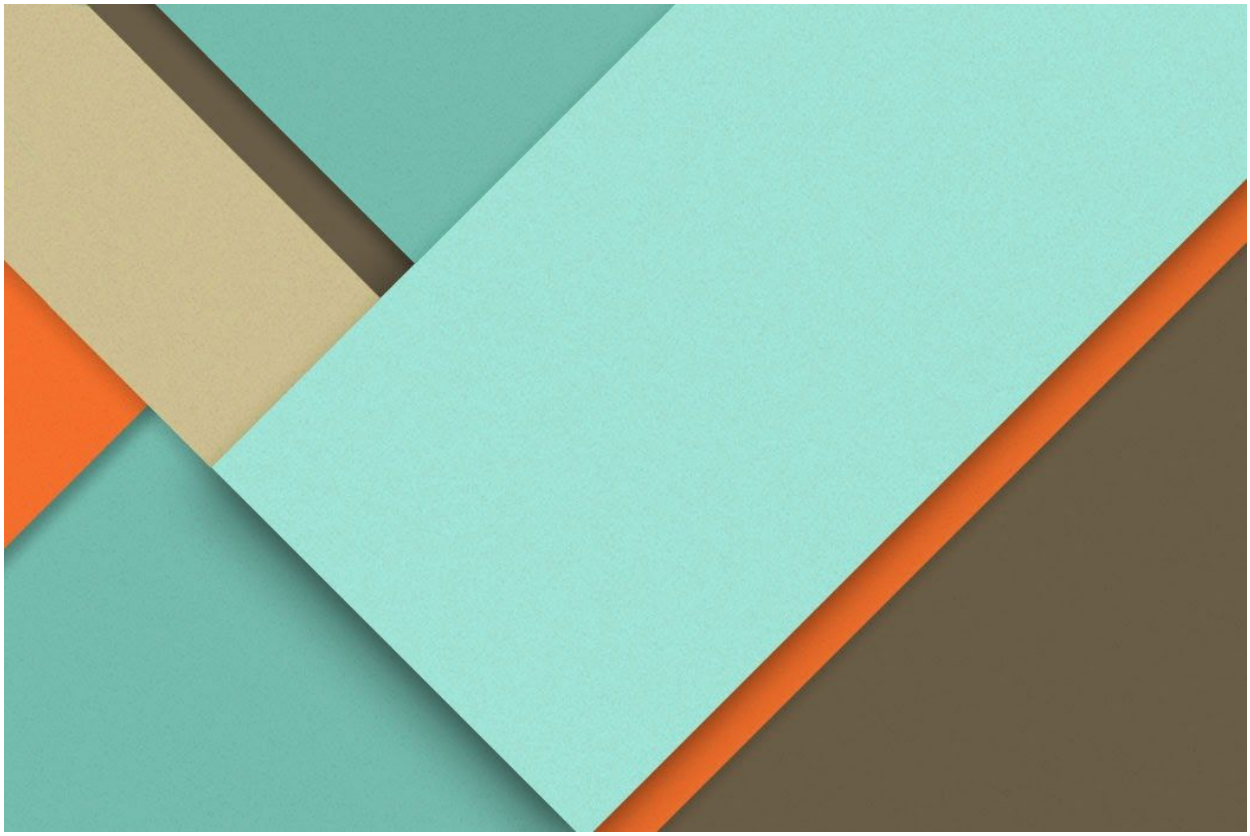


Signal Flow Graph

إسم الطالب : أفنان موسى مبروك موسى عبدالله رقم الجلوس : 15

المادة (الكود) : نظم التحكم الخطي (069yicc) الفرقة : الثانية

Overview

A signal-flow graph or signal-flowgraph (SFG), invented by Claude Shannon,but often called a Mason graph after Samuel Jefferson Mason who coined the term, is a specialized flow graph, a directed graph in which nodes represent system variables, and branches (edges, arcs, or arrows) represent functional connections between pairs of nodes. Thus, signal-flow graph theory builds on that of directed graphs (also called digraphs), which includes as well that of oriented graphs. This mathematical theory of digraphs exists, of course, quite apart from its applications.

SFGs are most commonly used to represent signal flow in a physical system and its controller(s), forming a cyber-physical system. Among their other uses are the representation of signal flow in various electronic networks and amplifiers, digital filters, state-variable filters and some other types of analog filters. In nearly all literature, a signal-flow graph is associated with a set of linear equations.

Goals

1. Generate the GUI to receive and send the data .
2. Calculate the forward paths.
3. Calculate the Loops.
4. Calculate the Non Touched Nodes.
5. Calculate the Delta.
6. calculate the overall transfer function using Mason Formula
7. Draw the single flow graph .

Source Code

https://drive.google.com/file/d/1ppKFqXJeYCXDWcWzeXesj-63xYPVVzGK/view?usp=sharing

Design :

- The user inserts the data in the frame then click submit :
  - The **GUI** class calls the class **Parser** to check that the Equations are valid and doesn't have any syntax error.
    The **Parser** returns true if valid ,false if Equation is wrong .
  - Then **Gui** calls class **OrganizationFormula** to get the linked list of **Forward Paths ,Loops** .
  - Then the GUI calls class Summation by **Loops** to get the **NonTouchingLoop**.
  - Get the Small Delta and Big Delta by using **NonTouchingLoop,Loops,Forward Paths.**
  - Then make the final formula by above data from class **FinalFormula**.

Main data structures

I. Map (Hash map)

- Map **map** to store the point which refers to the index of row and column and the string of the path from start point to this vertex .
- Map **mapValues** to store the point which refers to the index of row and column and the string of the gain from start point to this vertex .

II. Array two Dimensional

- to store the gain of all vertices in **Base[][]** in the parser from the GUI to the Class of Calculate Mason Formula.

III. LinkedList of LinkedList

- To store the Forward Paths in the class **OrganizationFormula** after calling the method **ForwardPath.**
- To store the Loop Paths in the class **OrganizationFormula** after calling the method **LoopsPath .**
- To store the index of rows which correspond to a non-null element in **Base[][]** for each column in the class **OrganizationFormula** store it in **Store.**
- To store the index of columns which correspond to a non-null element in **Base[][]** for each row in the class **OrganizationFormula** store it in **StoreLoops.**
- To store the expression of a **small Delta** as a string.
- To store the index of the loop which performs the **NonTouchNode** .

IV. Stack

- ➢ If the column in the loop function contains different gain in multi rows store the index of all rows and columns to use it in recursion form .

Algorithms description

I. Forward path

To obtain all forward paths we loop over the nodes starting from the input node and store each node that wasn't visited before. And if the path reaches the output node it's considered as a forward one.

II. Loops

The Loop algorithm is the same as the forward path algorithm but it must return to the start node again to be a loop. We start looping from each node to determine if this node belongs to some loops or not. If it belongs, so store these loops in a data structure and then remove this node from nodes to avoid repeated loops.

III. Non - touched loops

Once obtaining the loops, we need to loop over them to get all non-touched loops. So this function checks if a loop has a common node with any other loop or not. And it saves every group of non-touched loops together.

IV. Big - Delta

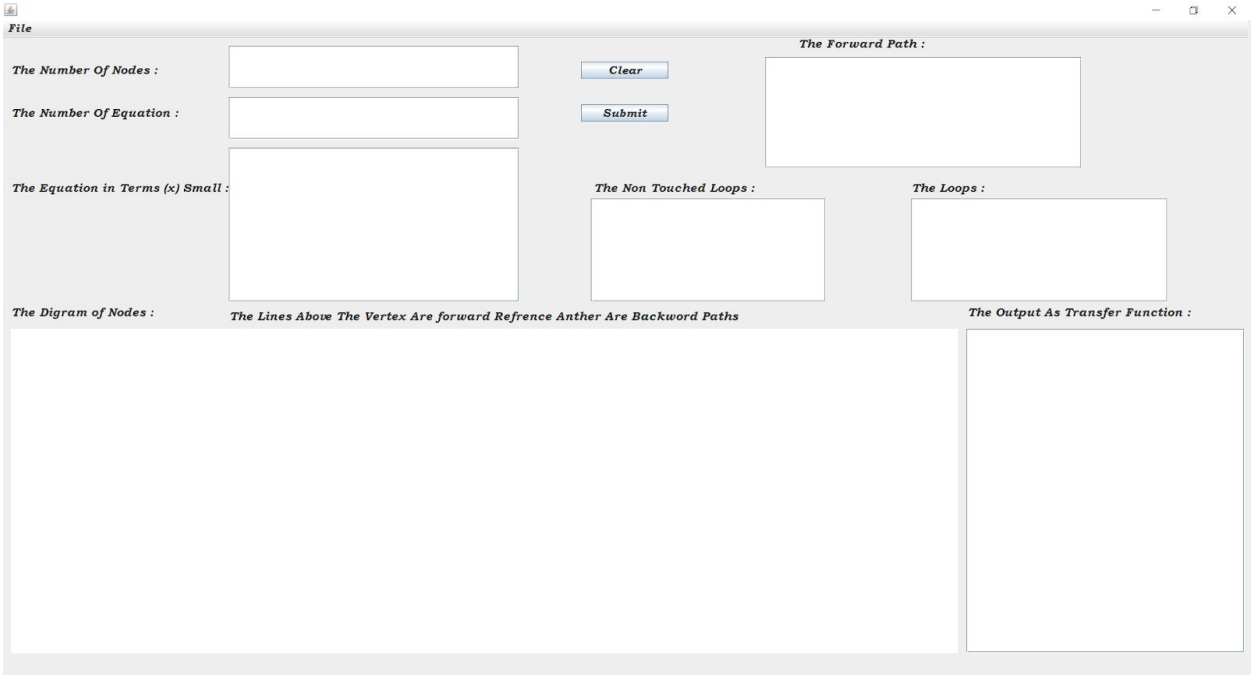
To get big-delta we need to add the gain of all individual loops together and then calculate the gain of all two-non-touched loops and then all three-non-touched until all the loops are touched.

V. Delta

To calculate delta of each path we need to get all the non-touched loops to this path and repeat the algorithm of big-delta again.

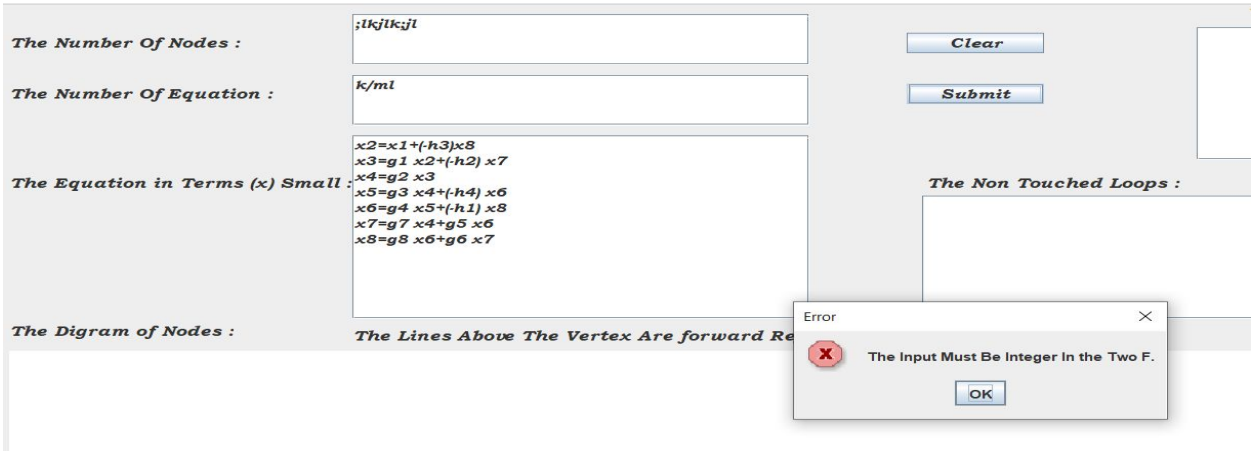
Sample Runs

- ➢ At Starting the program all text areas are not Available except the text area to insert the Number of Nodes , Number of Equations , Equations terms . the submit button won't take any action until the user enters the 3 inputs.
- ➢ If the user inputs invalid type in any text ,warning message will appear and no



solution is calculated :

- As shown if the user enters a string in the first or in the second text area the Error message will appear to him until he enters a number in this place .



➔ Example 1

➢ If the input is as shown in the photo :

The Number Of Nodes : 8

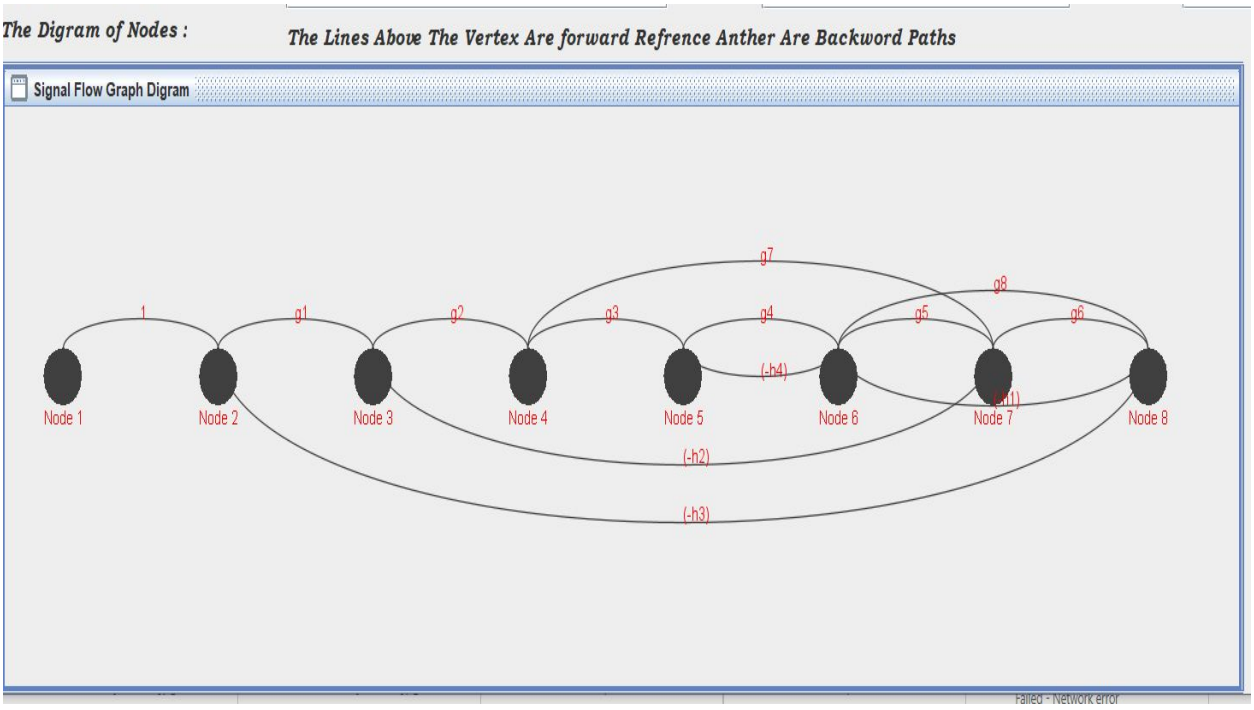
The Number Of Equation : 7

The Equation in Terms (x) Small :

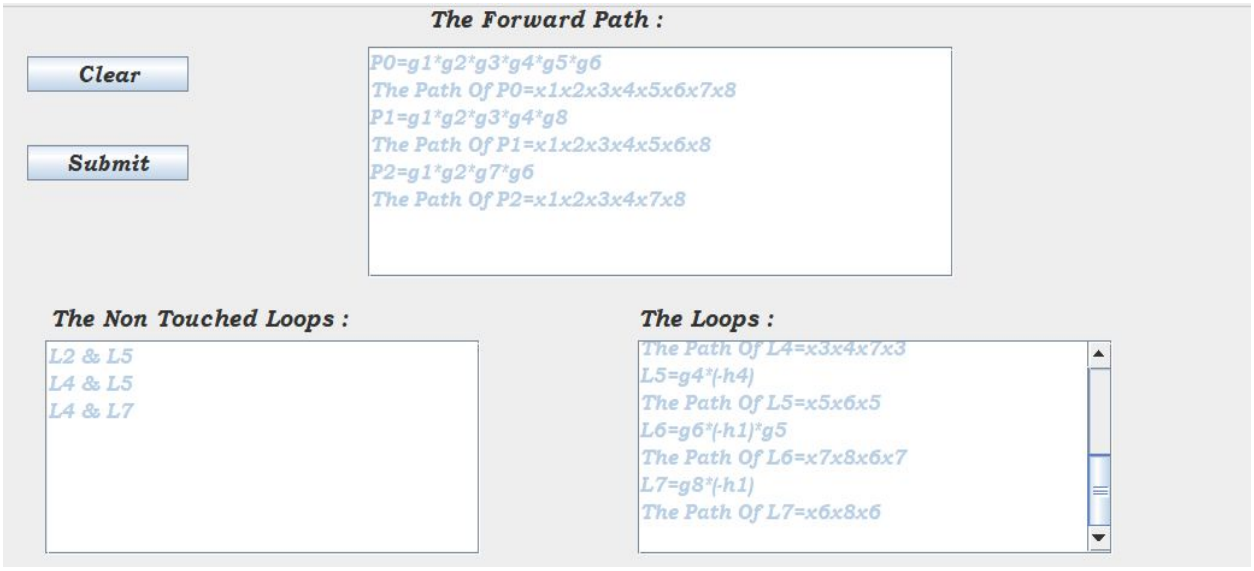
x2=x1+(-h3)x8
x3=g1 x2+(-h2) x7
x4=g2 x3
x5=g3 x4+(-h4) x6
x6=g4 x5+(-h1) x8
x7=g7 x4+g5 x6
x8=g8 x6+g6 x7

➢ The Single flow graph is :

The Digram of Nodes : The Lines Above The Vertex Are forward Refrence Anther Are Backword Paths



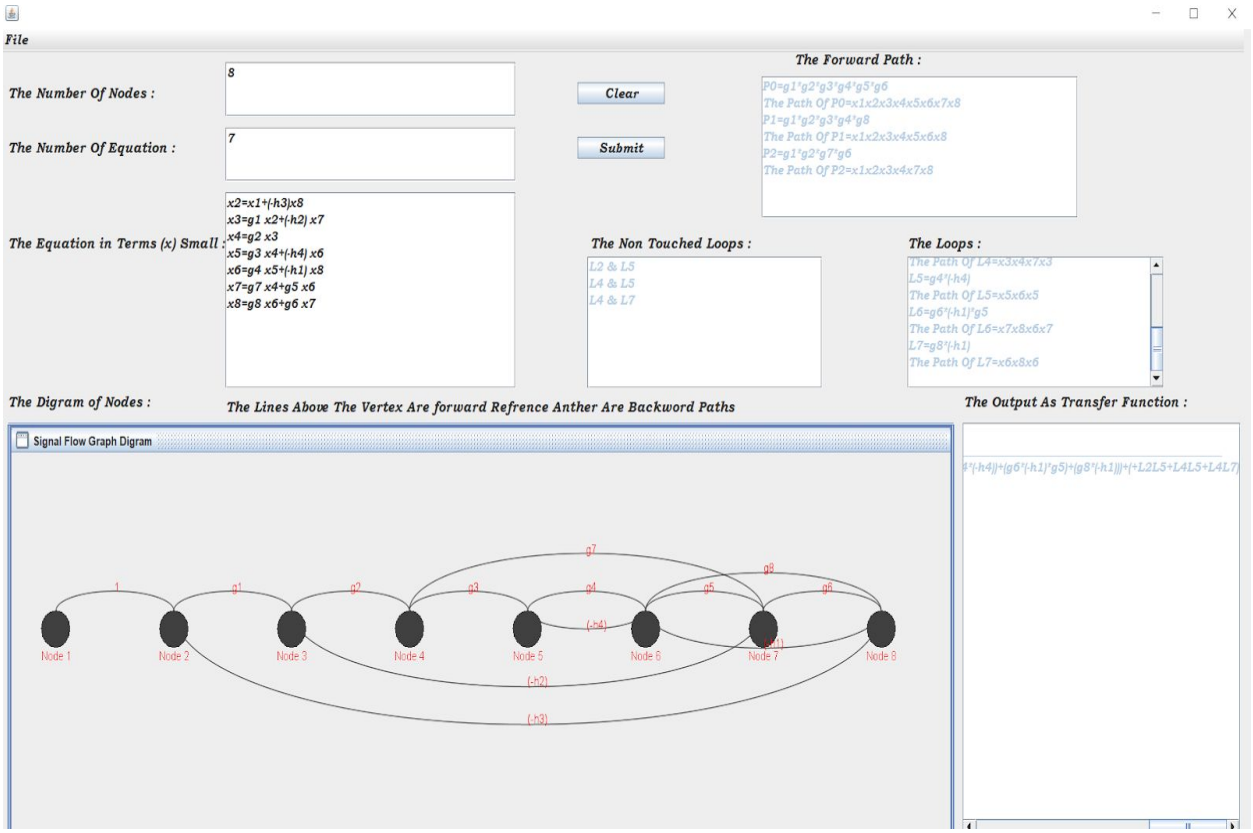
➢ **The Output Forward Path ,NOn Touching Loops ,Loops are be:**



➢ **The transfer function is :**

In this the transfer function is very tall so there is a scrollbar appear in the bottom.

➢ **The total figure for Example oneis :**

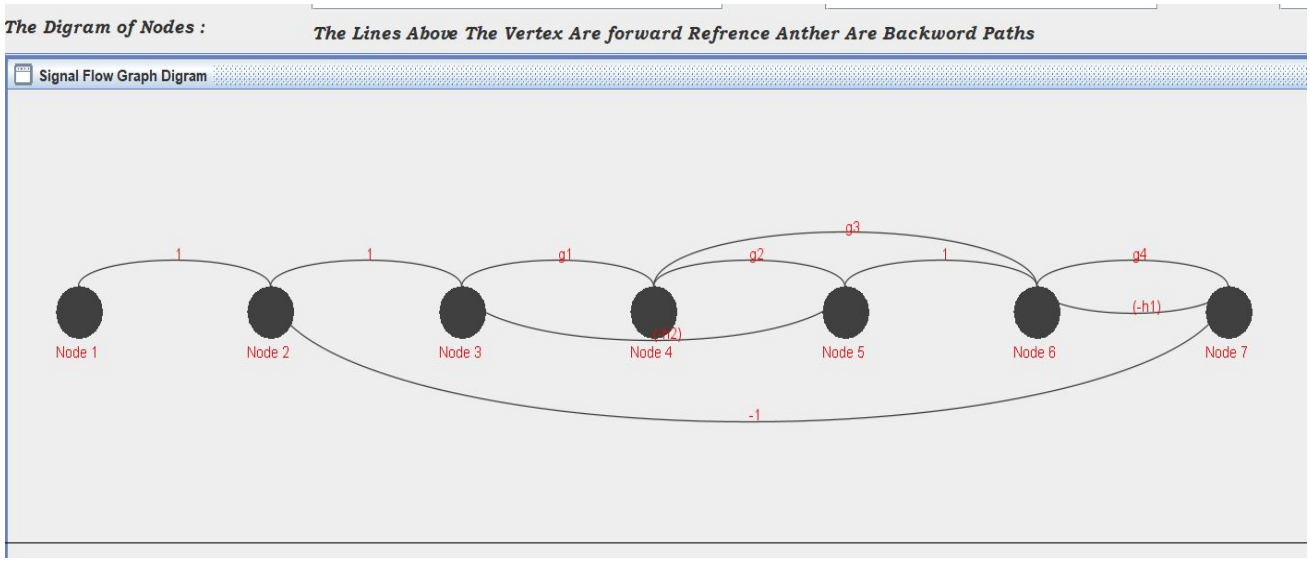


➔ Example 2

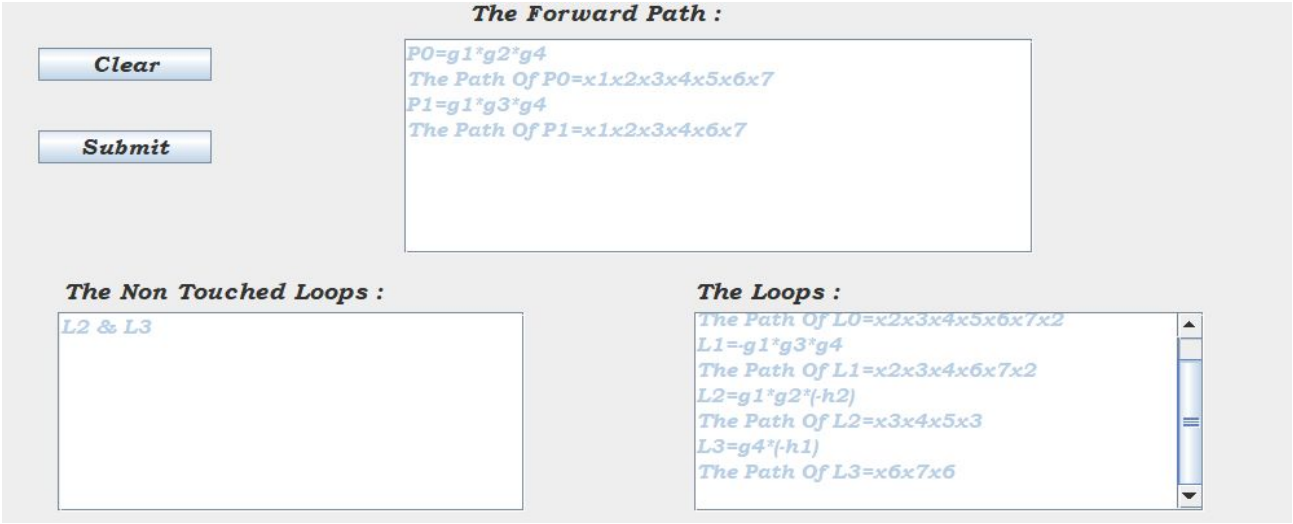
➢ **If the input is as shown in the photo :**

The Number Of Nodes :	7
The Number Of Equation :	6
The Equation in Terms (x) Small :	x2=x1+-x7 x3=x2+(-h2)x5 x4=g1x3 x5=g2x4 x6=x5+g3x4+(-h1)x7 x7=g4x6

➢ **The Single flow graph is :**



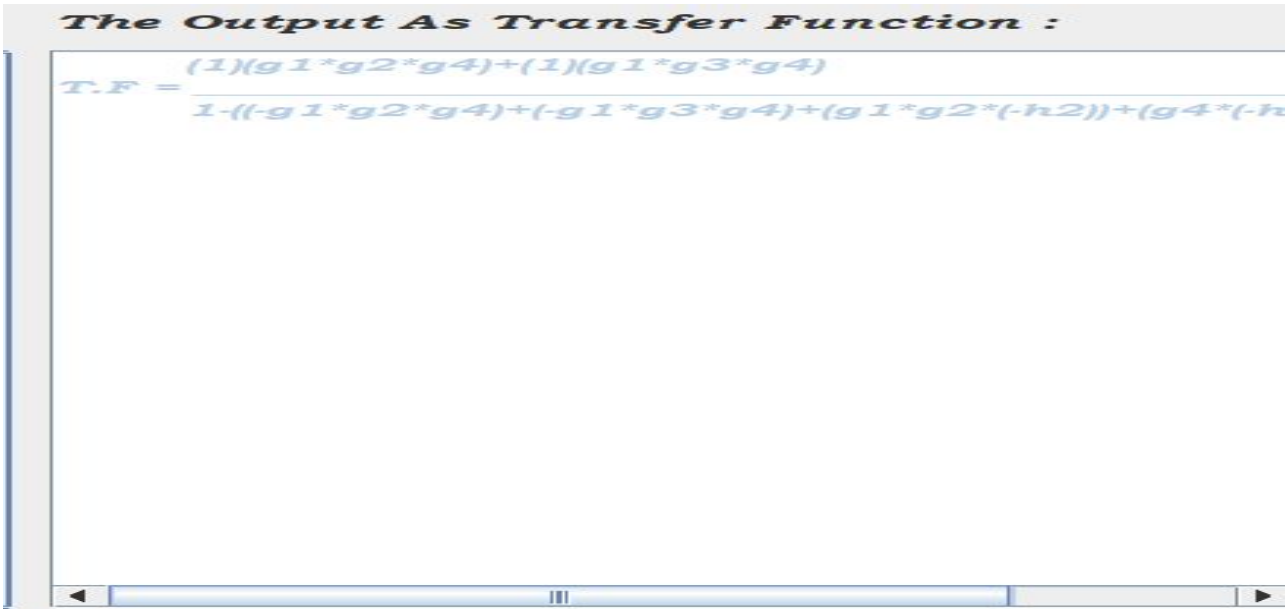
➢ **The Output Forward Path ,NOn Touching Loops ,Loops are be:**

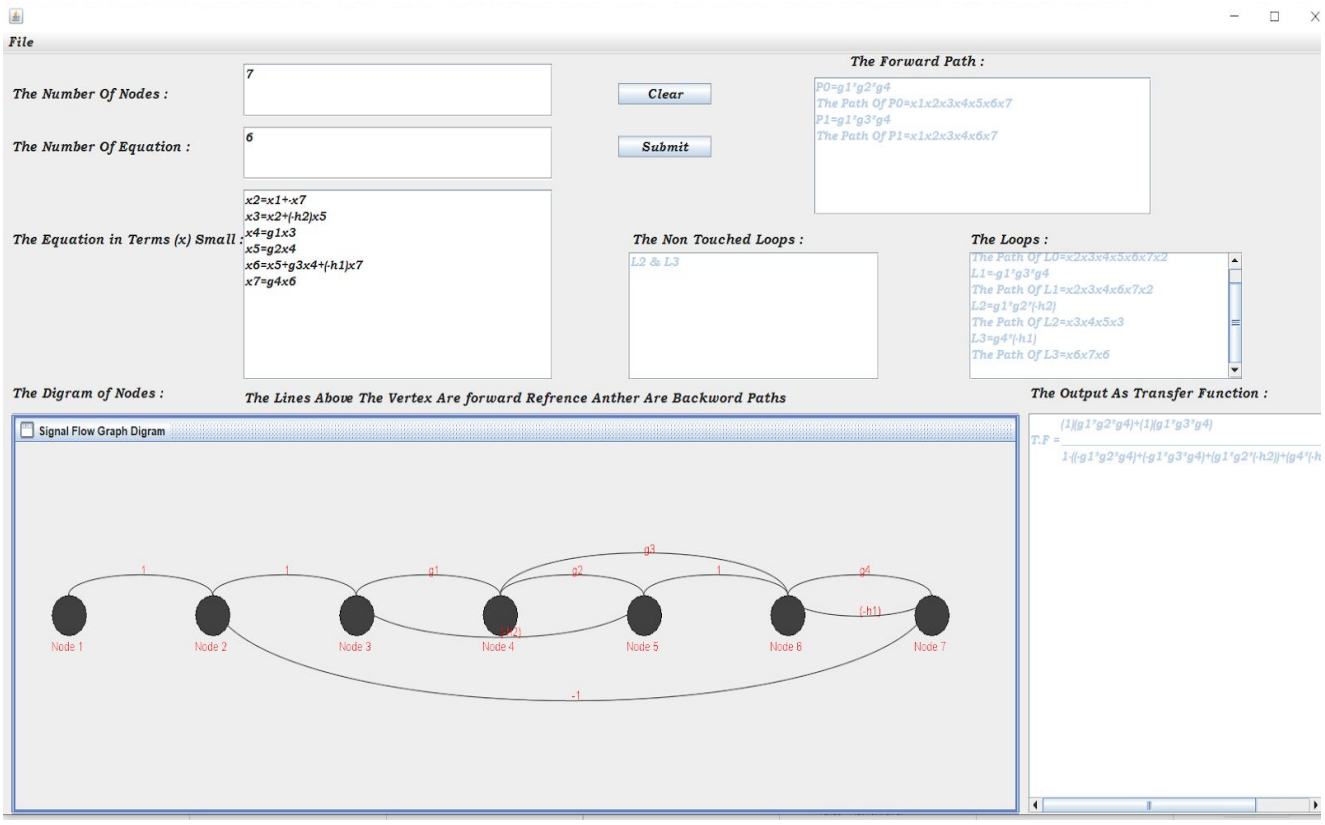


➢ **The transfer function is :**

$$\frac{(1)(g1*g2*g4)+(1)(g1*g3*g4)}{1-((-g1*g2*g4)+(-g1*g3*g4)+(g1*g2*(-h2))+(g4*(-h1)))+(+L2L3)}$$

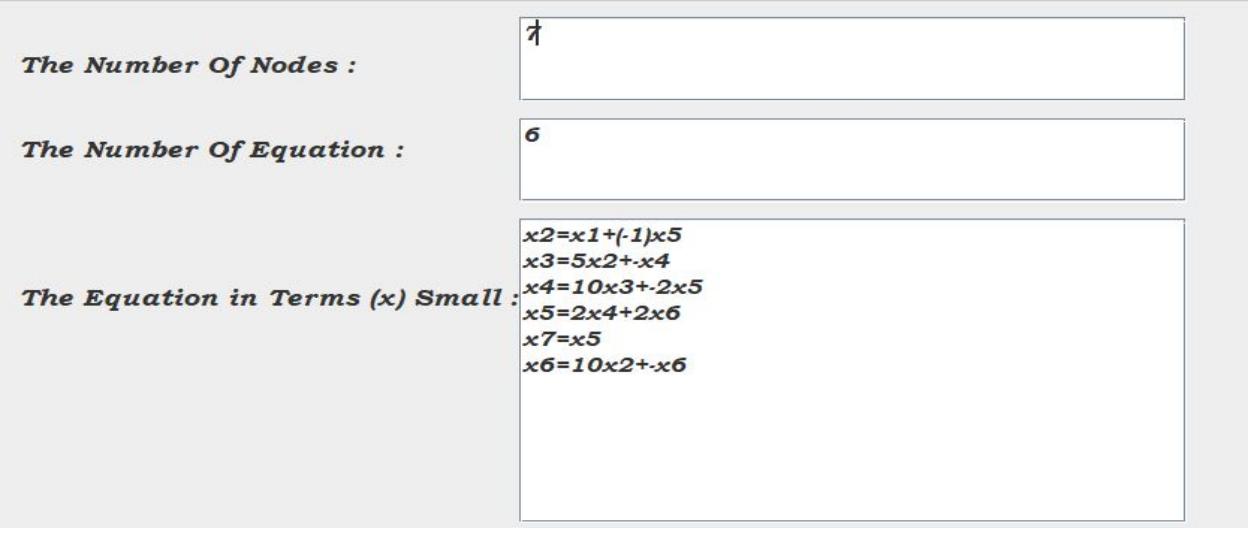
➢ **The total figure for Example two is :**


The Output As Transfer Function :
T.F =
(1)(g1*g2*g4)+(1)(g1*g3*g4)
1-((-g1*g2*g4)+(-g1*g3*g4)+(g1*g2*(-h2))+(g4*(-h

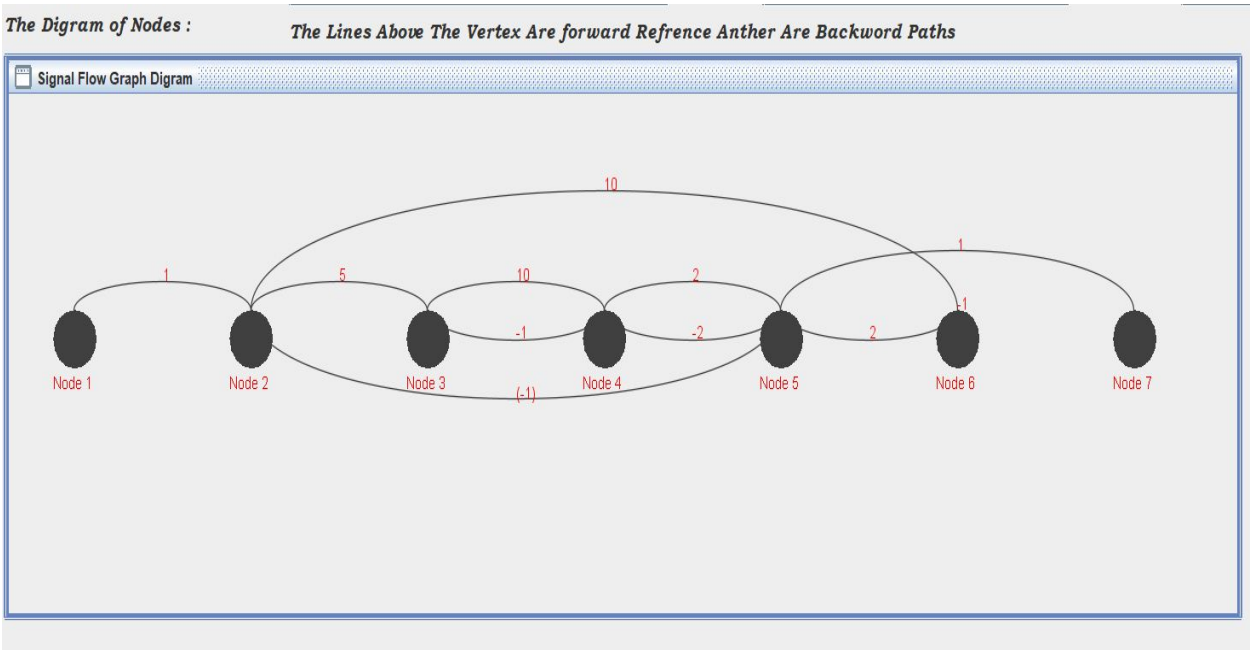

File
The Number Of Nodes :
7
The Number Of Equation :
6
Clear
Submit
The Forward Path :
P0=g1*g2*g4
The Path Of P0=x1x2x3x4x5x6x7
P1=g1*g3*g4
The Path Of P1=x1x2x3x4x6x7
The Equation in Terms (x) Small :
x2=x1+x7
x3=x2+(-h2)x5
x4=g1x3
x5=g2x4
x6=x5+g3x4+(-h1)x7
x7=g4x6
The Non Touched Loops :
L2 & L3
The Loops :
The Path Of L0=x2x3x4x5x6x7x2
L1=-g1*g3*g4
The Path Of L1=x2x3x4x6x7x2
L2=g1*g2*(-h2)
The Path Of L2=x3x4x5x3
L3=g4*(-h1)
The Path Of L3=x6x7x6
The Digram of Nodes :
The Lines Above The Vertex Are forward Refrence Anther Are Backword Paths
The Output As Transfer Function :
Signal Flow Graph Digram
Node 1
Node 2
Node 3
Node 4
Node 5
Node 6
Node 7
1
1
g1
g3
g2
1
g4
(-h1)
-1

➔ Example 3

➢ **If the input is as shown in the photo :**



➢ **The Single flow graph is :**



➢ **The Output Forward Path ,NOn Touching Loops ,Loops are be:**

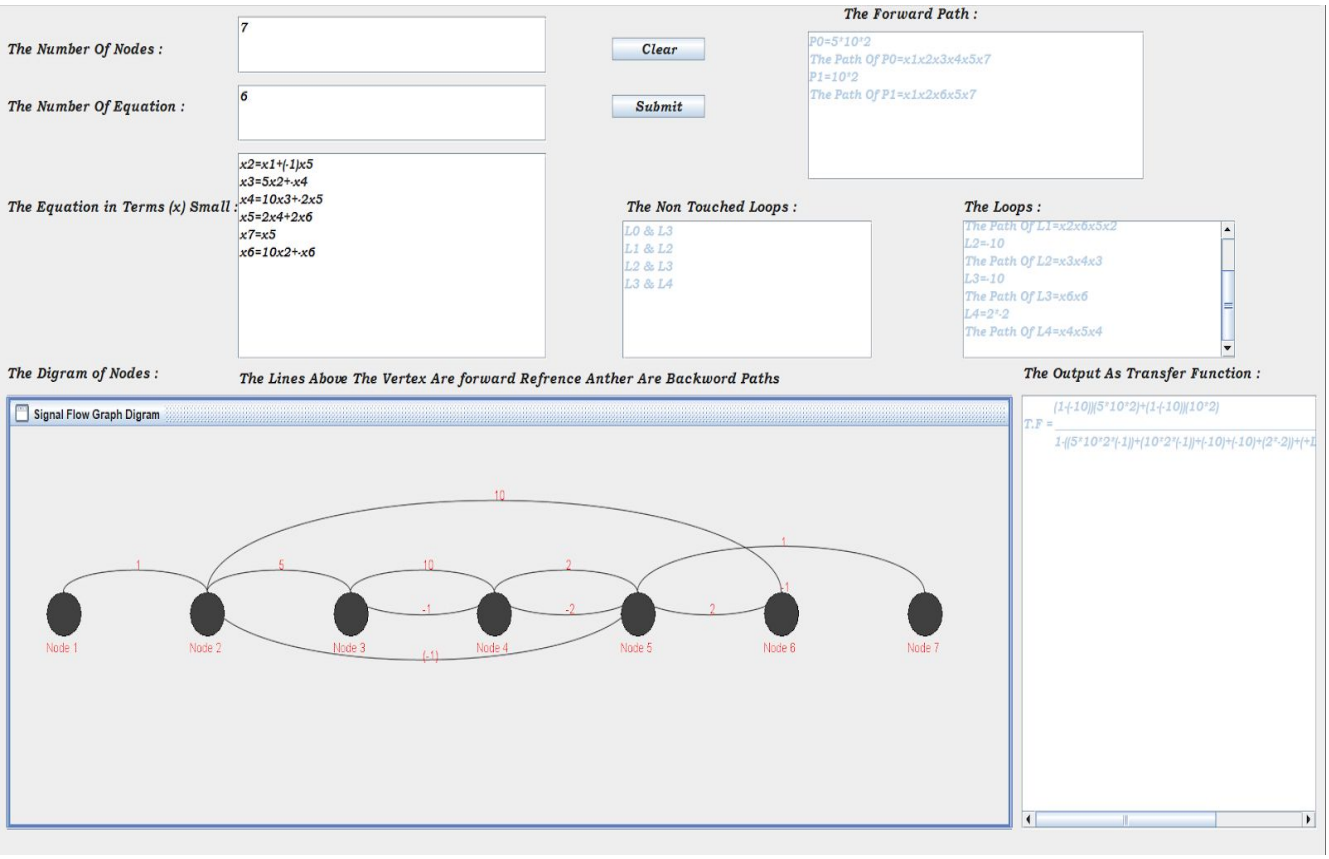
➢ **The transfer function is :**

$$\frac{(1-(-10))(5*10*2)+(1-(-10))(10*2)}{1-((5*10*2*(-1))+(10*2*(-1))+(-10)+(-10)+(2*-2))+(+L0L3+L1L2+L2L3+L3L4)}$$

The Output As Transfer Function :

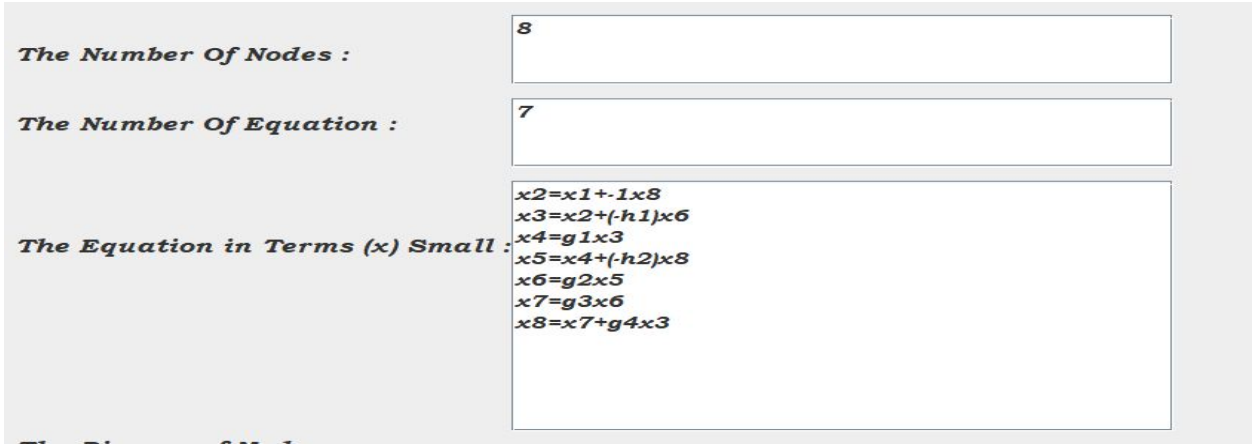
$$T.F = \frac{(1-(-10))(5*10*2)+(1-(-10))(10*2)}{1-((5*10*2*(-1))+(10*2*(-1))+(-10)+(-10)+(2*-2))+(+L}$$

➢ **The total figure for Example three is :**

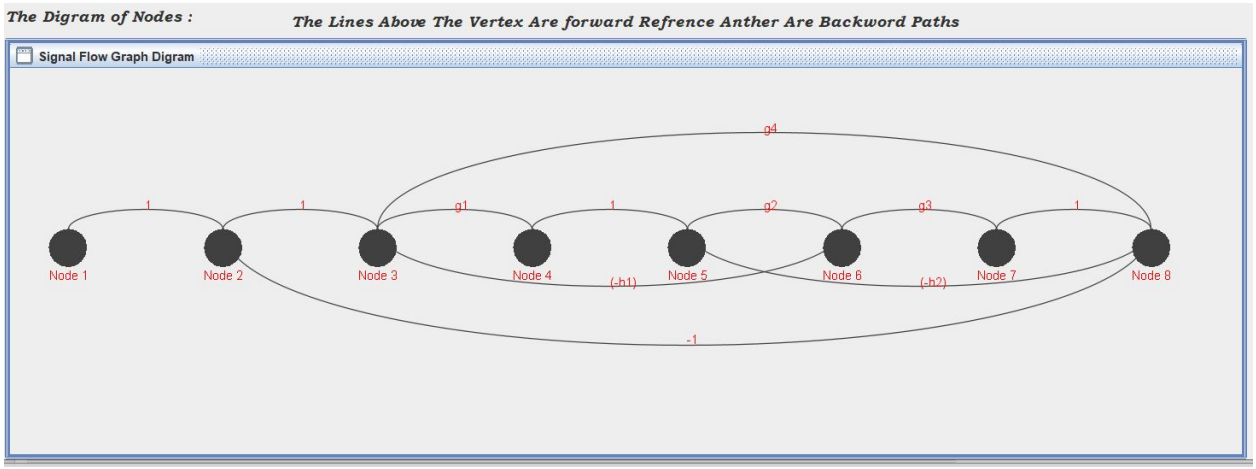


➔ Example 4

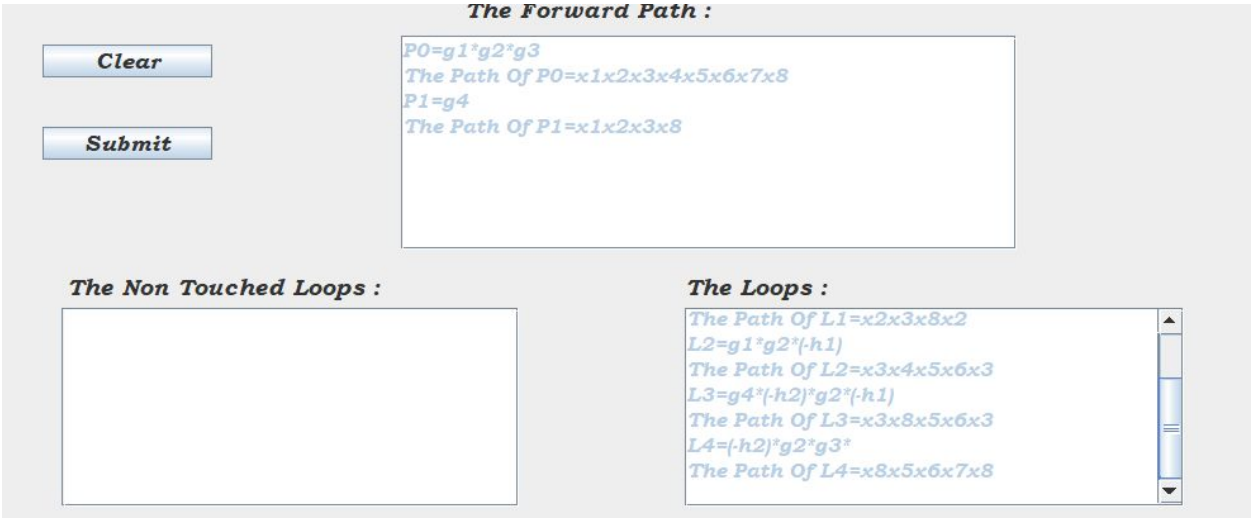
➢ **If the input is as shown in the photo :**



➢ **The Single flow graph is :**



➢ **The Output Forward Path ,Non Touching Loops ,Loops are be:**



➢ **The transfer function is :**

```
(1)(g1*g2*g3)+(1)(g4)
_____________________
1-((-g1*g2*g3)+(-g4)+(g1*g2*(-h1))+(g4*(-h2)*g2*(-h1))+((-h2)*g2*g3*))
```

The Output As Transfer Function :

$$T.F = \frac{(1)(g1*g2*g3)+(1)(g4)}{1-((-g1*g2*g3)+(-g4)+(g1*g2*(-h1))+(g4*(-h2}$$

➢ **The total figure for Example four is :**

The Number Of Nodes : 8

The Number Of Equation : 7

Clear

Submit

The Forward Path :

P0=g1'g2'g3
The Path Of P0=x1x2x3x4x5x6x7x8
P1=g4
The Path Of P1=x1x2x3x8

The Equation in Terms (x) Small :

x2=x1+-1x8
x3=x2+(-h1)x6
x4=g1x3
x5=x4+(-h2)x8
x6=g2x5
x7=g3x6
x8=x7+g4x3

The Non Touched Loops :

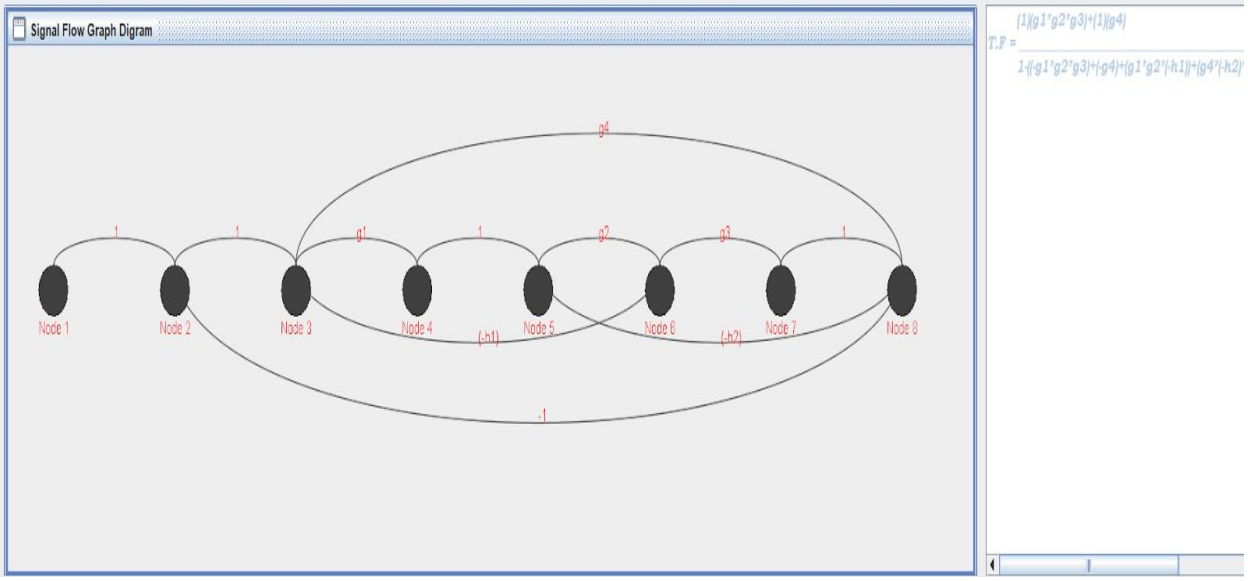
The Loops :

The Path Of L1=x2x3x8x2
L2=g1'g2'(-h1)
The Path Of L2=x3x4x5x6x3
L3=g4'(-h2)'g2'(-h1)
The Path Of L3=x3x8x5x6x3
L4=(-h2)'g2'g3'
The Path Of L4=x8x5x6x7x8

The Digram of Nodes :

The Lines Above The Vertex Are forward Refrence Anther Are Backword Paths

The Output As Transfer Function :



T.F = (1)(g1'g2'g3)+(1)(g4) / 1-(g1'g2'g3)+(-g4)+(g1'g2'(-h1))+(g4'(-h2)