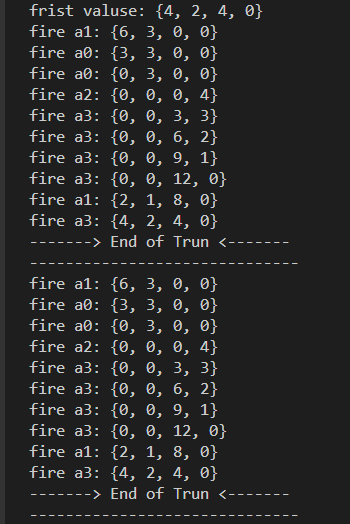
}

void printBuffer(){

 printf("{%d, %d, %d, %d}\n", index\_b1, index\_b2, index\_b3, index\_b4);

}

خروجی:



**قسمت امتیازی:**   
با استفاده از امکانات domain dataflow سعی میکنیم شبیه سازی بکنیم. هر node را یک subsystem dataflow قرار میدهیم.

