DC Circuit Simulator:

**Testing**

Juraj Pavlović

2020./2021.

Content

[Testing 2](#_Toc67692227)

[Objective 1 2](#_Toc67692228)

[Objective 2 16](#_Toc67692229)

[Objective 3 19](#_Toc67692230)

[Objective 4 28](#_Toc67692231)

[Objective 5 38](#_Toc67692232)

[Objective 6 41](#_Toc67692233)

# Testing

In order to properly test the program a series of tests will have to be planned and completed so that when the program reaches the end user there are no unexpected errors or outputs within the program. Tests will be based on the objectives from Analysis section in order to see if all objectives have been achieved.

## Objective 1

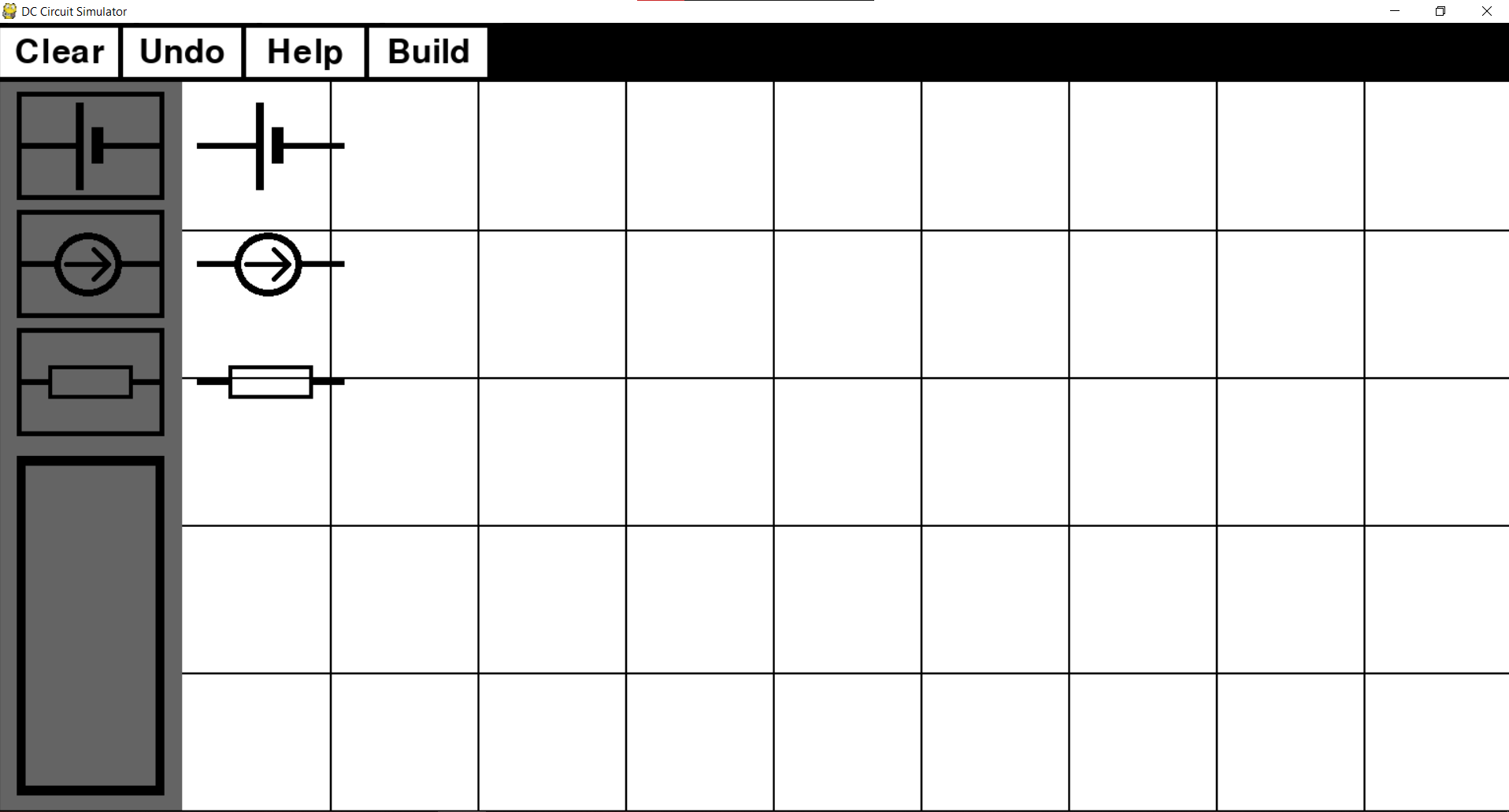
Objective 1 focuses on allowing the user to construct a circuit. Below is my objective 1.

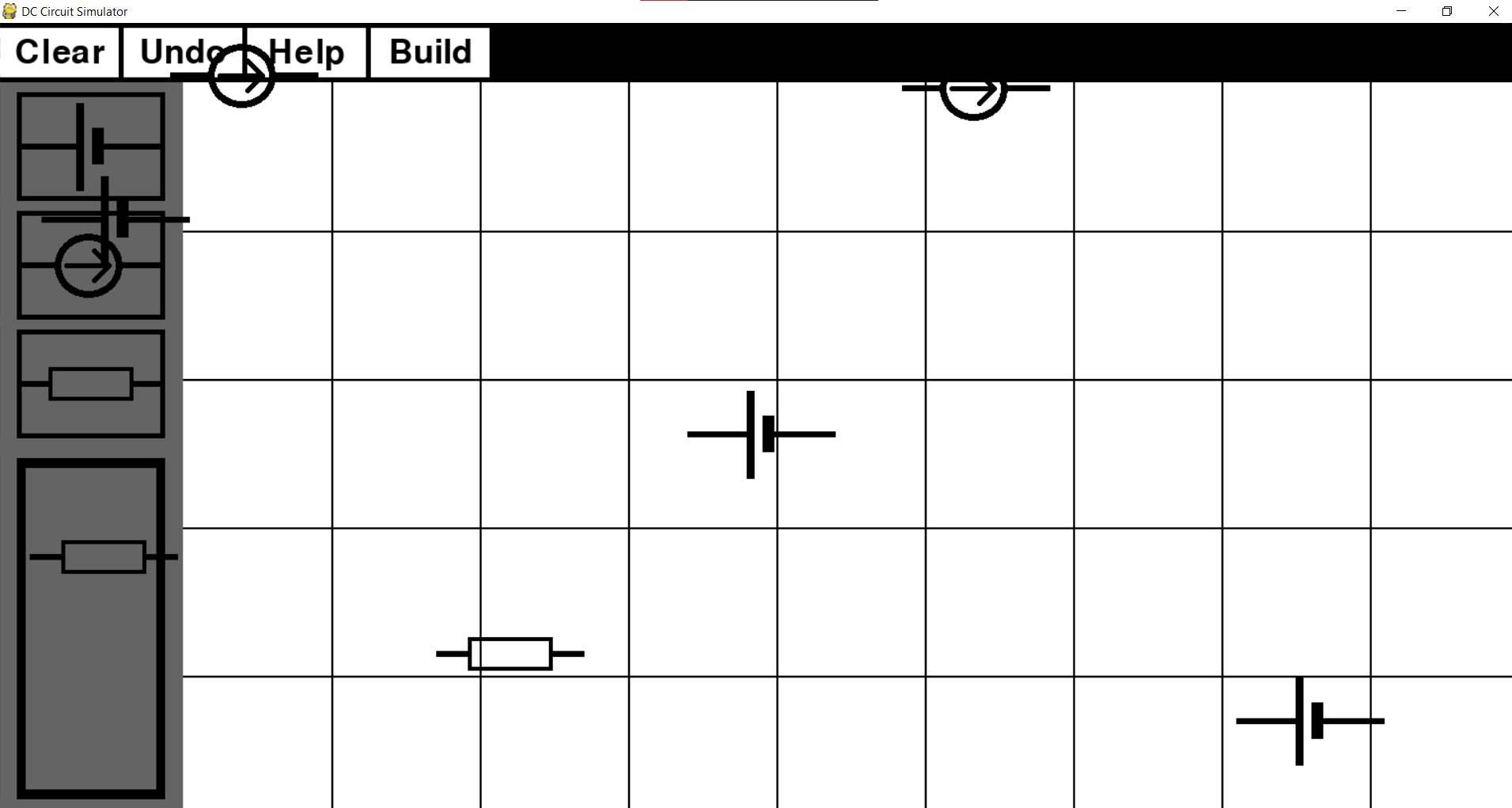
1. *Make a GUI (Graphical User Interface) which will give the user all necessary options to construct a DC circuit.*
   1. *There needs to be one button for each component which will allow the user to add the component to the main tab. By clicking on the button, a chosen component should appear in the main tab.*
   2. *A drag and drop system needs to be created for all the components which are currently in the main tab. Once a component is added to the main tab, a user should be able to change its x and y coordinates by clicking on the component and dragging it across the screen.*
   3. *There needs to be a way for the user to create a network of components in the main tab. User will be able to create a connection between two components by clicking on them with mouse’s right click button. By repeating this process, a user will be able to create a DC circuit network*
   4. *When a component in the main tab is selected by the user, a suitable feedback should be given to the user showing which component they have selected. This will be done by changing the components appearance so that it becomes unique in respect to other components of the same type in the main tab.*
   5. *For a selected component, the user should be able to see its name, value, unit prefix, unit and a short description of components use in the circuit. This will be displayed in the specific part of the screen designed for this use.*
   6. *Each type of component should have their default value and unit prefix, but once a component is selected and its characteristics are displayed, the user should be able to alter the component’s value and unit prefix by typing their own components attributes.*
   7. *No value a user inputs as a component value and a unit prefix should cause a program to crash. A validation system will be used to check if correct values have been entered. If values the user entered are incorrect or the unit prefixes do not exist, the user will be shown an appropriate message explaining that only numbers can be used as the component values or the user will be shown a list of possible unit prefixes.*

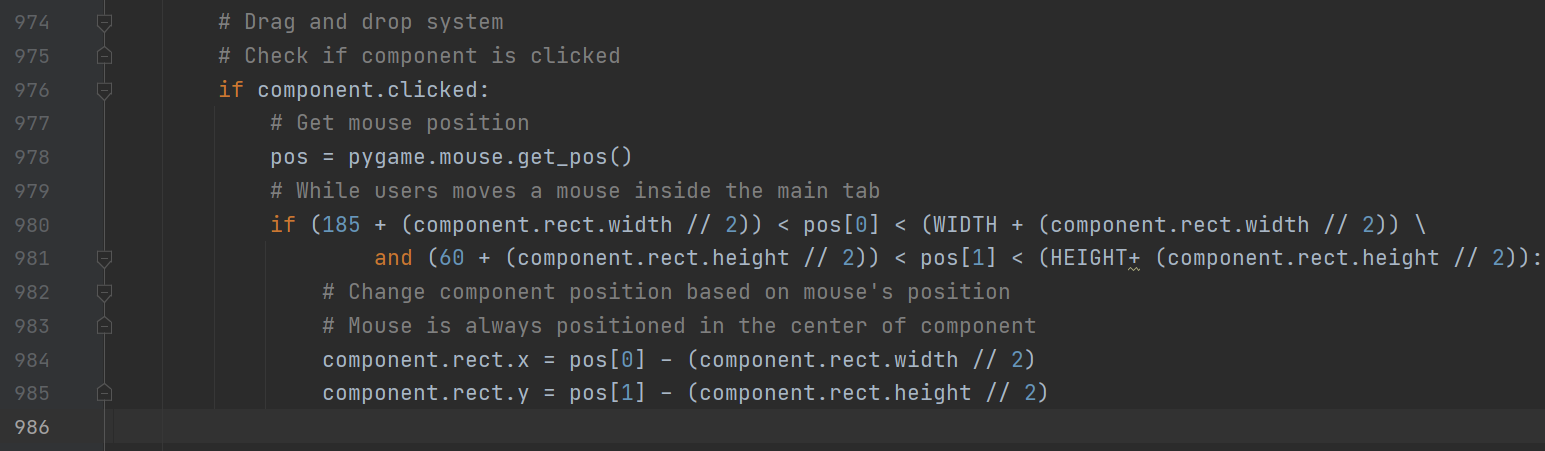
Below is a test table with description of each test, users input and outcomes. A short comment on the test is also included. Screenshot evidence of the tests are included below the table. Each screenshot is referenced by a test number.

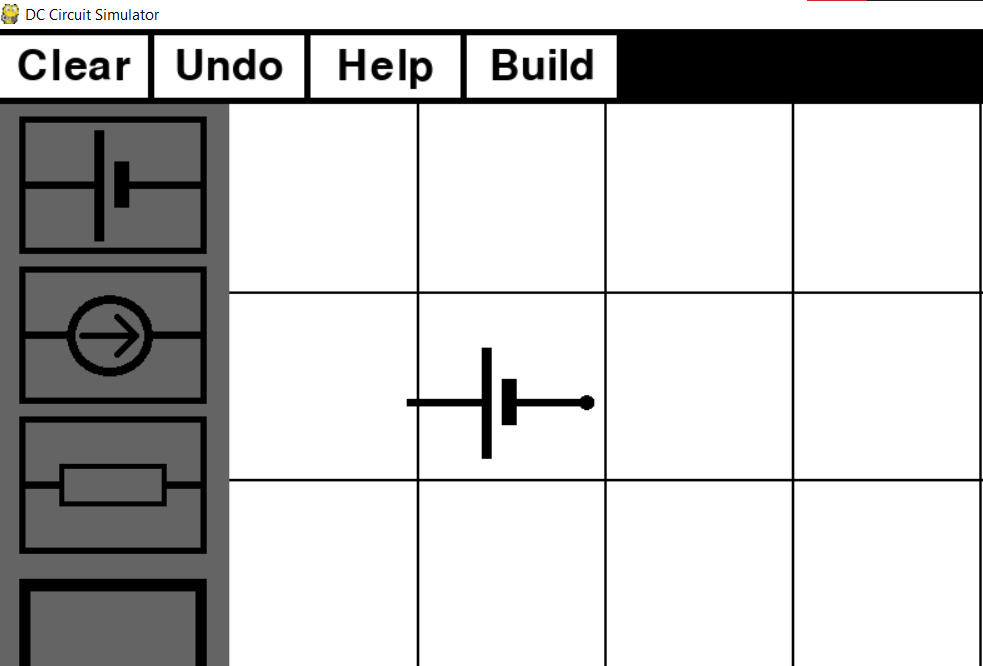
If the test failed, suitable improvements are made to the code. Screenshot of code improvements are included below test’s evidence.

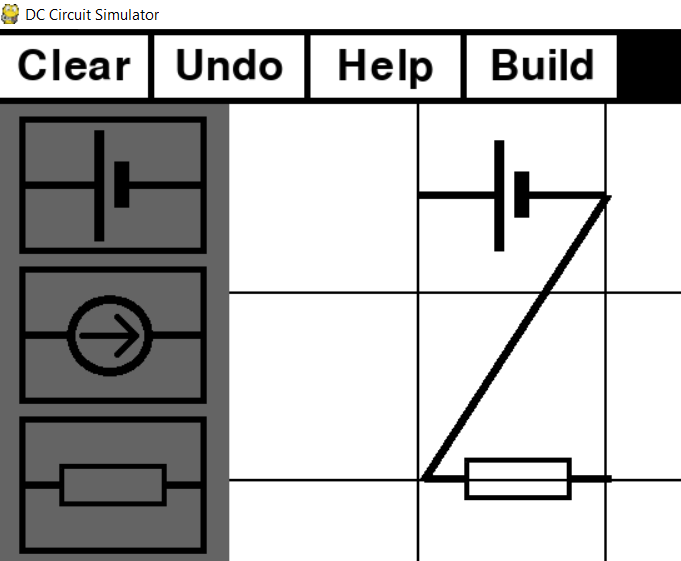
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Objective number being tested** | **Test description** | **User input** | **Expected outcome** | **Actual outcome** | **Comment on the outcome**  **(Pass/Fail)** |
| 1 | 1.1. | Button with independent voltage source image is pressed | Left mouse click on the independent voltage source image | Independent voltage source gets added to the main tab | Independent voltage source appeared next to the independent voltage source image | Correct component was added to the main tab.  **Pass** |
| 2 | 1.1 | Button with independent current source image is pressed | Left mouse click on the independent current source image | Independent current source gets added to the main tab | Independent current source appeared next to the independent current source image | Correct component was added to the main tab.  **Pass** |
| 3 | 1.1 | Button with resistor image is pressed | Left mouse click on the resistor image | Resistor gets added to the main tab | Resistor appeared next to the resistor image | Correct component was added to the main tab.  **Pass** |
| 4 | 1.2 | Component is dragged to the other side of the main tab and then dropped. | User clicks on the component with left mouse click. While holding the mouse button, mouse cursor position is changed. Mouse button is released. | Component’s position is changed according to the change of the mouse cursor position. | Component is no longer at its previous position. Its position was changed according to the change of the mouse cursor position. | User was able to drag and drop a component (change its position).  **Pass** |
| 5 | 1.2 | Component is dragged to the toolbar and then dropped. Component should not move outside the component tab. | User clicks on the component with left mouse click. While holding the mouse button, mouse cursor position is changed. Mouse button is released. | Component’s position is changed according to the change of the mouse cursor position until it reaches the edge of component tab. | Component is no longer at its previous position. Its position was changed according to the change of the mouse cursor position. | User was able to drag and drop a component (change its position), even though component’s position should stop changing once it reached the edge of the component tab.  **Fail** |
| 6 | 1.2 | Component is dragged to the component tab and then dropped. Component should not move outside the component tab. | User clicks on the component with left mouse click. While holding the mouse button, mouse cursor position is changed. Mouse button is released. | Component’s position is changed according to the change of the mouse cursor position until it reaches the edge of component tab. | Component is no longer at its previous position. Its position was changed according to the change of the mouse cursor position. | User was able to drag and drop a component (change its position), even though component’s position should stop changing once it reached the edge of the component tab.  **Fail** |
| 7 | 1.3 | To connect two components together, the user first needs to mark first component as ‘ready to connect’ | Right mouse click on component’s right side | Component gets marked as ‘ready to connect’ indicated by black circle on right side of the component | Component’s image changed to image with black circle on its right side. | Right side of component got marked as ‘ready to connect’.  **Pass** |
| 8 | 1.3 | User needs to be able to create a network between two components. | Right mouse click on another component’s left side. | A black line should be drawn, joining first component’s right side and second component’s left side. | A black line joining right and left side of two components appeared. | A network between two components was created.  **Pass** |
| 9 | 1.4 | Selected component needs to change its appearance to show the user that it was selected. | Left mouse click on a component. | Component image needs to change. | A black rectangle appeared, framing component’s image. | Component’s image changed in such way that user can differentiate selected and non-selected components.  **Pass** |
| 10 | 1.5 | User should be able to see characteristics of a selected component (name, vale, short description). | Left click on a component. | Component’s characteristics get displayed to the user. | Component’s characteristics are displayed in the component tab. | User can check component’s characteristic by selecting a component.  **Pass** |
| 11 | 1.6 | User should be able to change component’s value | Left mouse click on selected component’s value. User enters new value. | New value gets displayed and stored. | New value got displayed and stored. | User was able to change component’s value.  **Pass** |
| 12 | 1.6 | User should be able to change component’s unit prefix | Left mouse click on selected component’s unit prefix. User enters new unit prefix. | New unit prefix gets displayed and stored. | New unit prefix got displayed and stored. | User was able to change component’s unit prefix.  **Pass** |
| 13 | 1.7 | Presence check for value. | Value is left empty. | Appropriate message occurs. | ‘No value entered. Please enter component’s value.’ is displayed | Component’s value cannot be empty.  **Pass** |
| 14 | 1.7 | Range check for value | Only ‘0’ is inputted as component’s value. | Appropriate message occurs. | ‘Component’s value cannot be zero. Please enter a valid value.’ is displayed | Component’s value cannot be ‘0’.  **Pass** |
| 15 | 1.7 | Cross-reference check for value | Letter ‘T’ is entered as component’s value | Appropriate message occurs. | ‘Only numbers can be entered as component’s value.’ is displayed | Component’s value cannot contain letters.  **Pass** |
| 16 | 1.7 | Cross-reference check for unit prefix | A number ‘9’ is entered as a unit prefix | Appropriate message occurs. | ‘Unknown unit prefix entered. Please enter a valid unit prefix. Valid unit prefixes: 'f', 'p', 'n', 'u', 'm', '', 'k', 'M', 'G', 'T'.’ is displayed. | Component’s value needs to be a valid unit prefix. User was shown a list of valid unit prefixes.  **Pass** |
| 17 | 1.7 | Cross-reference check for unit prefix | A letter ‘b’ is entered as a unit prefix | Appropriate message occurs. | ‘Unknown unit prefix entered. Please enter a valid unit prefix. Valid unit prefixes: 'f', 'p', 'n', 'u', 'm', '', 'k', 'M', 'G', 'T'.’ is displayed. | Component’s unit prefix needs to be a valid unit prefix. User was shown a list of valid unit prefixes.  **Pass** |
| 18 | 1.7 | Length check for unit prefix | Two letters are entered as a unit prefix | Appropriate message occurs. | ‘Unit prefix be only one character.’ is displayed. | Component’s unit prefix can only consist of one valid letter.  **Pass** |

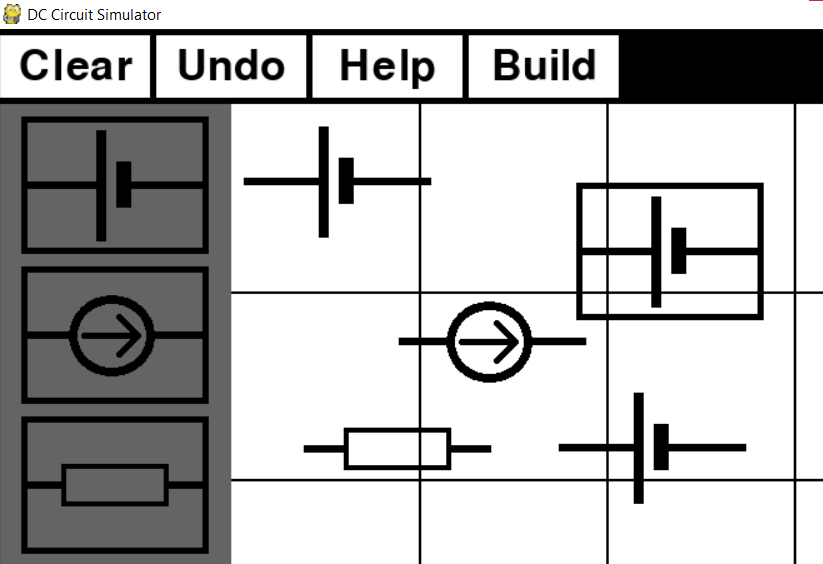
Screenshot evidence for tests 1, 2, 3:

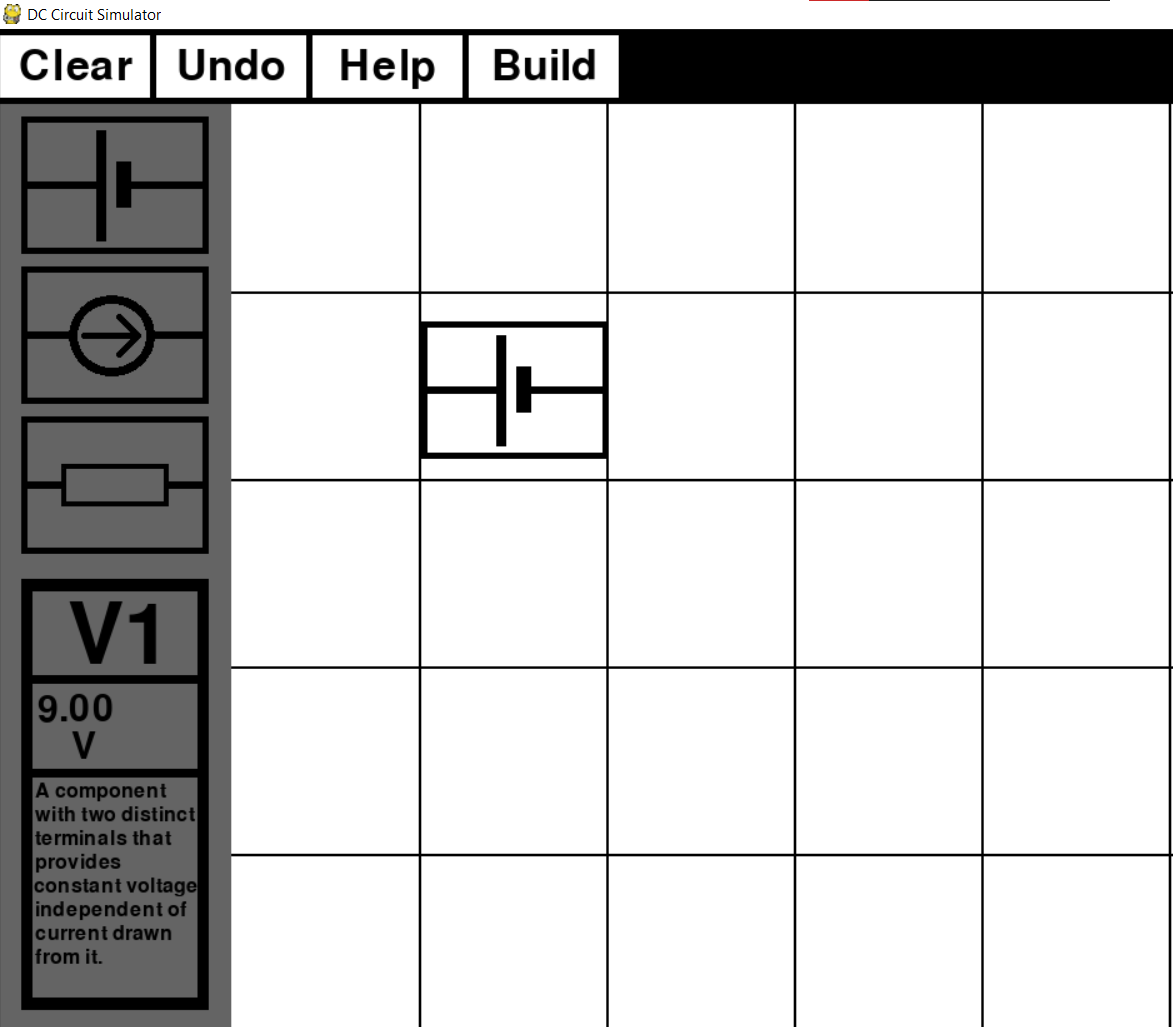
Screenshot evidence for tests 4, 5, 6: 

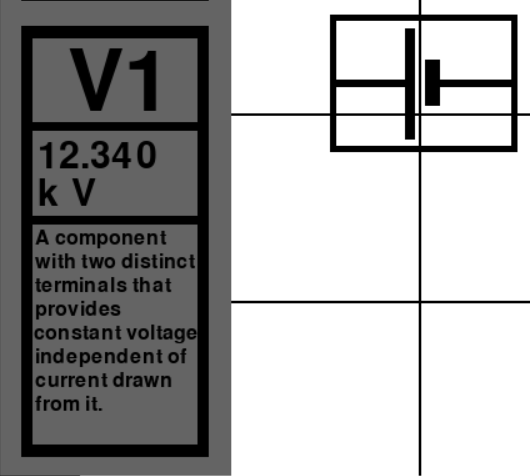
Component should not be allowed to move outside the component tab. Below is a code which fixes this issuse. 

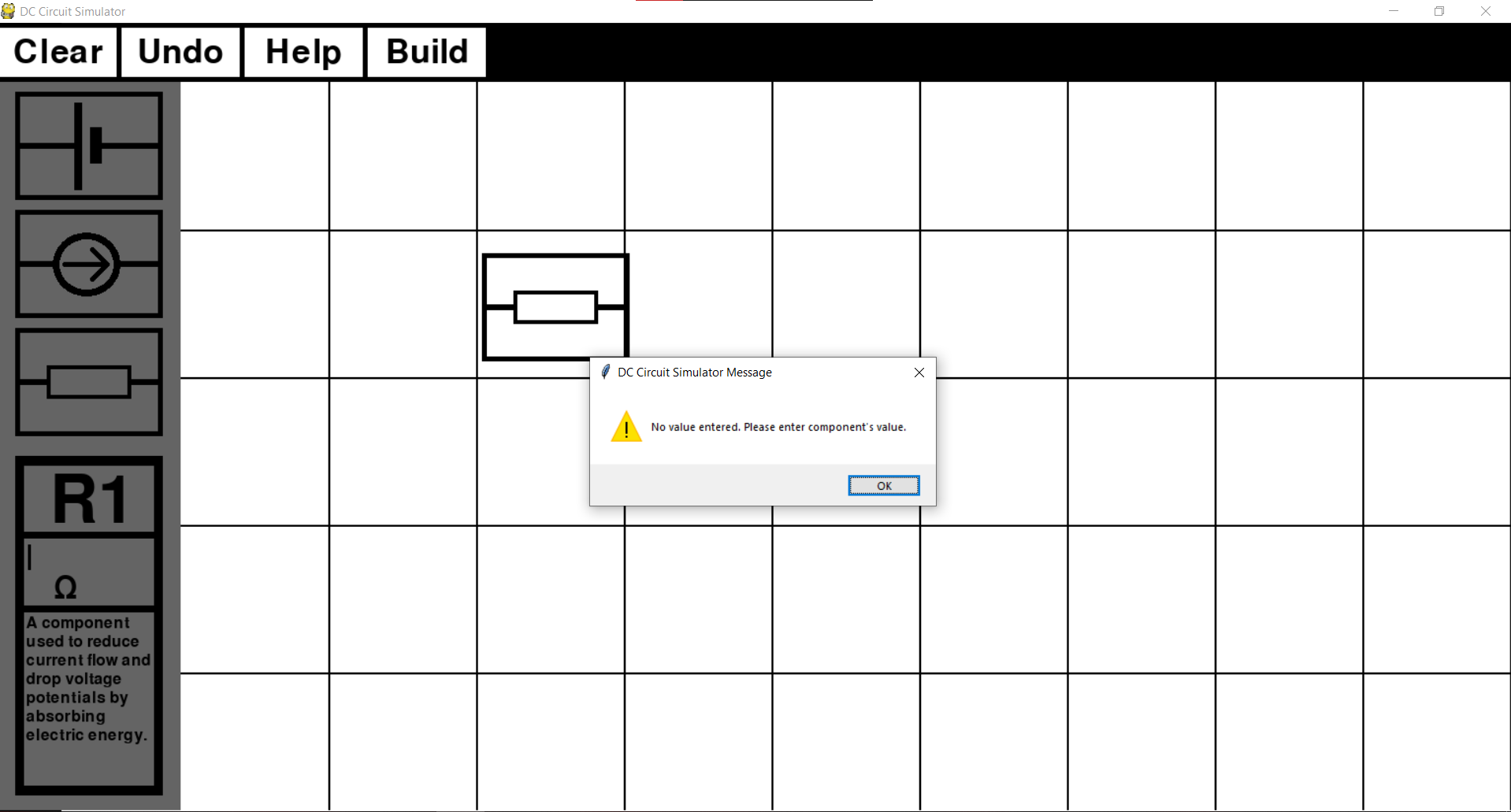
Screenshot evidence for test 7: 

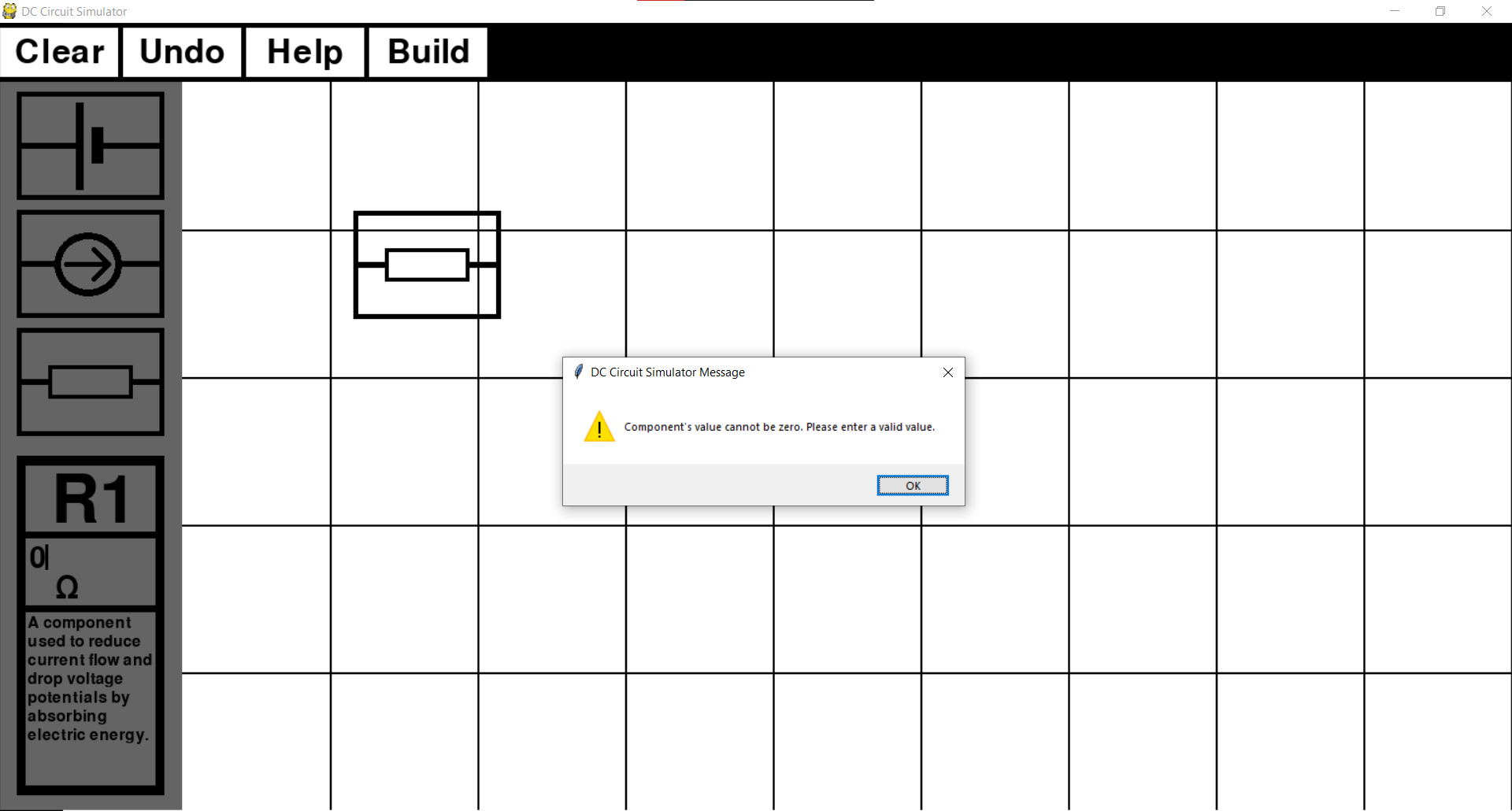
Screenshot evidence for test 8: 

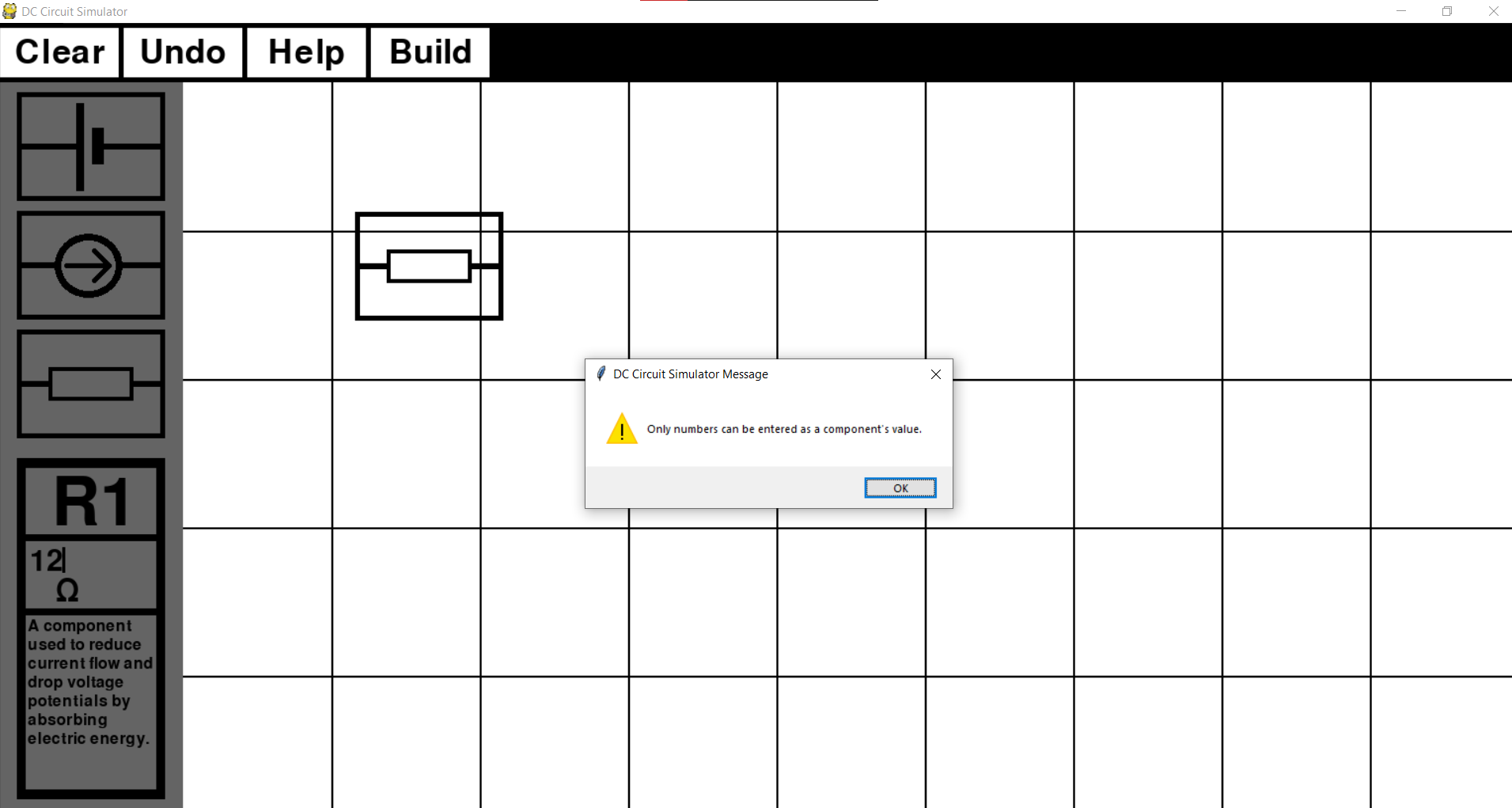
Screenshot evidence for test 9: 

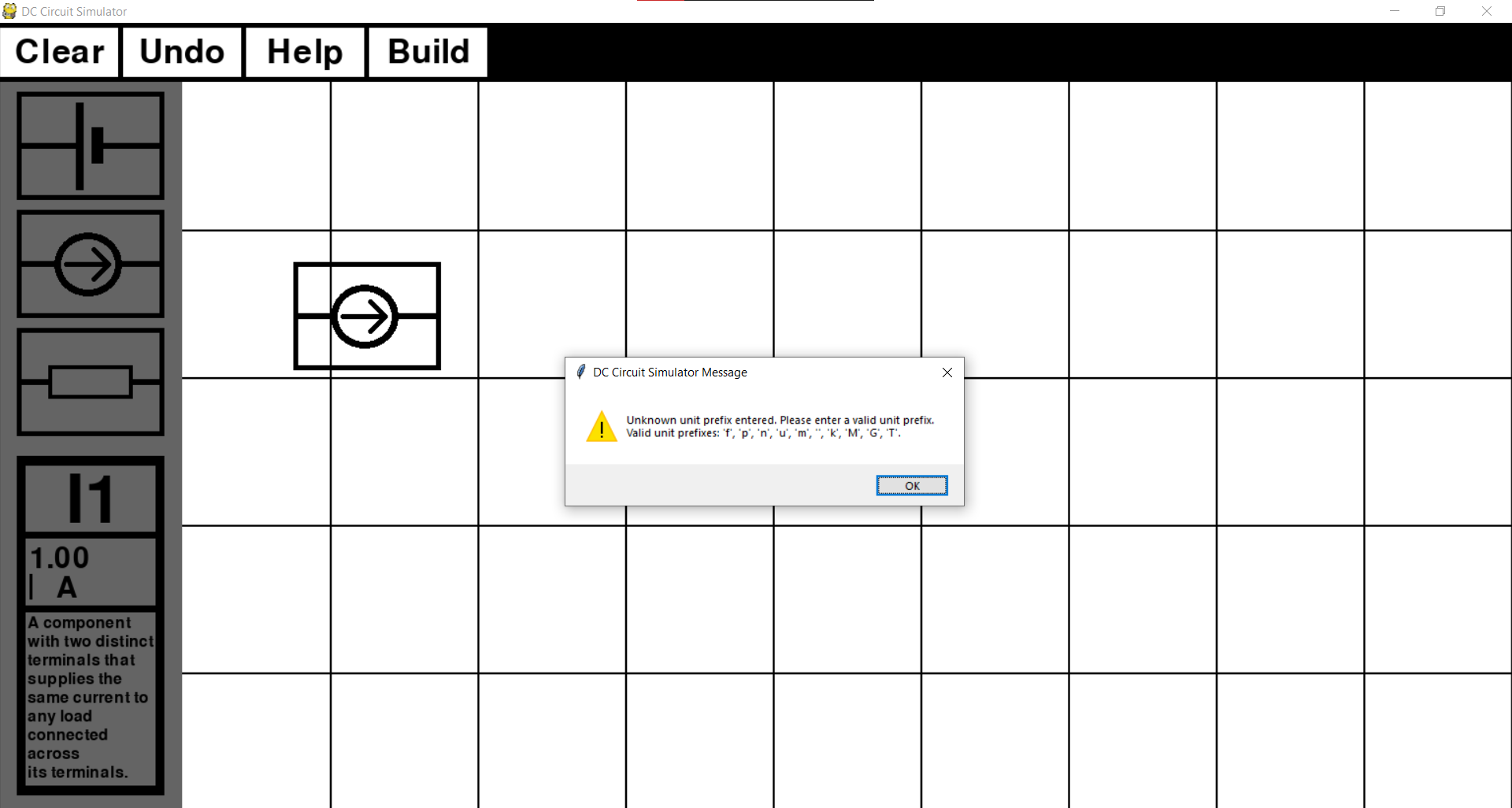
Screenshot evidence for test 10: 

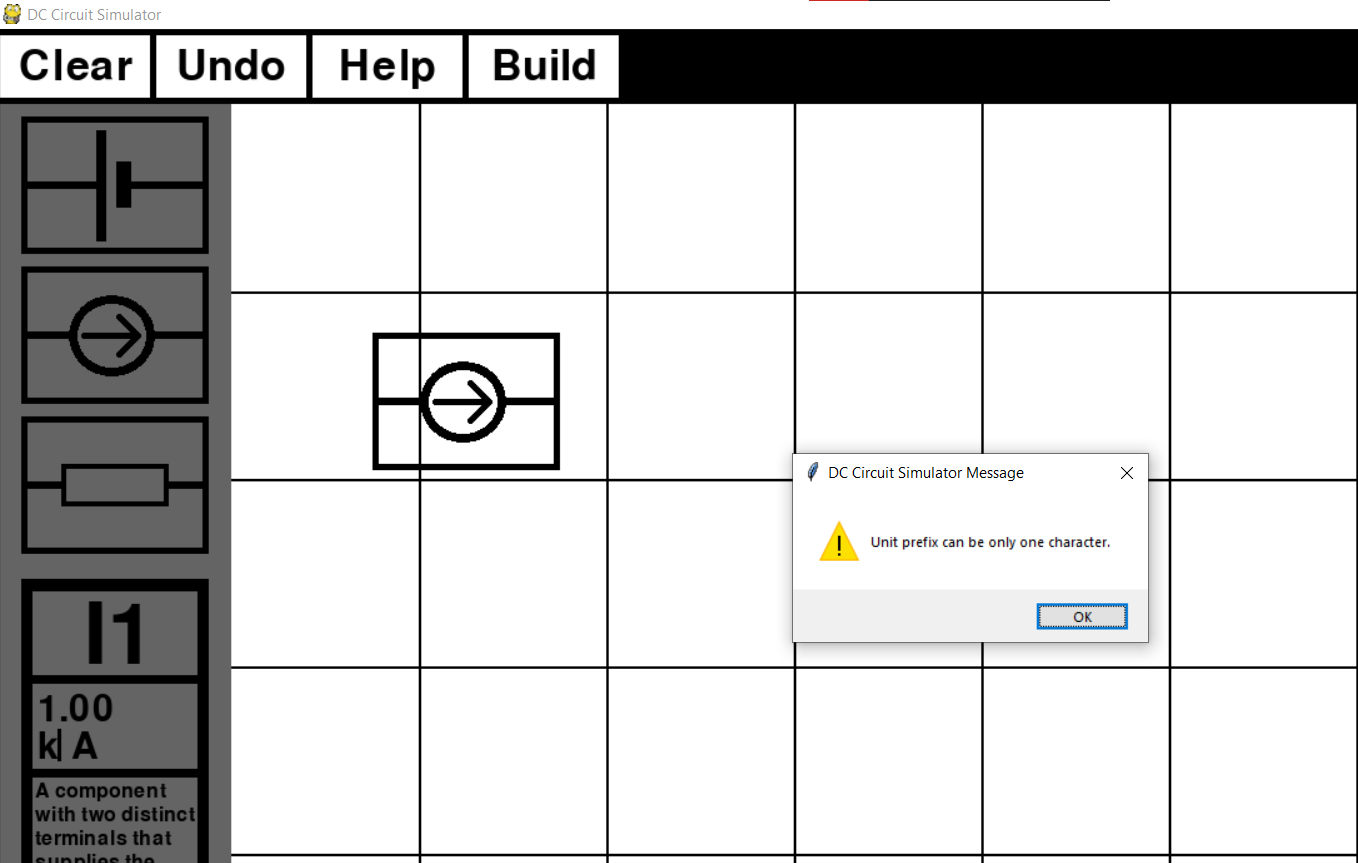
Screenshot evidence for test 11, 12:

Screenshot evidence for test 13:

Screenshot evidence for test 14: 

Screenshot evidence for test 15: 

Screenshot evidence for test 15, 16: 

Screenshot evidence for test 17: 

## Objective 2

Objective 2 builds up on idea of giving user the ability to construct a circuit. With objective 2 I wanted to provide additional features to the simulator which can help the user to construct a circuit but are not necessary to do so. Below is my objective 2.

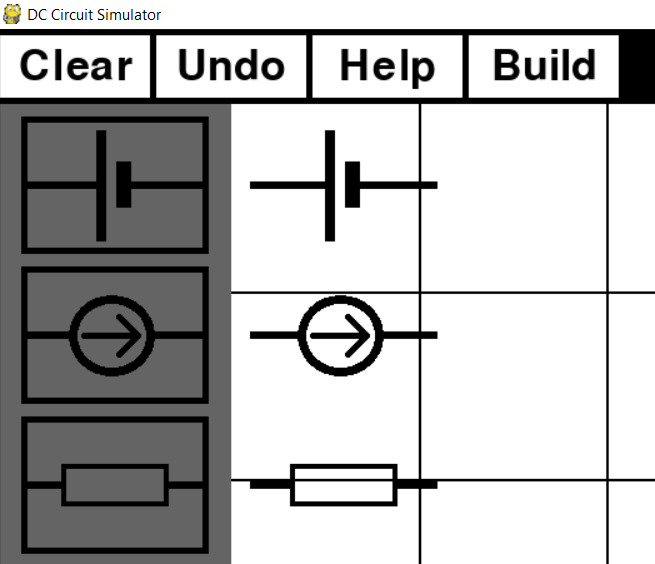
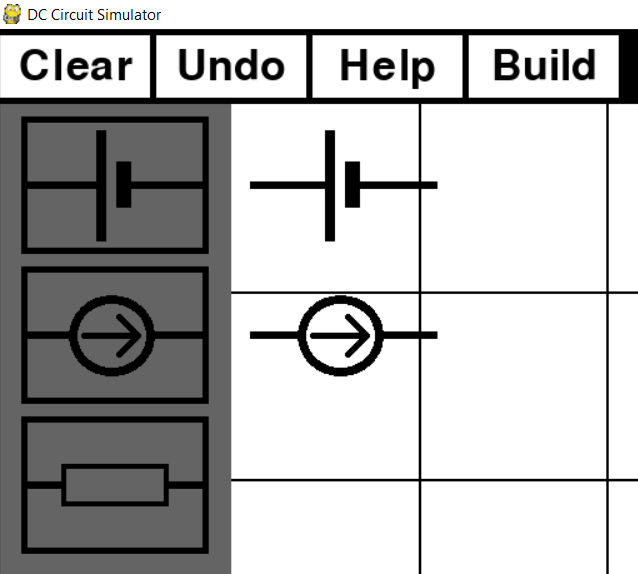
1. *In case the user makes a mistake while building a circuit (adds a wrong component to the main tab, makes a wrong connection between two components, etc.) they should be able to correct it by clicking on the specific buttons in the toolbar.*
   1. *A button which will undo the user’s latest action should be included, so that if the user notices their mistake immediately, they can quickly correct it.*
   2. *In case a whole circuit is built in an incorrect way or the user would like to build a new circuit, the user should be able to clear the whole main tab. This button will delete all components which are currently there.*

Test table is below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Objective number being tested** | **Test description** | **User input** | **Expected outcome** | **Actual outcome** | **Comment on the outcome**  **(Pass/Fail)** |
| 1 | 2.1 | Three components are added to the main tab and ‘undo’ button is pressed. | Left mouse click on ‘undo’ button. | Last component added to the main tab gets deleted. | Last component added to the main tab was deleted. | A user can delete last component added to the main tab.  **Pass** |
| 2 | 2.1 | Latest link made between two components should be deleted with the ‘undo’ button. | User connects three components together and presses ‘undo’ button. | Last link made between two components gets deleted. | Last component added to the main tab was deleted together with all its links. | A user was not able to delete last link made.  **Fail** |
| 3 | 2.2 | Three components are added to the main tab and ‘clear’ button is pressed. | Left mouse click on ‘clear’ button. | All three components get deleted from main tab. | All three components get deleted from main tab. | The user can ‘clear’ all components from the main tab.  **Pass** |
| 4 | 2.2 | Three components connected in a network should get deleted together with their links. | Left mouse click on ‘clear’ button. | All three components get deleted from main tab together with their links. | All three components get deleted from main tab together with their links. | The user can ‘clear’ all components and their links from the main tab.  **Pass** |

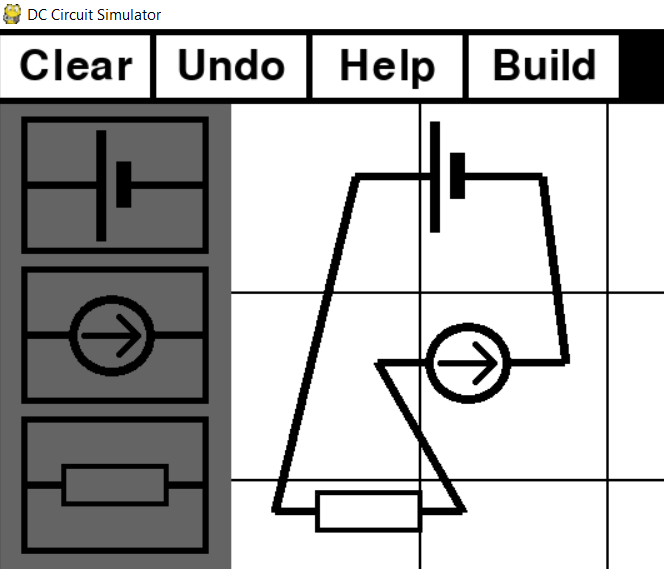
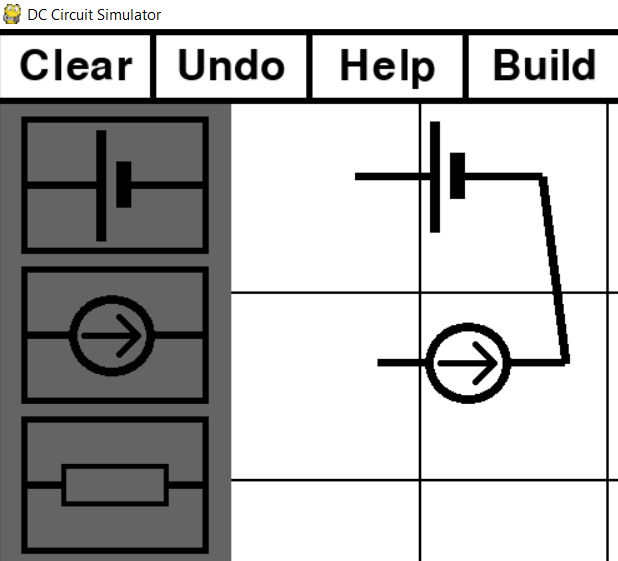
Screenshot evidence for test 1:

Before undo button was pressed: After undo button was pressed:

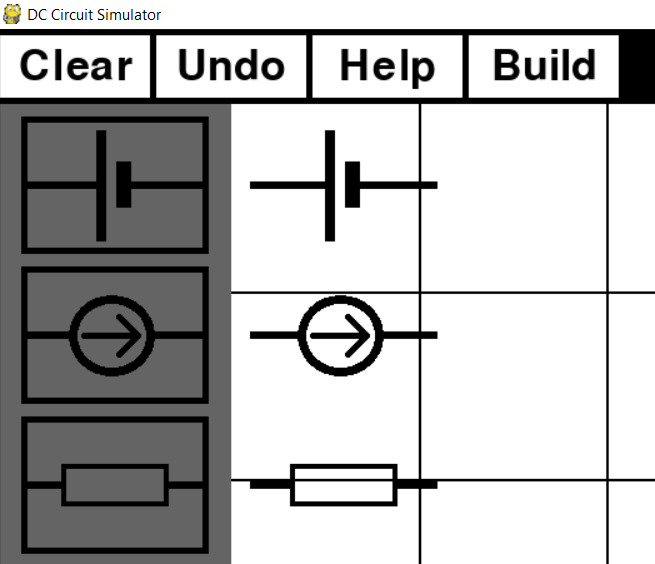
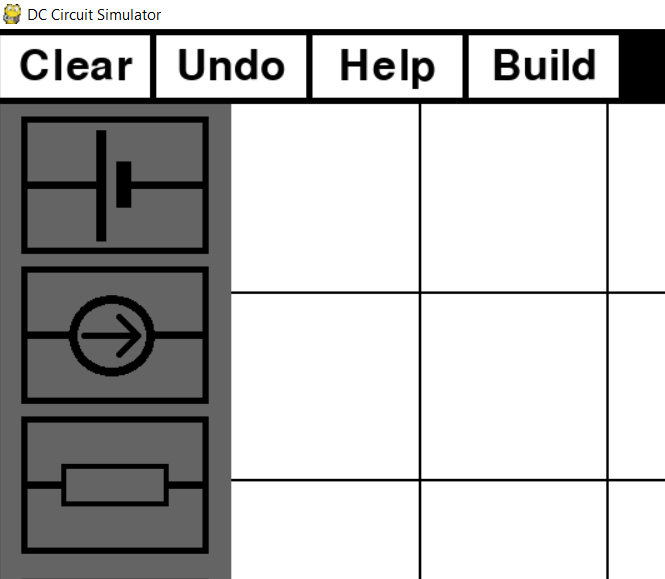
Screenshot evidence for test 2:

Before undo button was pressed: After undo button was pressed:

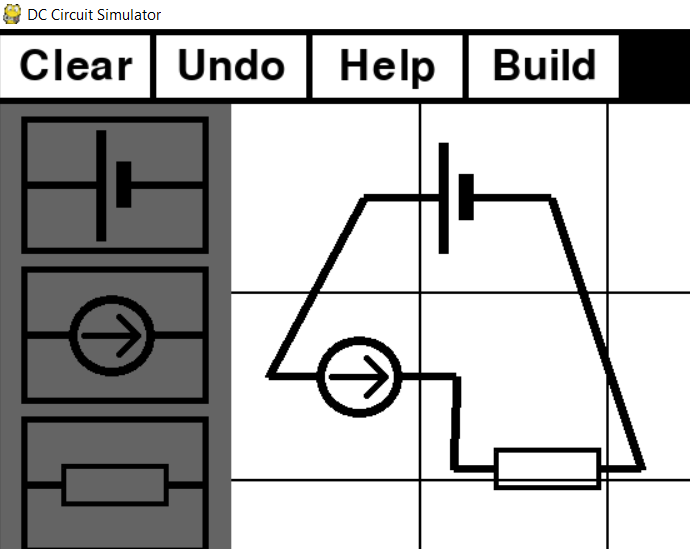
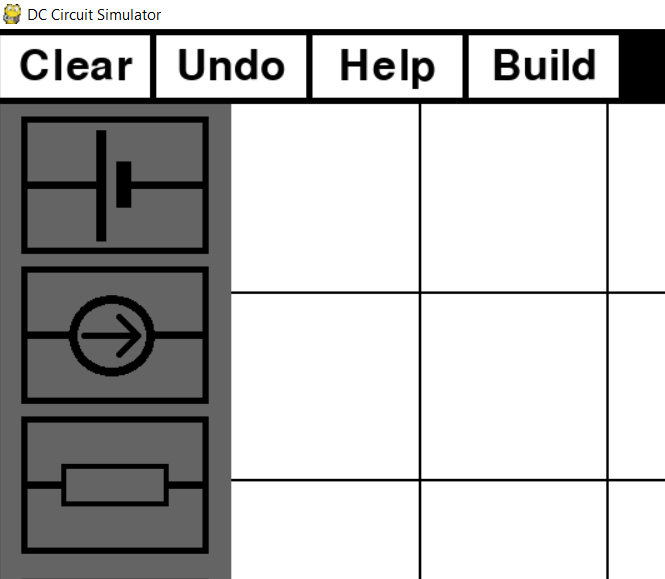
Screenshot evidence for test 3:

Before clear button was pressed: After clear button was pressed:

Screenshot evidence for test 3:

Before clear button was pressed: After clear button was pressed:

## Objective 3

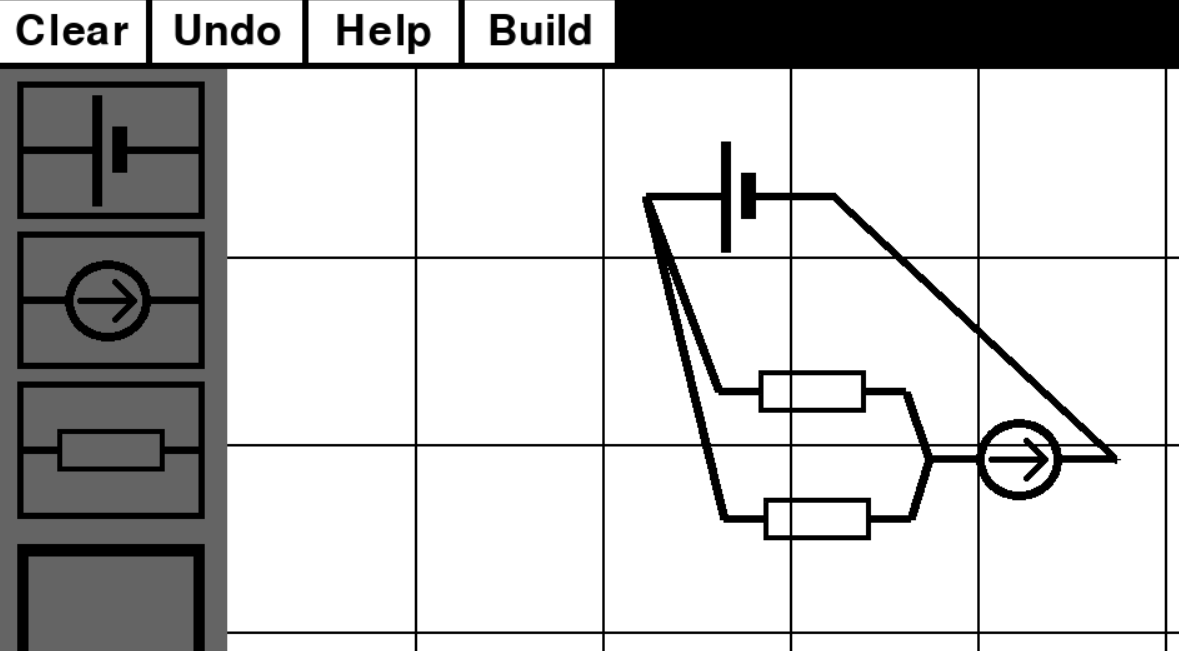
With objective 3 I aimed for my program to gather information it of the build circuit and covert it into a circuit netlist.

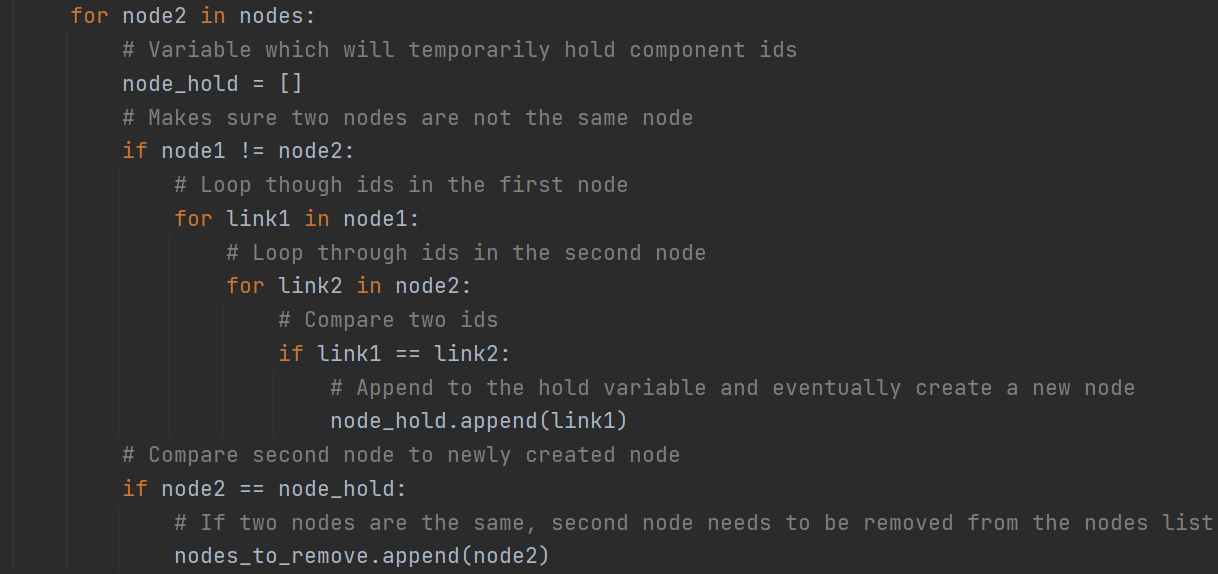
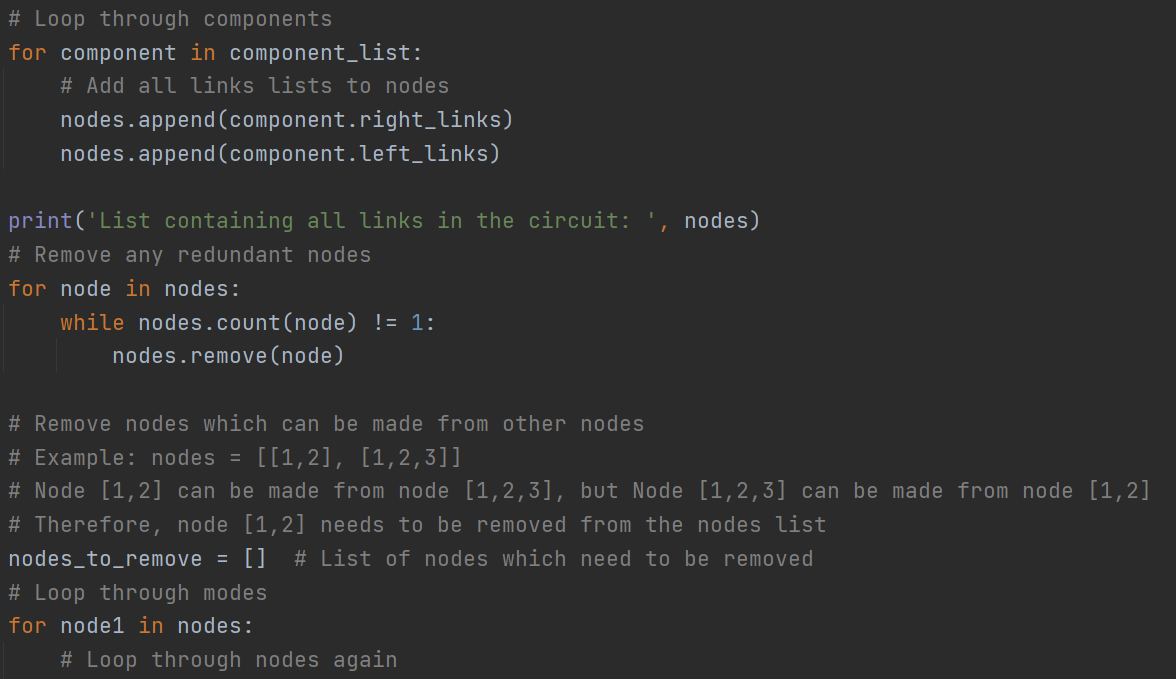
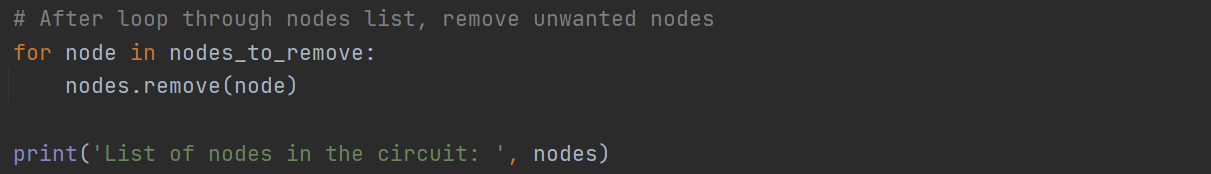
1. *Once a circuit is completed, it should be analysed and converted into a format which computer can understand.*
   1. *Once the analysis of the circuit is complete, a netlist in a text format should be created and stored as an external file.*
   2. *If the program detects an error, analysis should be stopped, and an appropriate message should be displayed informing the user that circuit was built in an incorrect way.*

First two tests show that the simulator takes data about links between components, converts it into a one list containing nodes and then makes a netlists file. To test the program’s validation of a circuit, a number of incorrect circuits will be built and analysed by the program.

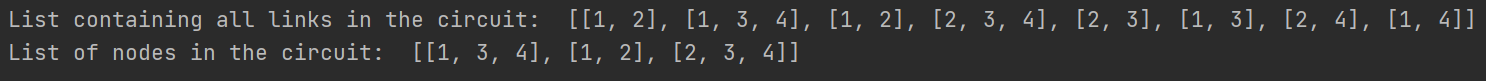
Test table is below.

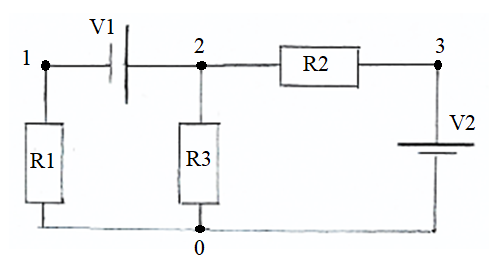
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Objective number being tested** | **Test description** | **User input** | **Expected outcome** | **Actual outcome** | **Comment on the outcome**  **(Pass/Fail)** |
| 1 | 3.1 | From a list containing circuit links, a nodes list needs to be created | Create a network between components | List of nodes in that circuit. | List containing only circuit nodes | From a links list a nodes list was created.  **Pass** |
| 2 | 3.1 | Test the way a program creates a circuit netlist by creating an example circuit from the analysis section. | Create a circuit same as the example circuit in analysis. | Same netlist as the example’s circuit netlist. | Netlist with identical component names and values. Even though nodes are not identical, they are still both correct. Same nodes are represented with different numbers, e.g. node represented with ‘2’ in example circuit is represented as ‘0’ node in the netlist circuit. | Netlist names, nodes and values correspond to the example circuit.  **Pass** |
| 3 | 3.2 | Try analysing a circuit with no components in the main tab. | Press built button with empty main tab. | Appropriate message occurs warning the user about their mistake. | ‘Not enough components to construct a circuit.’ is displayed. | Circuit is rejected and appropriate warning is displayed.  **Pass** |
| 4 | 3.2 | Try analysing a circuit with only one component in the main tab. | Add one component and press built button. | Appropriate message occurs warning the user about their mistake. | ‘Not enough components to construct a circuit.’ is displayed. | Circuit is rejected and appropriate warning is displayed.  **Pass** |
| 5 | 3.2 | Try analysing a circuit with two components in the main tab. | Add two components and press built button. | Appropriate message occurs warning the user about their mistake. | ‘Not enough components to construct a circuit.’ is displayed. | Circuit is rejected and appropriate warning is displayed.  **Pass** |
| 6 | 3.2 | Try analysing a circuit with three components in the main tab and no connections between them. | Add three components and press built button. | Appropriate message occurs warning the user about their mistake. | ‘Not all components are connected in a circuit. Please connect all components.‘ is displayed. | Circuit is rejected and appropriate warning is displayed.  **Pass** |
| 7 | 3.2 | Try analysing a circuit with three components connected together and one component with no connections. | Connect three components and press built button. | Appropriate message occurs warning the user about their mistake. | ‘Not all components are connected in a circuit. Please connect all components.‘ is displayed. | Circuit is rejected and appropriate warning is displayed.  **Pass** |
| 8 | 3.2 | Try analysing a circuit with four components connected together except for component one terminal. | Connect four components and press build button. | Appropriate message occurs warning the user about their mistake. | ‘Not all components are connected in a circuit. Please connect all components.‘ is displayed. | Circuit is rejected and appropriate warning is displayed.  **Pass** |

Screenshot evidence for test 1:

Screenshot of code for creating and printing links list and nodes list: 

Code output:



Evidence for test 2:

Example netlist:

V1 2 1 9

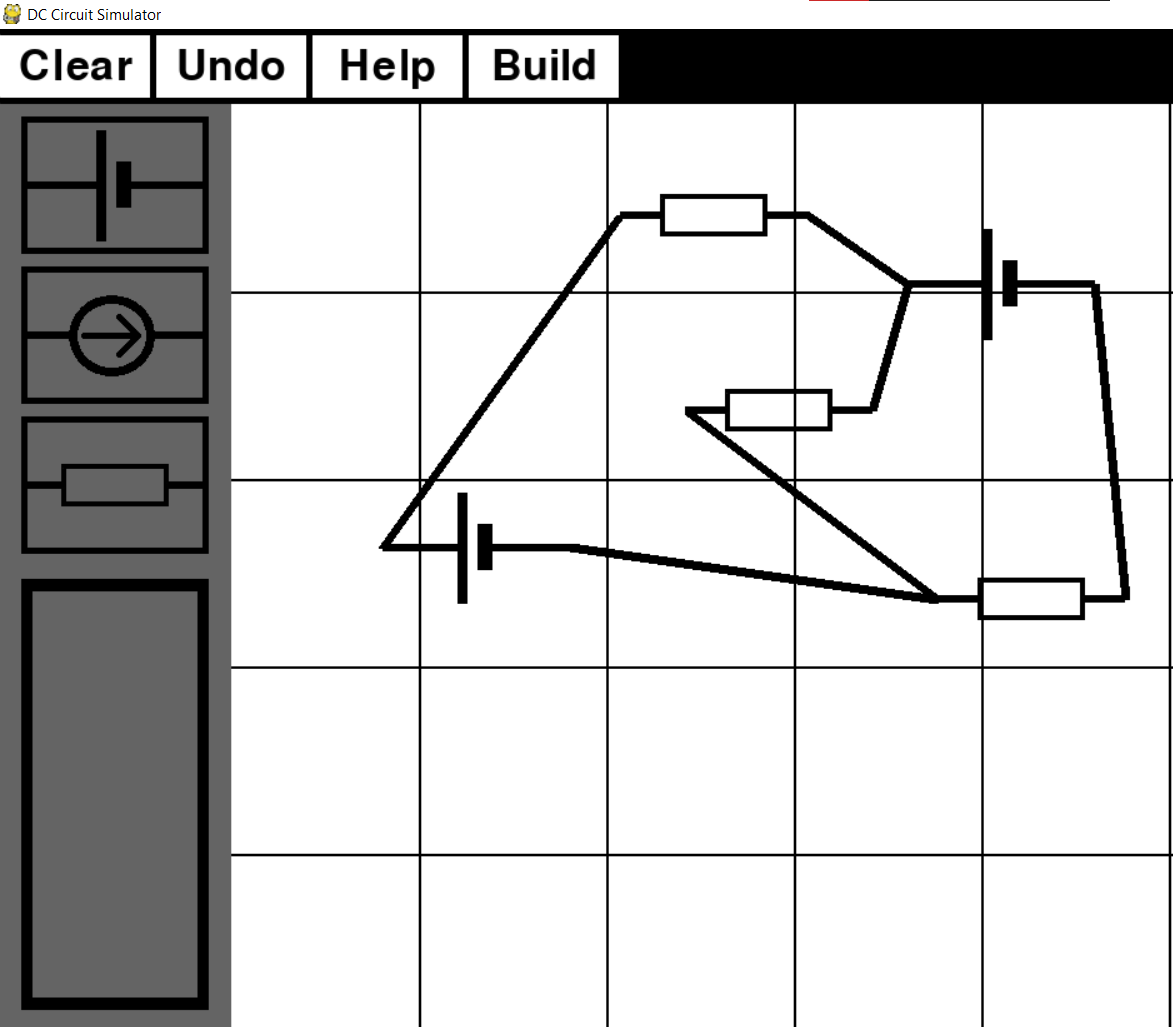
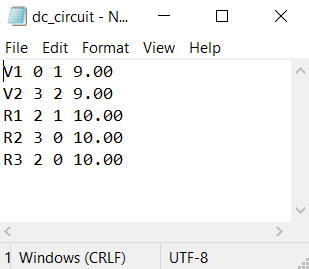
V2 0 3 9

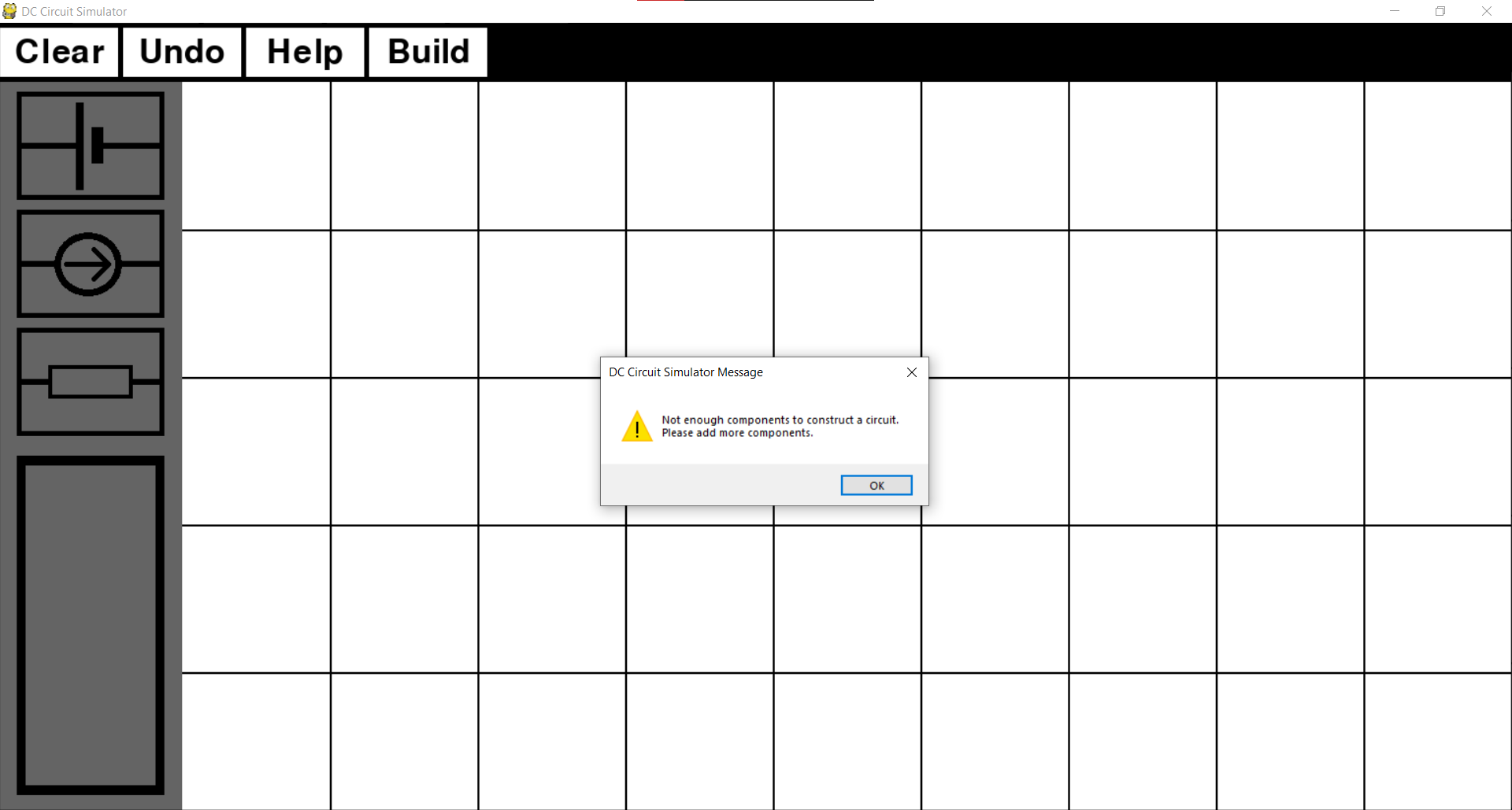
R1 1 0 10

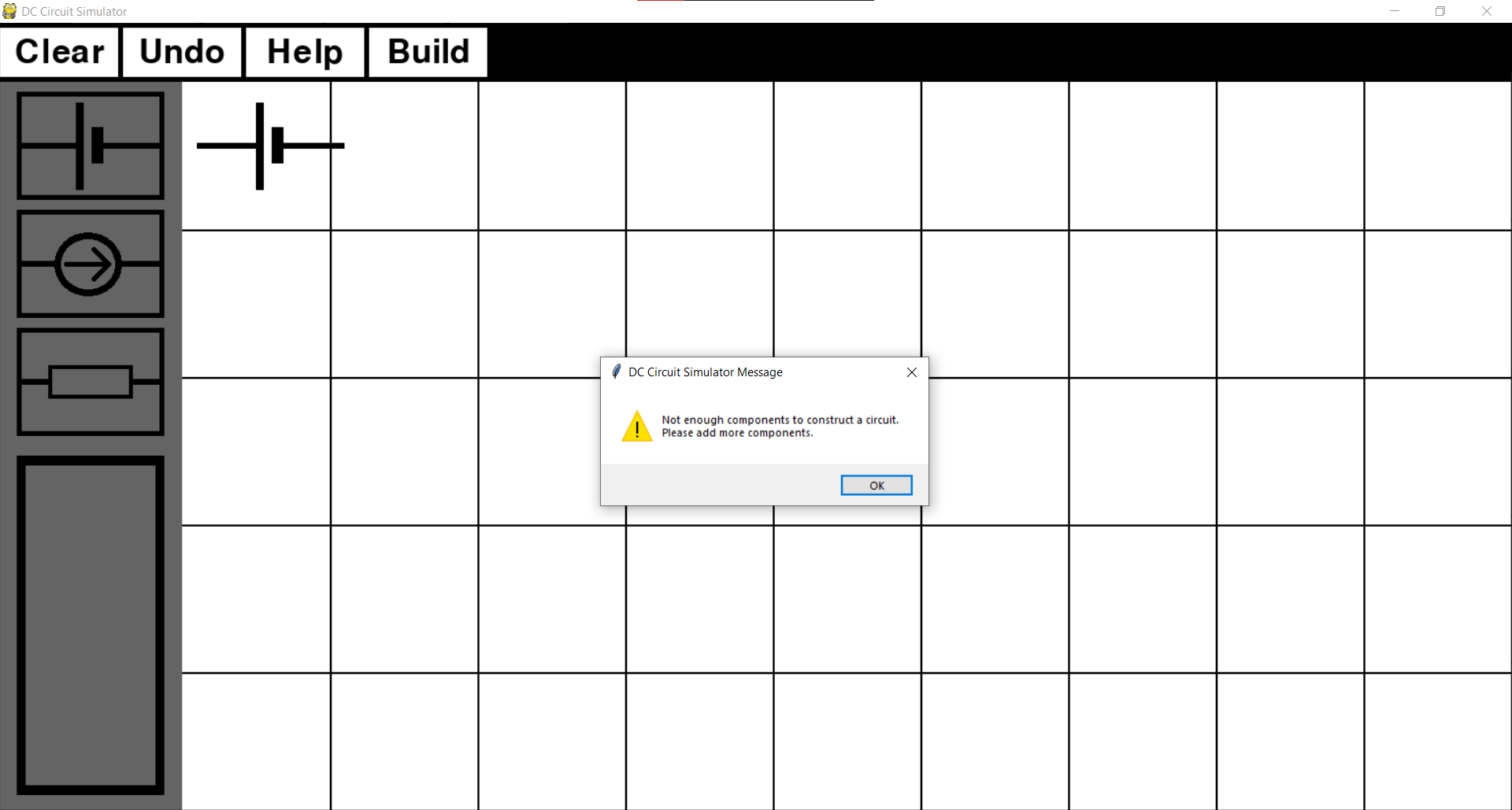
R2 2 3 10

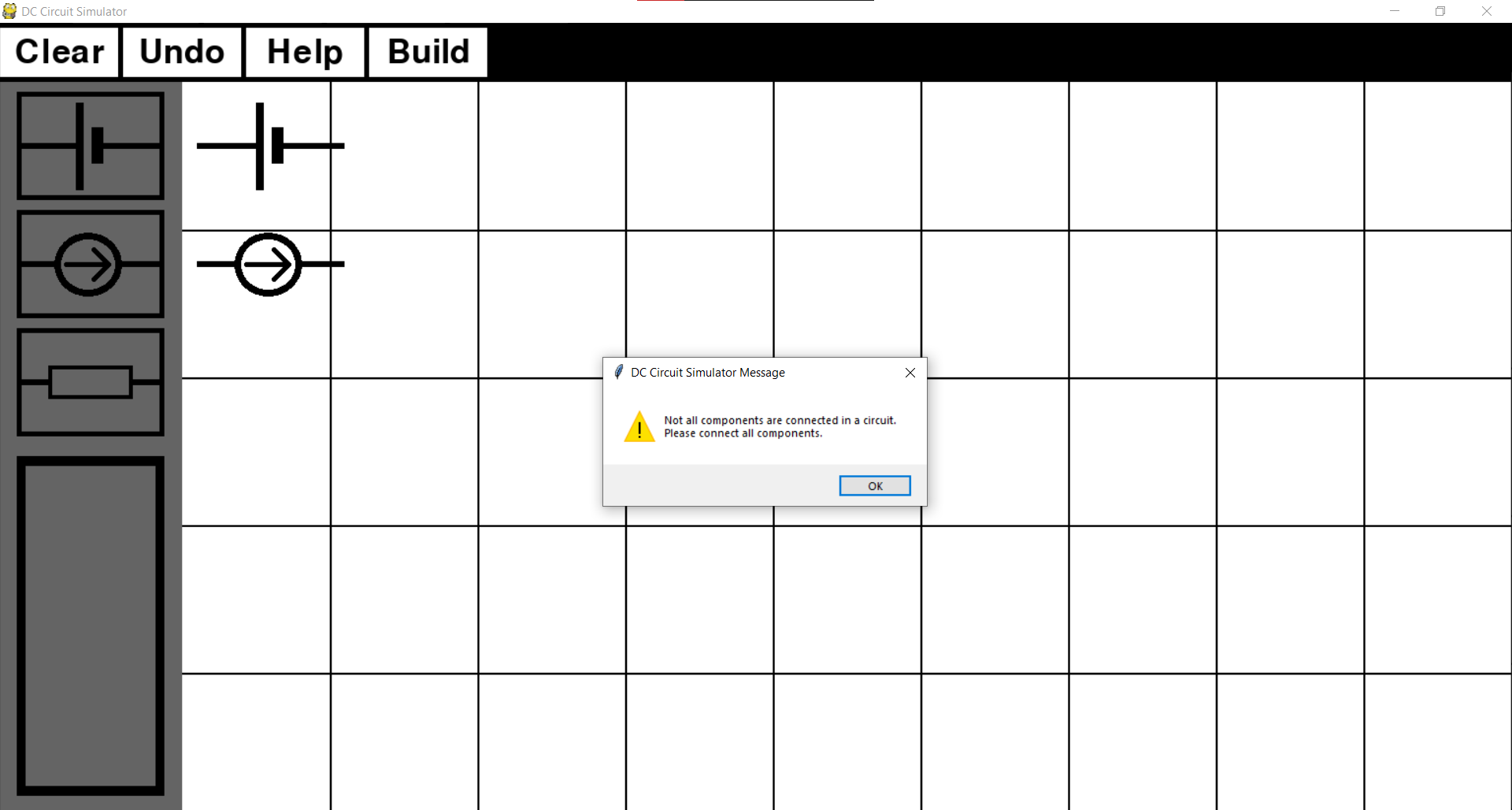
R3 2 0 10

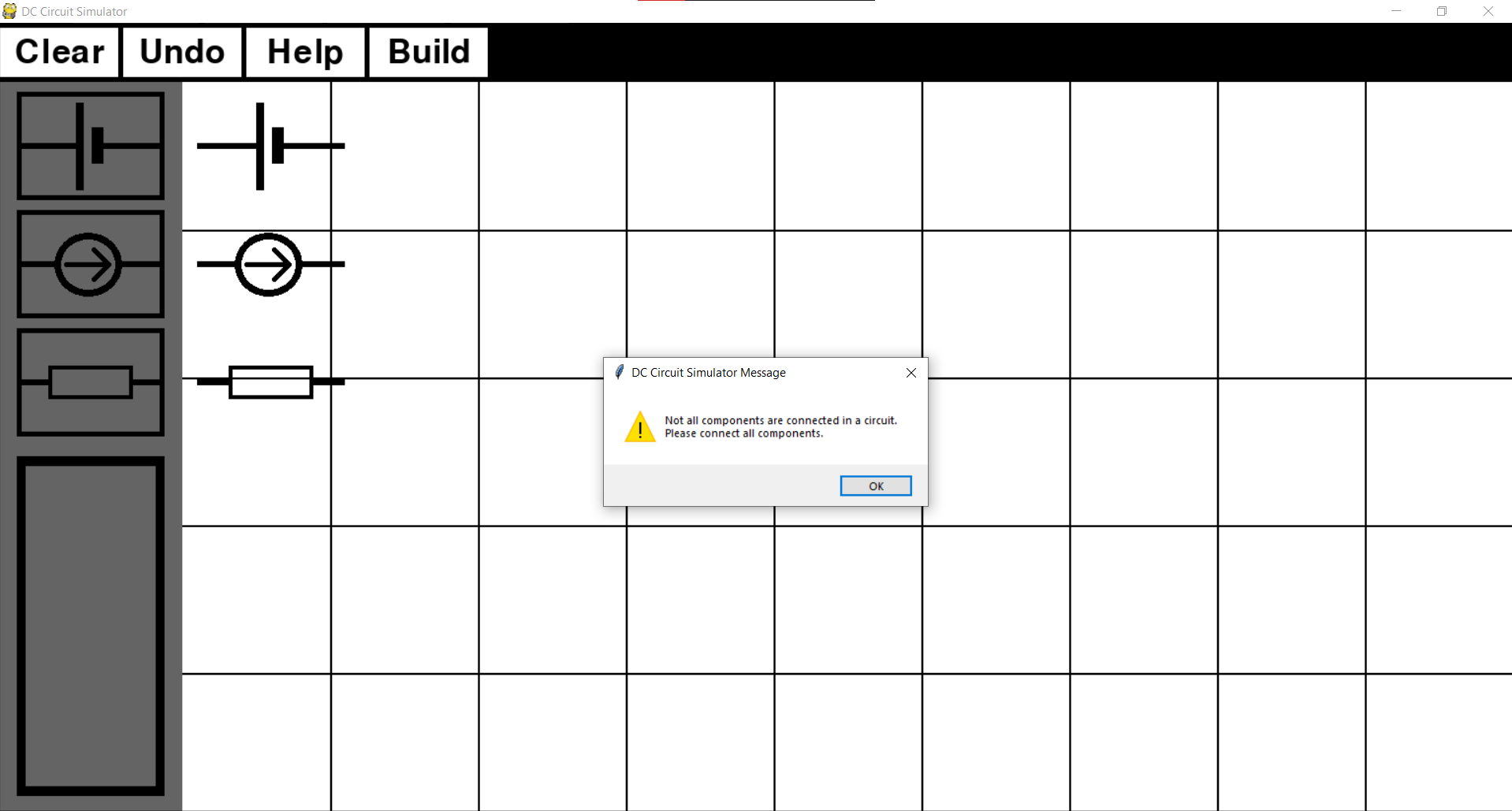
Screenshot of a circuit built in the simulator: Circuit netlist:

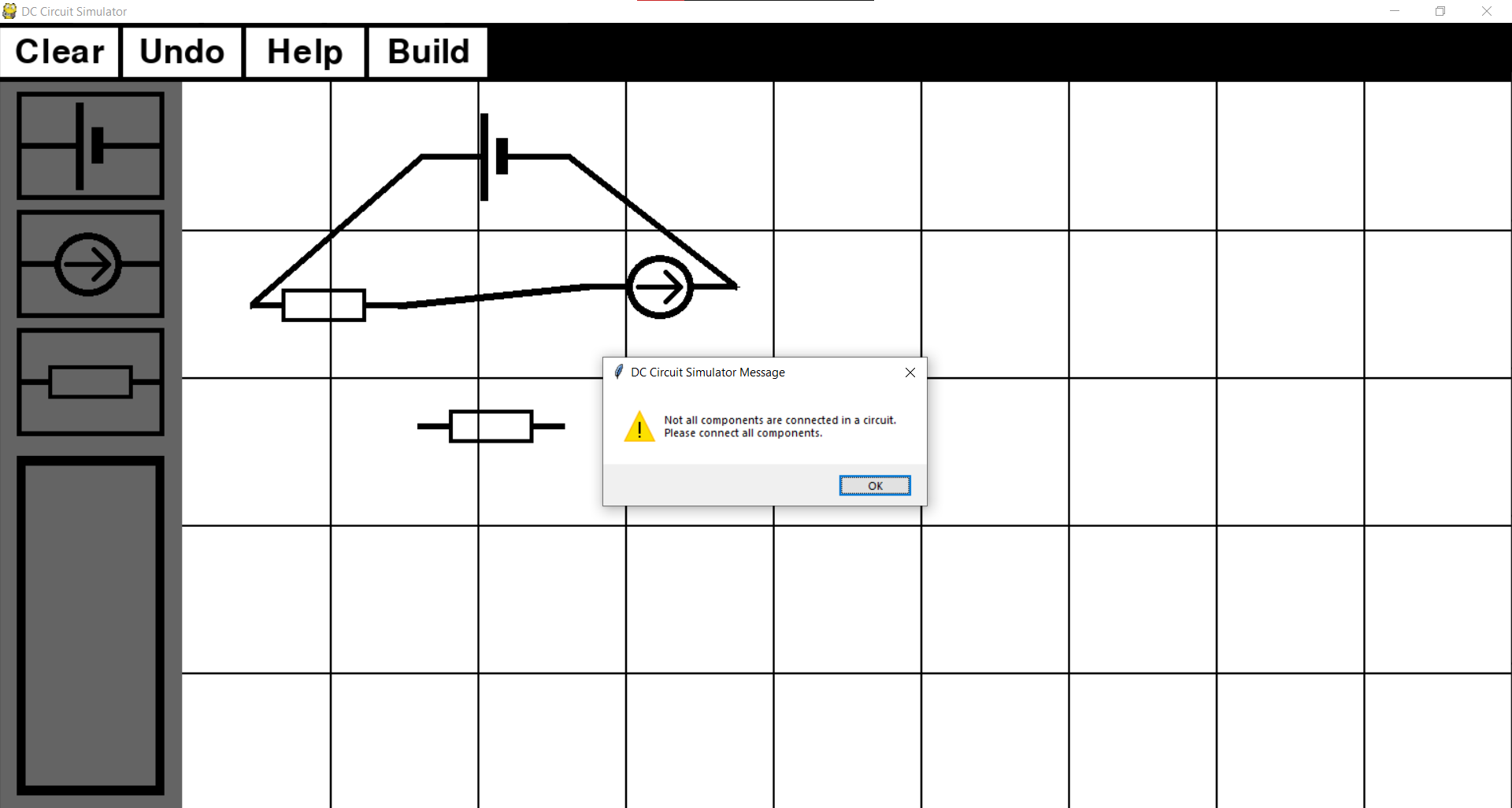
 

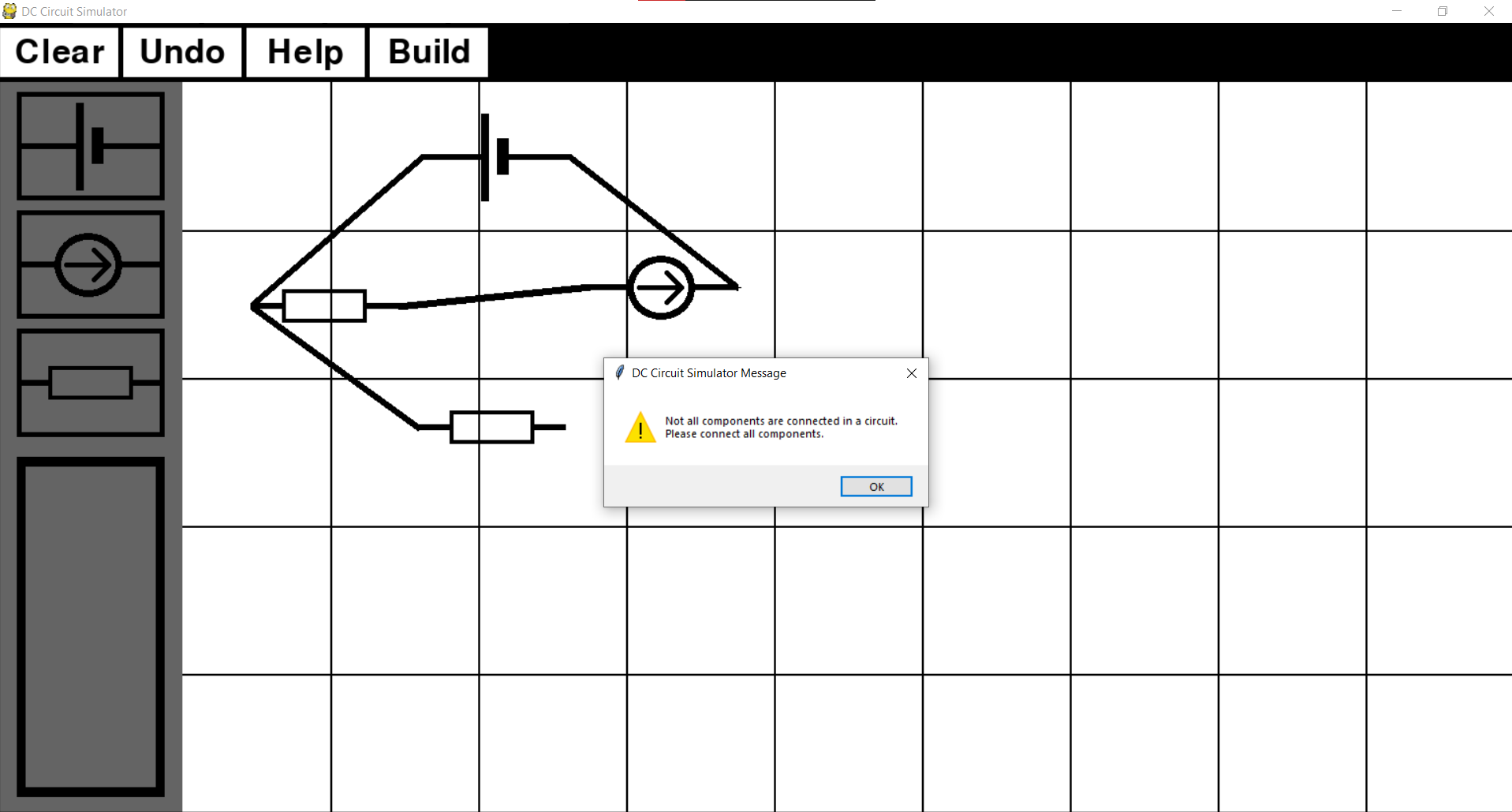
Screenshot evidence for test 3: 

Screenshot evidence for test 4: 

Screenshot evidence for test 5: 

Screenshot evidence for test 6: 

Screenshot evidence for test 7: 

Screenshot evidence for test 8: 

## Objective 4

My aim with objective 4 was to calculate voltages across each node and current flowing through every independent voltage source in the circuit. These are the most important values to know when solving a circuit question.

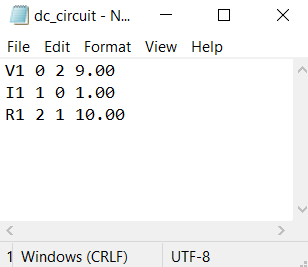
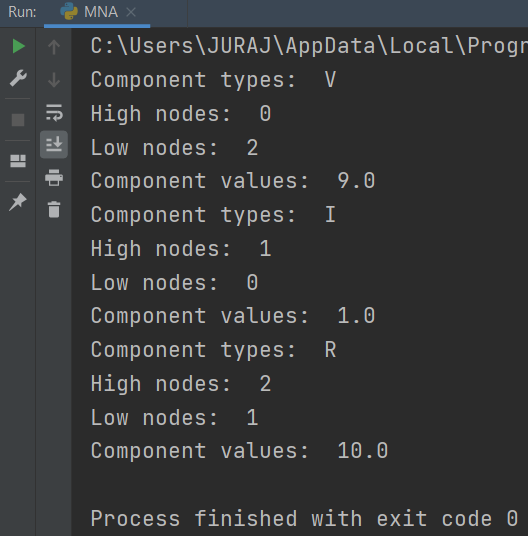
1. *Once a netlist is stored in a text file, MNA will be initialised and once it is complete, final results will be displayed to the user.*
   1. *The program should begin MNA analysis by processing the information of a netlist. This will be done by first parsing a netlist file and then mapping string nodes from netlist file into integer nodes which represent indices of the MNA matrix*
   2. *Construct A matrix by combining sub-matrices G, B, C and D*
   3. *Construct z matrix using the i matrix containing the voltages of independent voltage and e matrix containing the currents of the independent current sources.*
   4. *Solve the linear matrix equation to obtain x matrix*
   5. *Once x matrix is calculated, display MNA results to the user.*
   6. *If the circuit is shorted or for any other reason cannot be calculated so it produces and error, appropriate message should be displayed informing the user that circuit was incorrectly built.*

MNA will be tested by first checking if it passes the file in a correct way (do parsed values correspond to original values from the netlist). Then I will check if the results obtained by running the MNA and solving matrix equation, are correct by comparing it to an example circuit. Finally, I will test the MNA system by building circuit with normal boundary and erroneous data.

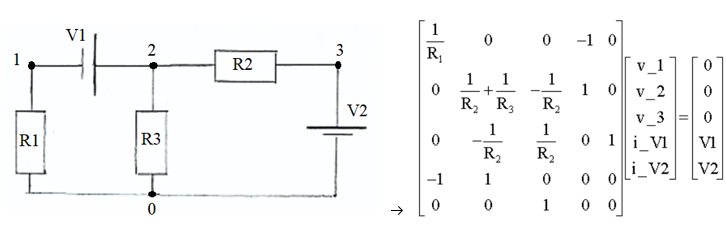
Test table is below

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Objective number being tested** | **Test description** | **User input** | **Expected outcome** | **Actual outcome** | **Comment on the outcome**  **(Pass/Fail)** |
| 1 | 4.1. | Check if MNA parses the netlist text file correctly. | User constructs a circuit and presses ‘build’ button | Parsed values corresponding to netlist file | Values correspond to each other | File gets parsed correctly.  **Pass** |
| 2 | 4.4 | Check if calculated x matrix corresponds to nodal voltages and currents flowing through voltage sources. | User constructs a circuit and presses ‘build’ button. | x matrix needs to be show correct circuit values. | x matrix has correct circuit values. | x matrix gives correct values.  **Pass** |
| 3 | 4.5 | User needs to see calculated results. | User constructs a circuit and presses ‘build’ button. | x matrix needs to be displayed to the user with explanation what each value represents. | Values displayed with using Tkinter message box appropriate explanation | User can see the circuit results.  **Pass** |
| 4 | 4.6 | Create a circuit with serial connections  (normal). | User constructs a circuit and presses ‘build’ button. | Output calculated circuit values. | Circuit values outputted. | Values calculated and displayed to the user.  **Pass** |
| 5 | 4.6 | Create a circuit with parallel connections (normal). | User constructs a circuit and presses ‘build’ button. | Output calculated circuit values. | Circuit values outputted. | Values calculated and displayed to the user.  **Pass** |
| 6 | 4.6 | Create a circuit with one incorrect parallel connection (boundary). | User constructs a circuit and presses ‘build’ button. | Display an appropriate warning to the user. | Program crashed. | A user should have been warned about their mistake.  **Fail** |
| 7 | 4.6 | Create a circuit which is shorted. (erroneous) | User constructs a circuit and presses ‘build’ button. | Display an appropriate warning to the user. | ‘Circuit was built in an incorrect way.’ is displayed. | Error detected, circuit is rejected, and user is warned about their mistake.  **Pass** |
| 8 | 4.6 | Create a circuit with 10 components where every component has a connection with every other component (erroneous). | User constructs a circuit and presses ‘build’ button. | Display an appropriate warning to the user. | ‘Circuit was built in an incorrect way.’ is displayed. | Error detected, circuit is rejected, and user is warned about their mistake.  **Pass** |
| 9 | 4.6 | Create a circuit with 20 components. | User constructs a circuit and presses ‘build’ button. | Output calculated circuit values. | Circuit values outputted. | Values calculated and displayed to the user.  **Pass** |

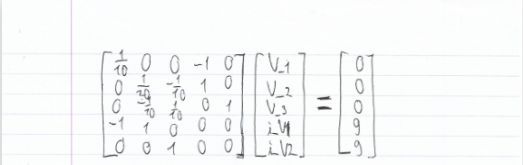
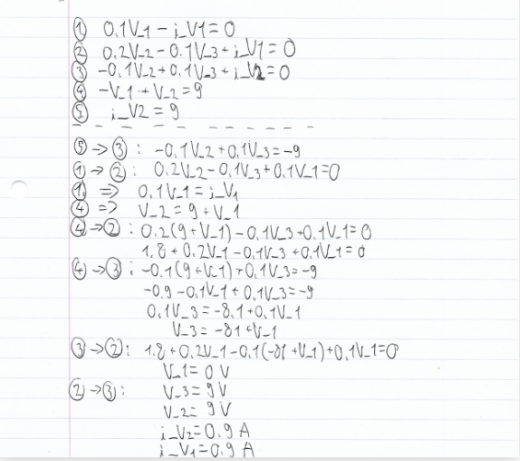
Evidence for test 1:

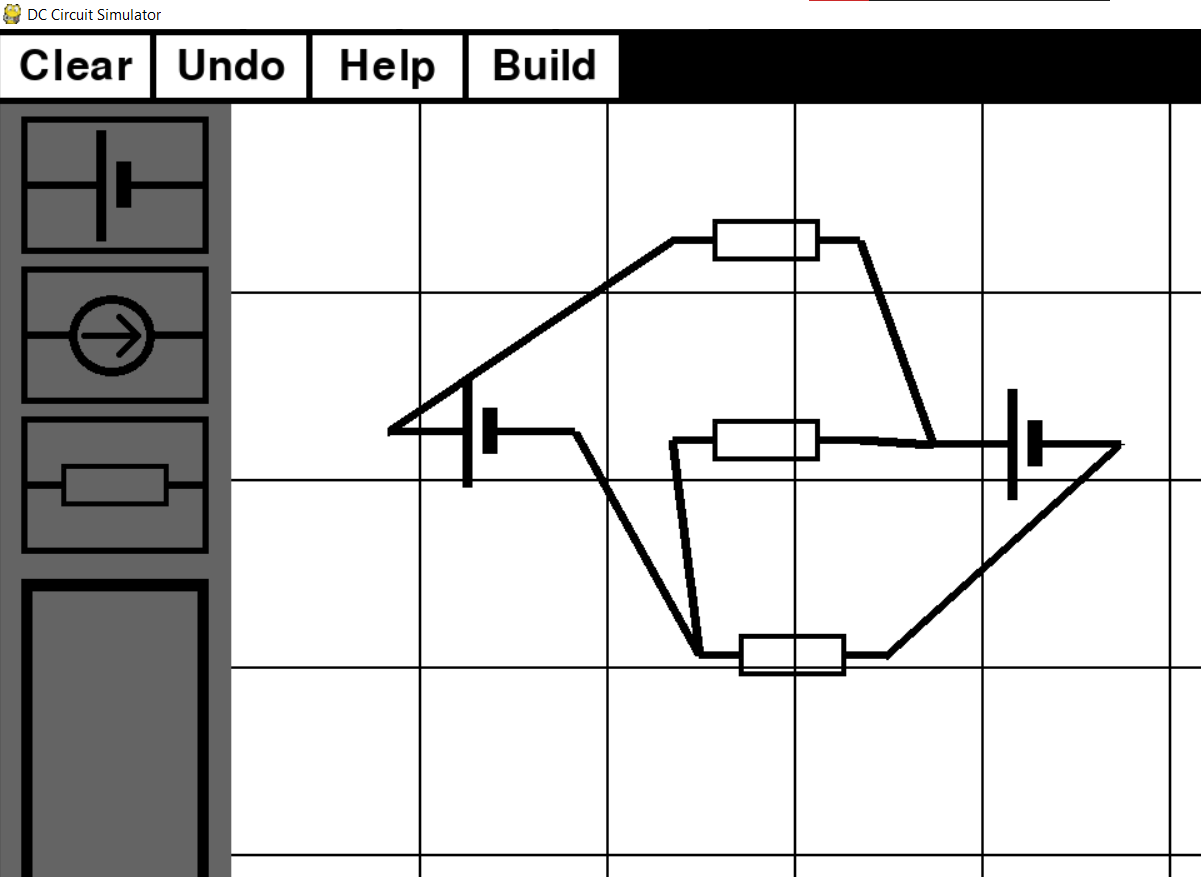
Netlist file: Parsed values: 

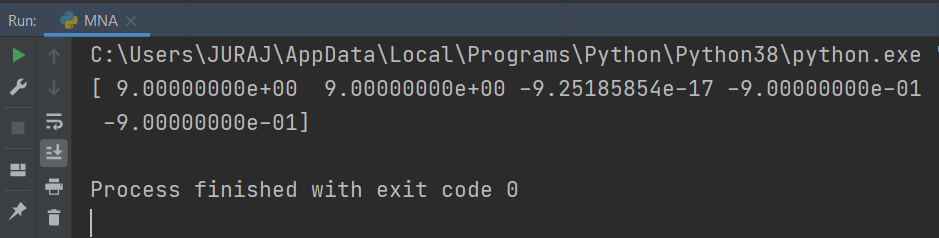
Evidence for test 2

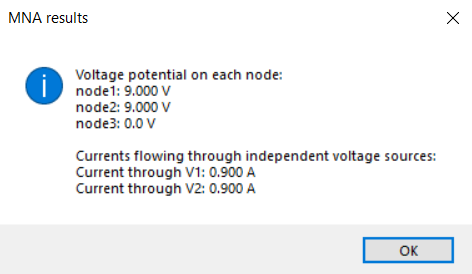
Example circuit with its matrix equation: 

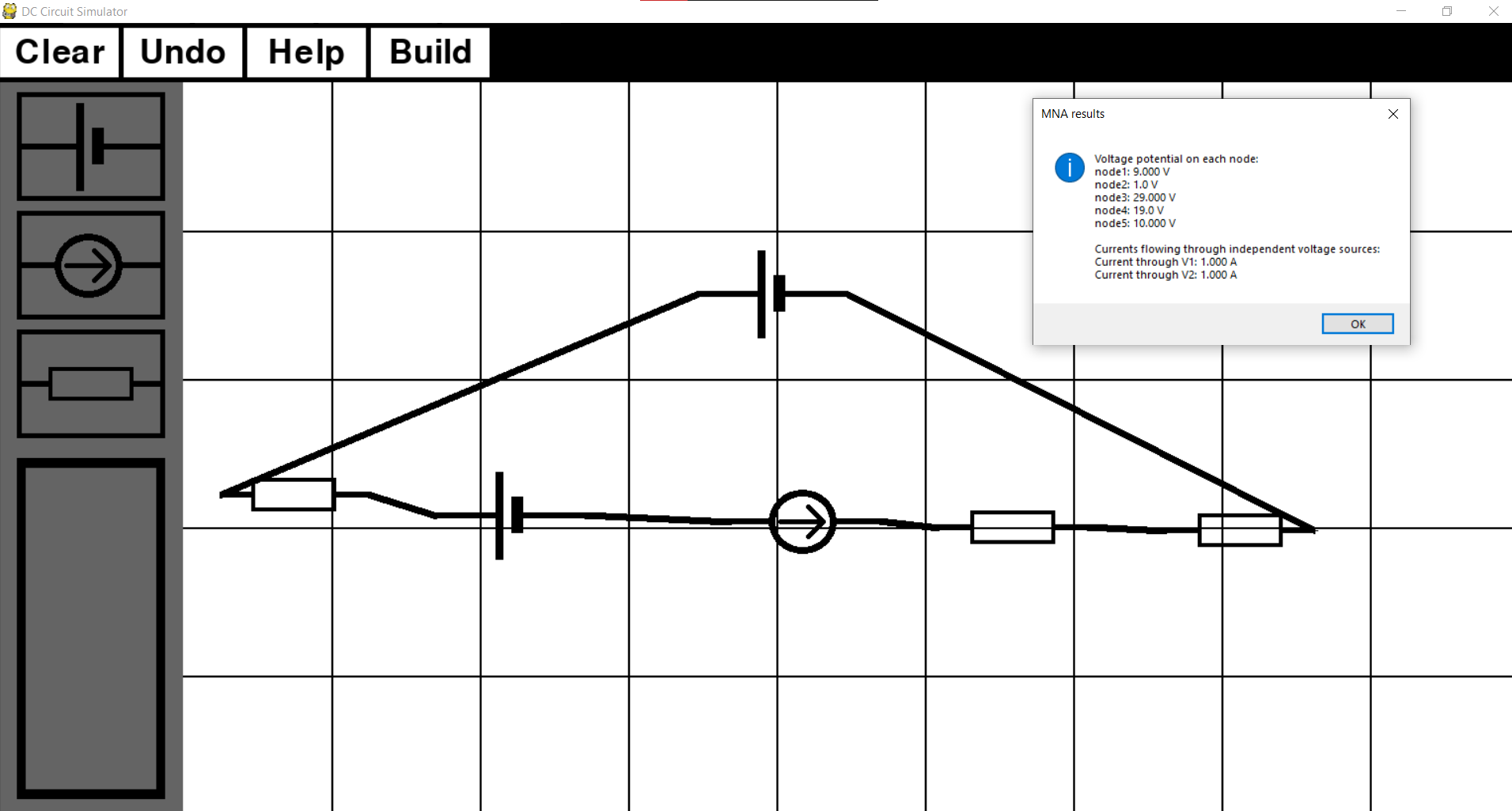
Lets us say that every resistor has value of 10 ohms and evrey independent voltage source has value of 9 volts. Below is a matrix calculation done by hand.

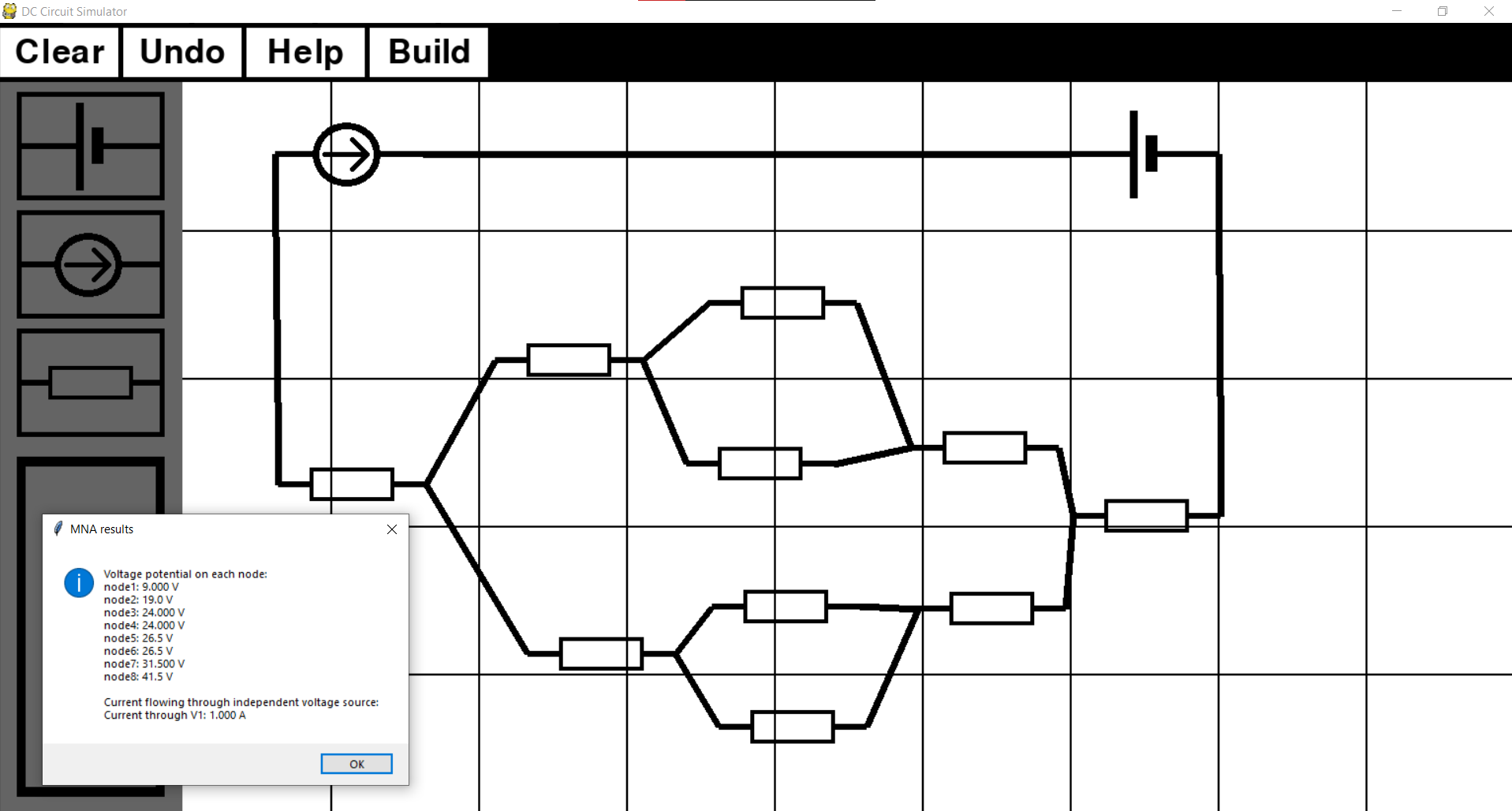
 

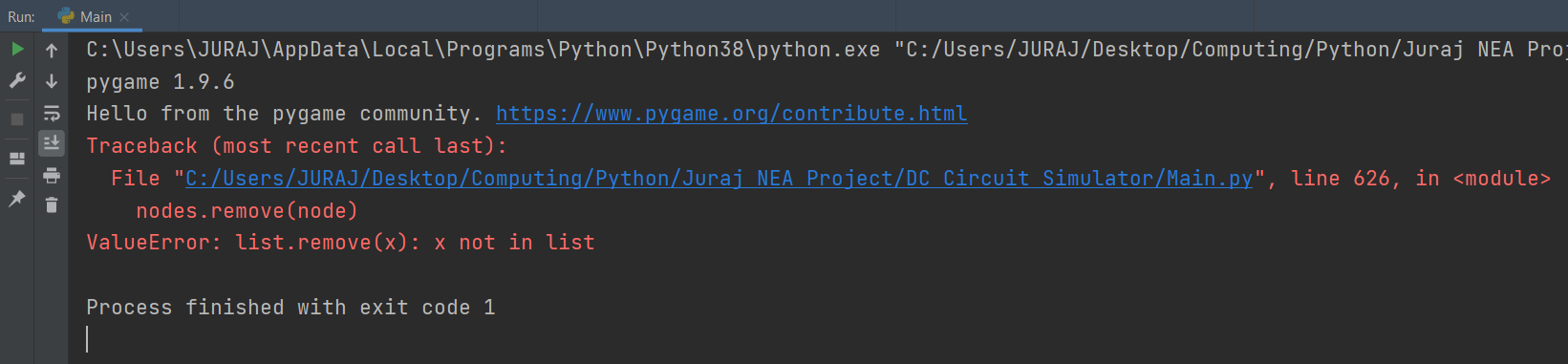
Screenshot of example circuit buit in the simulator: 

Screenshot of x matrix:

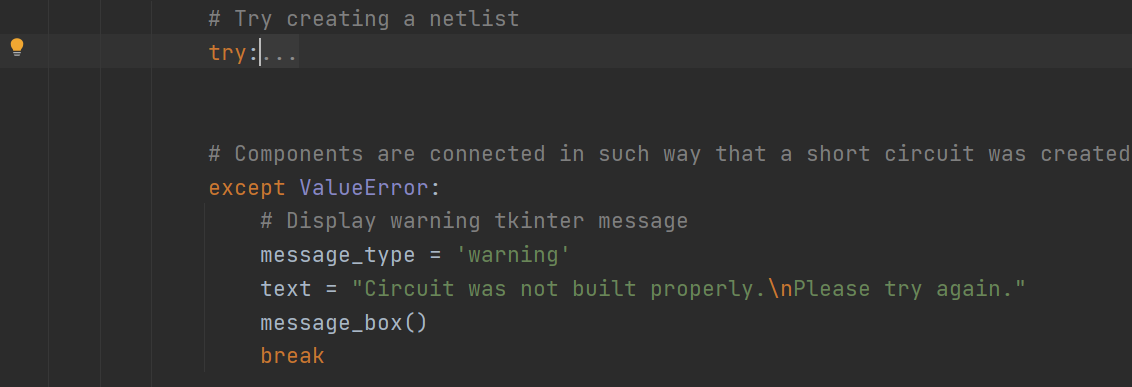
Screenshot evidence for test 3 (example circuit was used): 

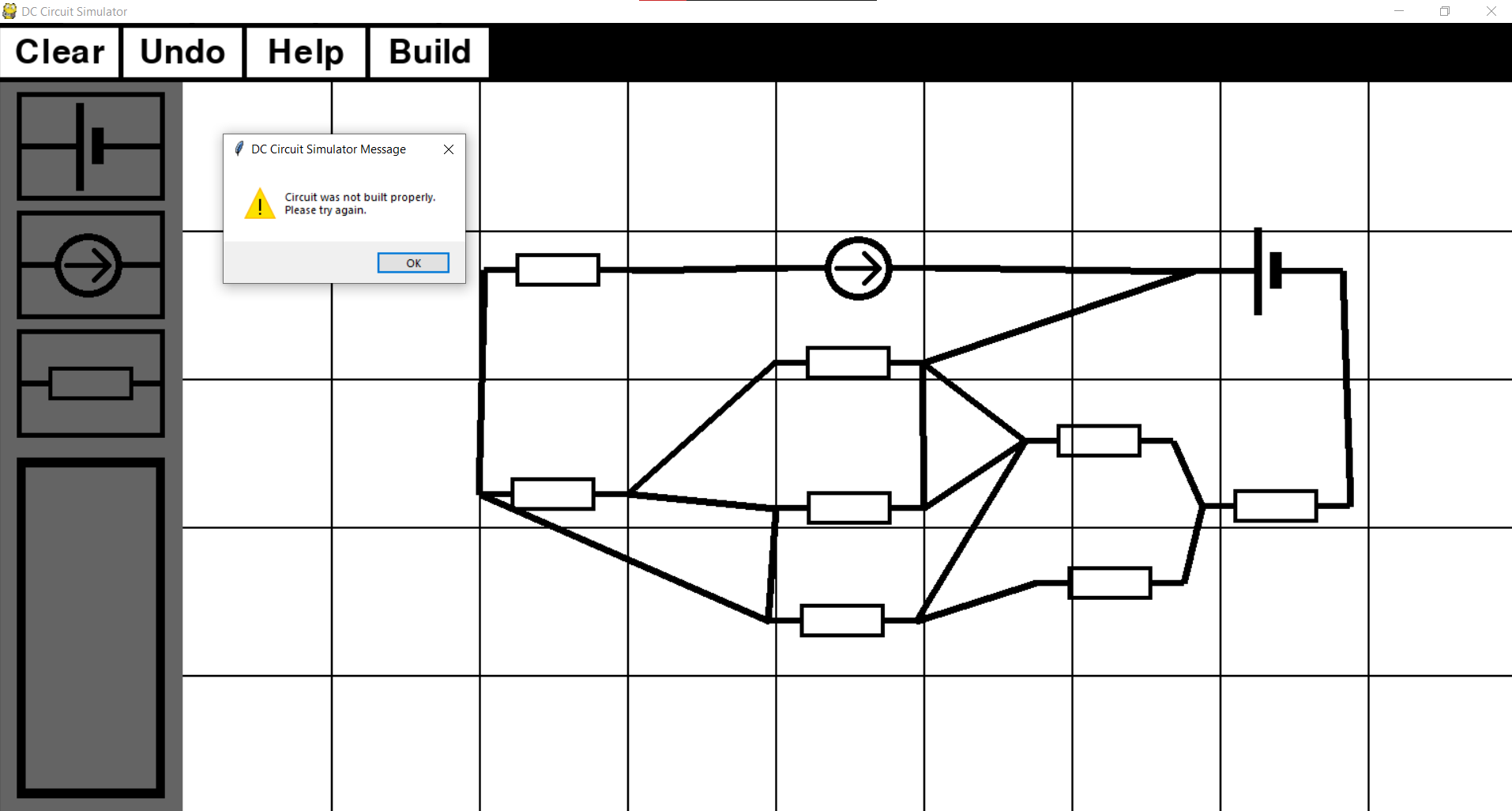
Screenshot evidence for test 4: 

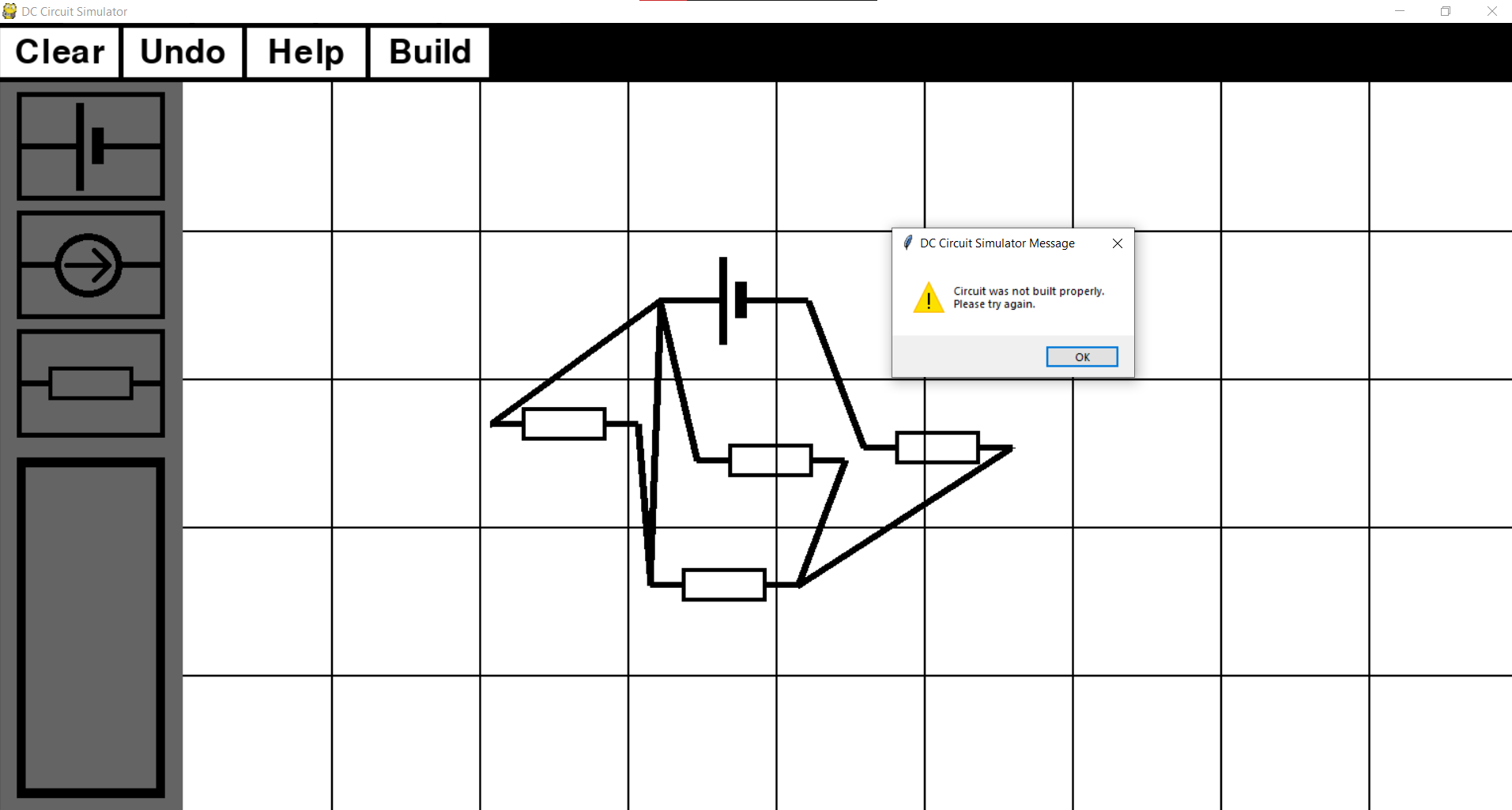
Screenshot evidence for test 5:

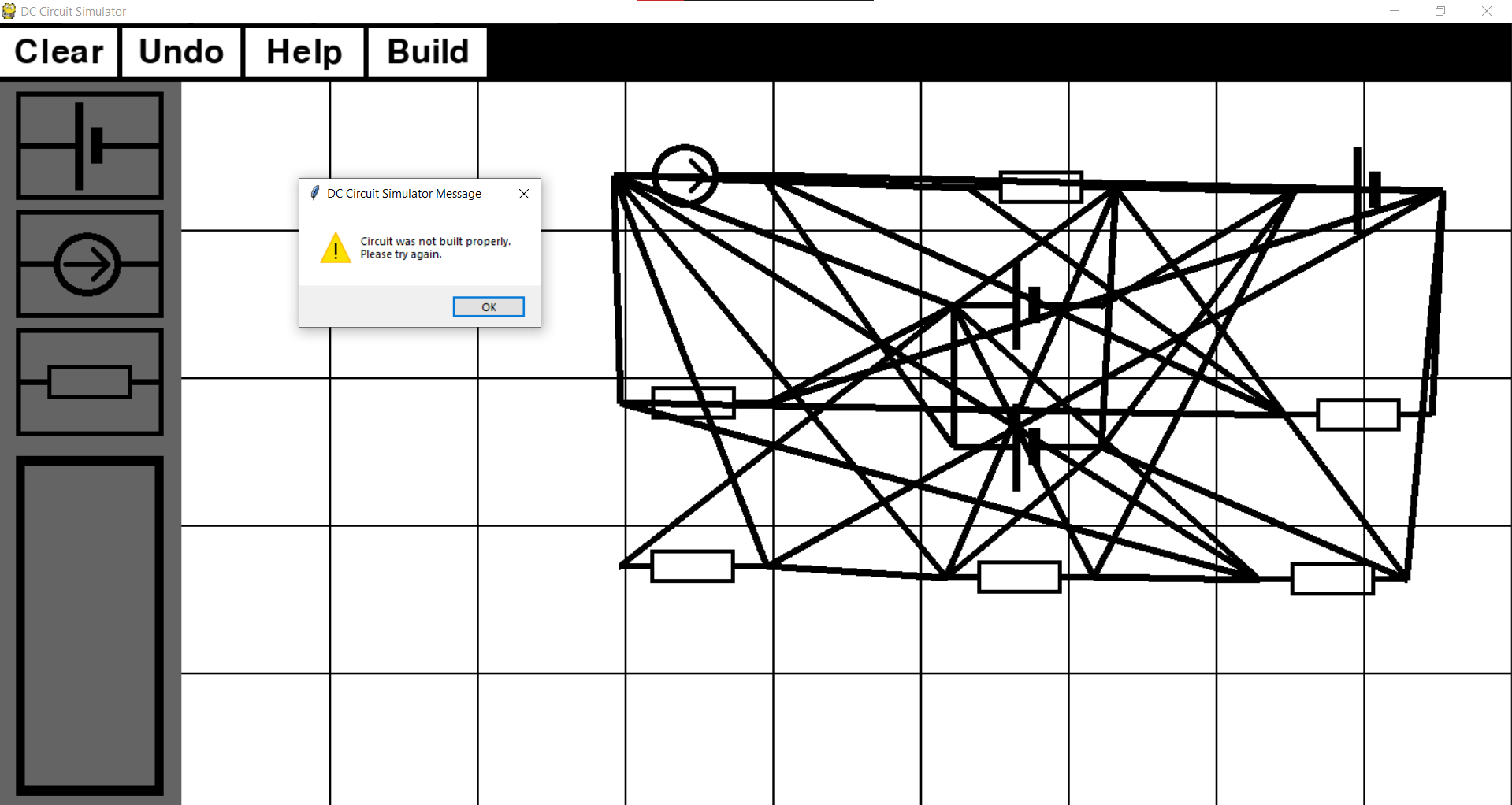
Screenshot evidence for test 6: 

To fix this error I will add try and except statement to the code:

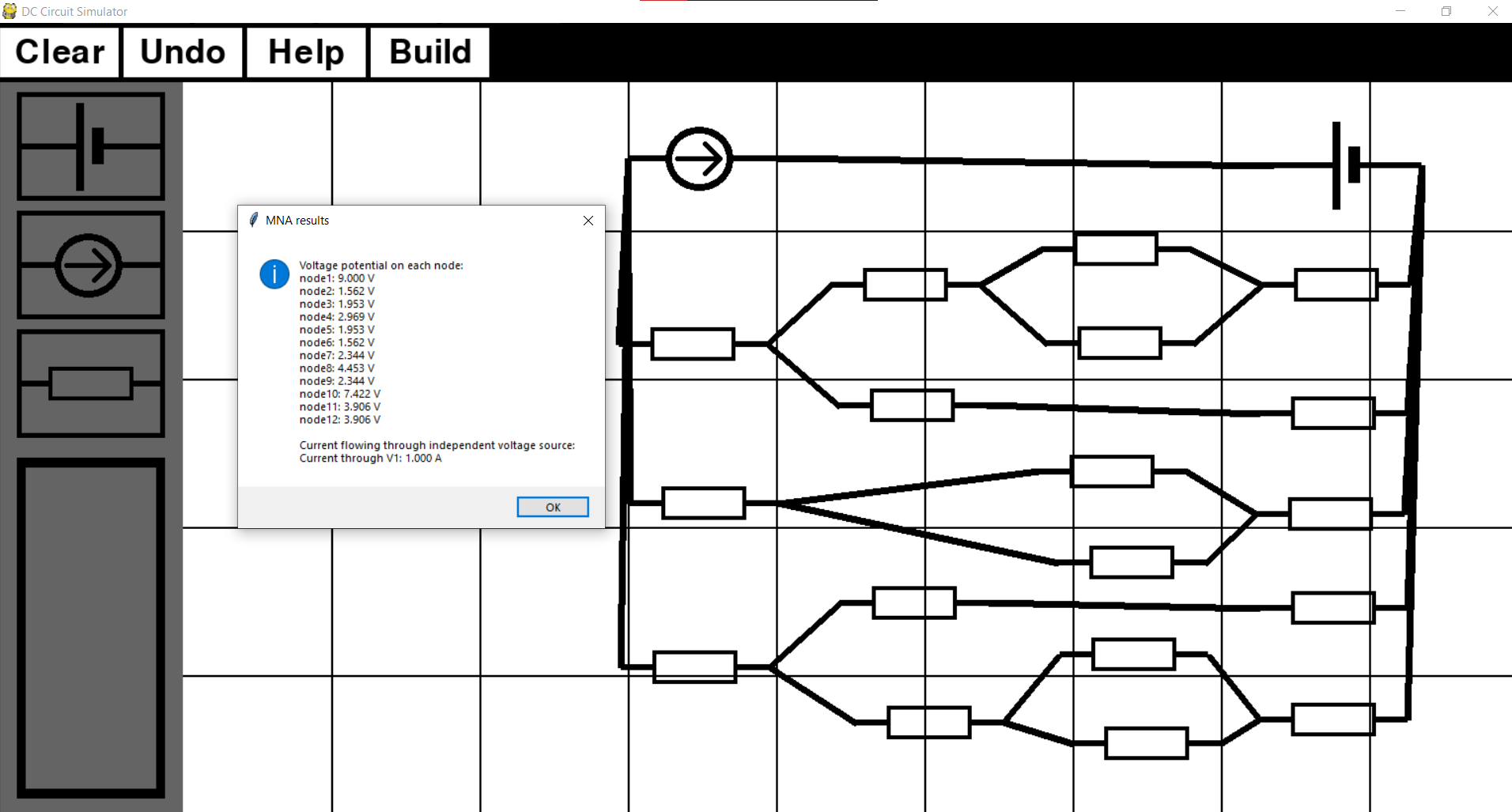


Screenshot with the new code: 

Screenshot evidence for test 7: 

Screenshot evidence for test 8: 

Screenshot evidence for test 9:



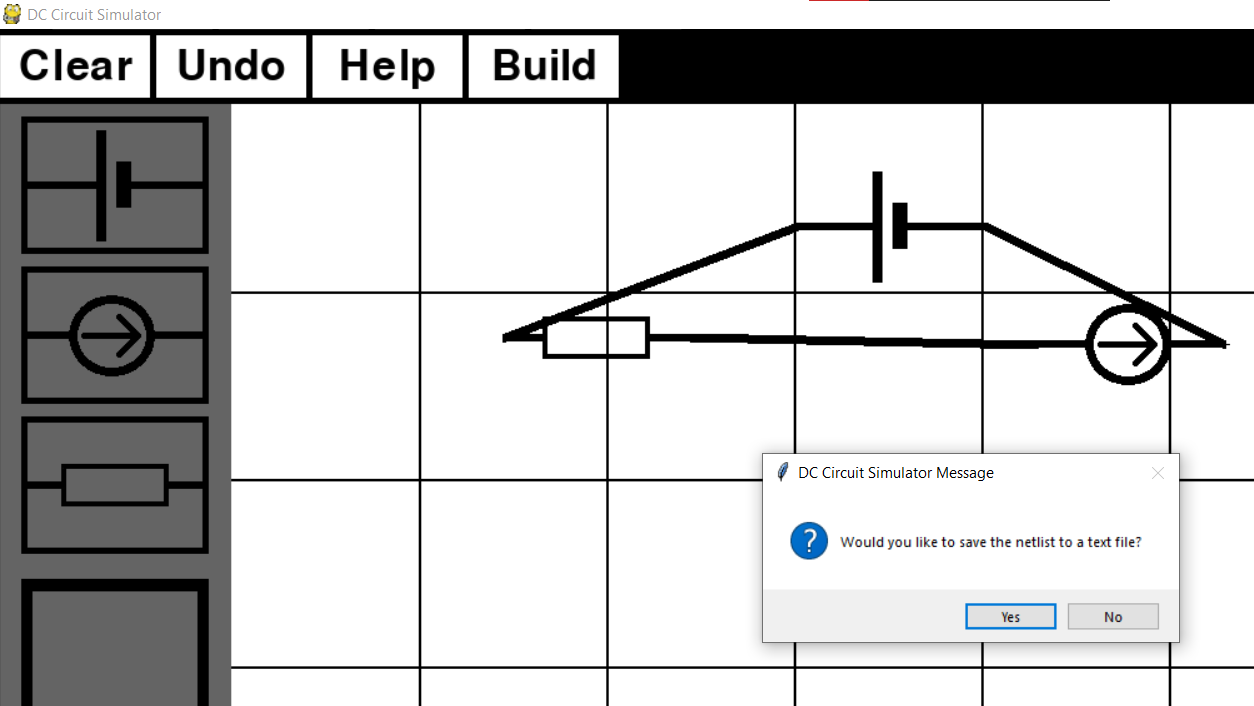
## Objective 5

1. *User will have an option to save the circuit once it was analysed. After the circuit’s netlist was created and stored in the external file, user will be asked if they would like to save the circuit. If the user chooses not to save it, the file containing the circuit’s information will be deleted after appropriate calculations have been made (MNA) and final results have been displayed to the user. If the user decides to save it, netlist information will be saved.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Objective number being tested** | **Test description** | **User input** | **Expected outcome** | **Actual outcome** | **Comment on the outcome**  **(Pass/Fail)** |
| 1 | 5. | User needs to be asked if they would like to save the netlist file | No | File does not get saved | File was not saved | No saved file.  **Pass** |
| 2 | 5. | When user decides to save a file, file should be saved and they should be informed about the name of the file. | Yes | File gets unique title and gets saved | File was saved as ‘dc\_circuit1.txt’ | File saved.  **Pass** |

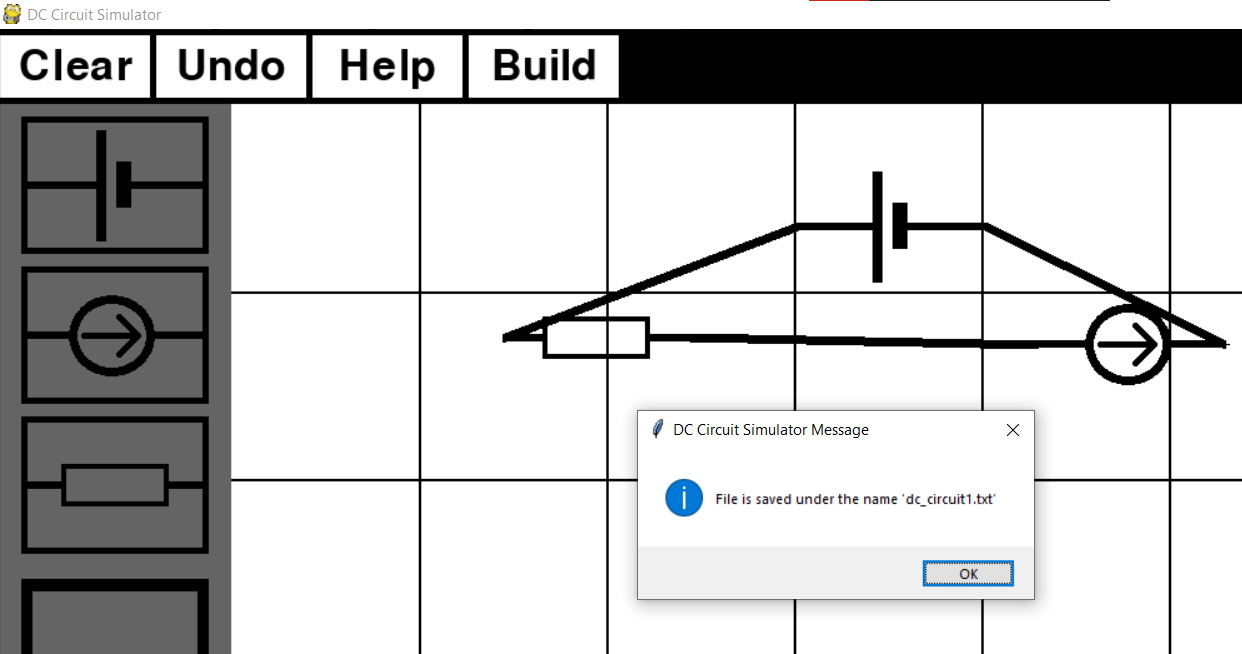
Screenshot evidence for test 1:

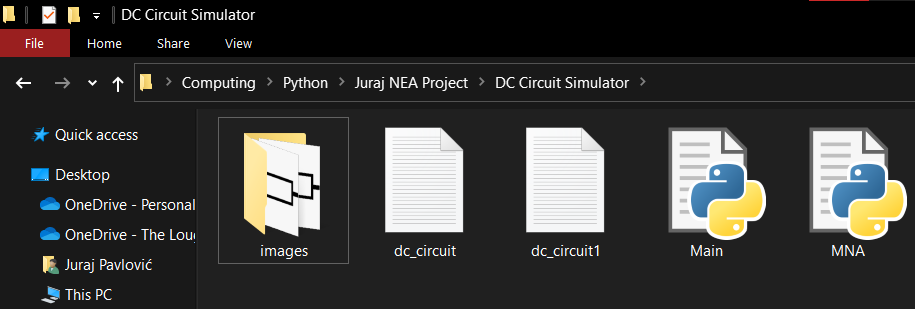
User was asked if they would like to save a file.



They answered ‘No’ so file didn’t get saved.

Screenshot evidence for test 2:

This time user answered ‘Yes’ so they were informed about file’s name. 

Screenshot of the saved file: 

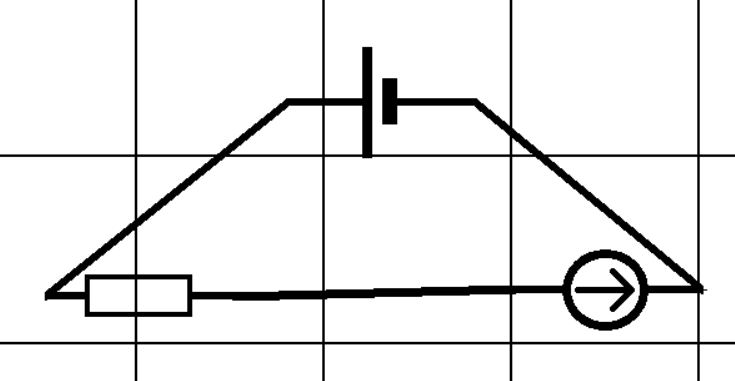
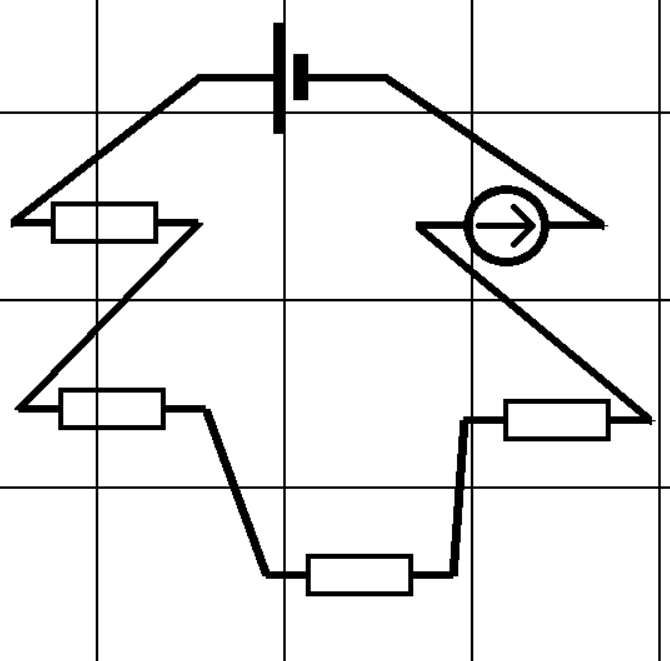
## Objective 6

1. *During MNA, once the matrices are constructed, they need to be stored using the sparse matrix optimisation techniques. This needs to reduce the overall time needed to perform all the calculations. To test it, I will be timing the program’s time to calculate the circuit outputs with and without the use of spare matrices.*

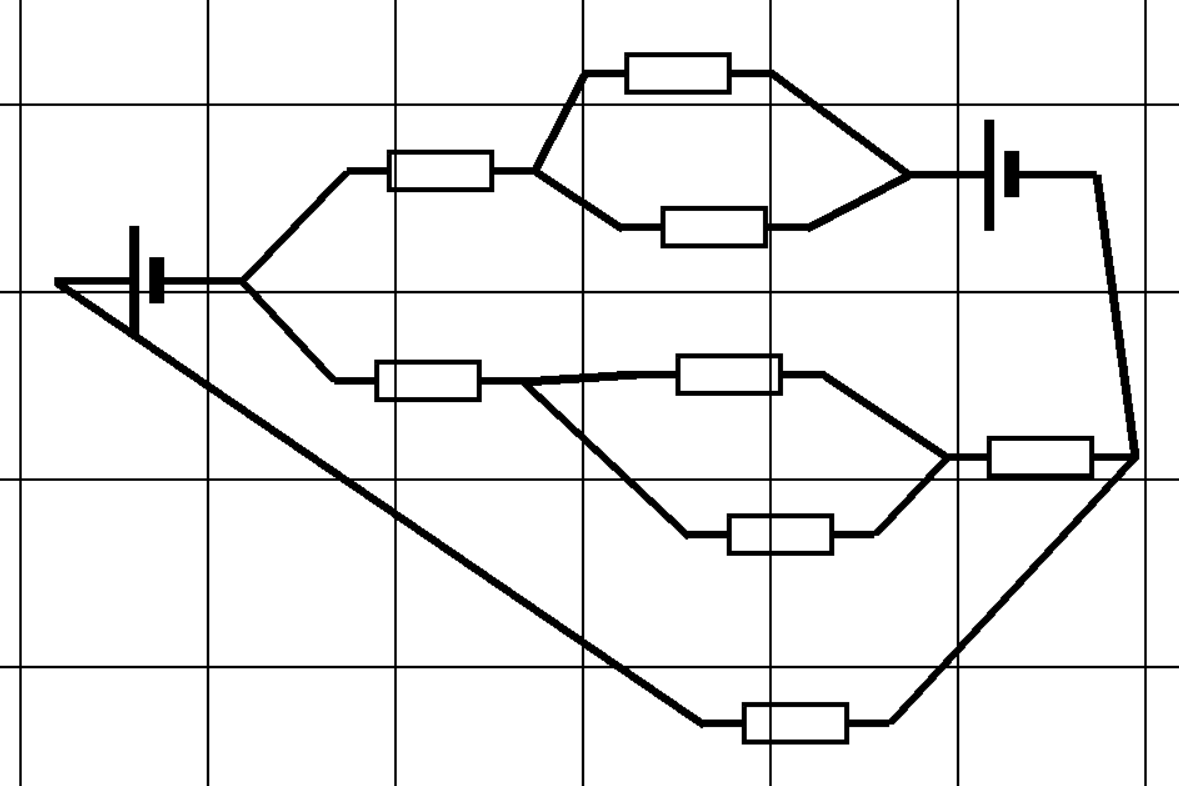
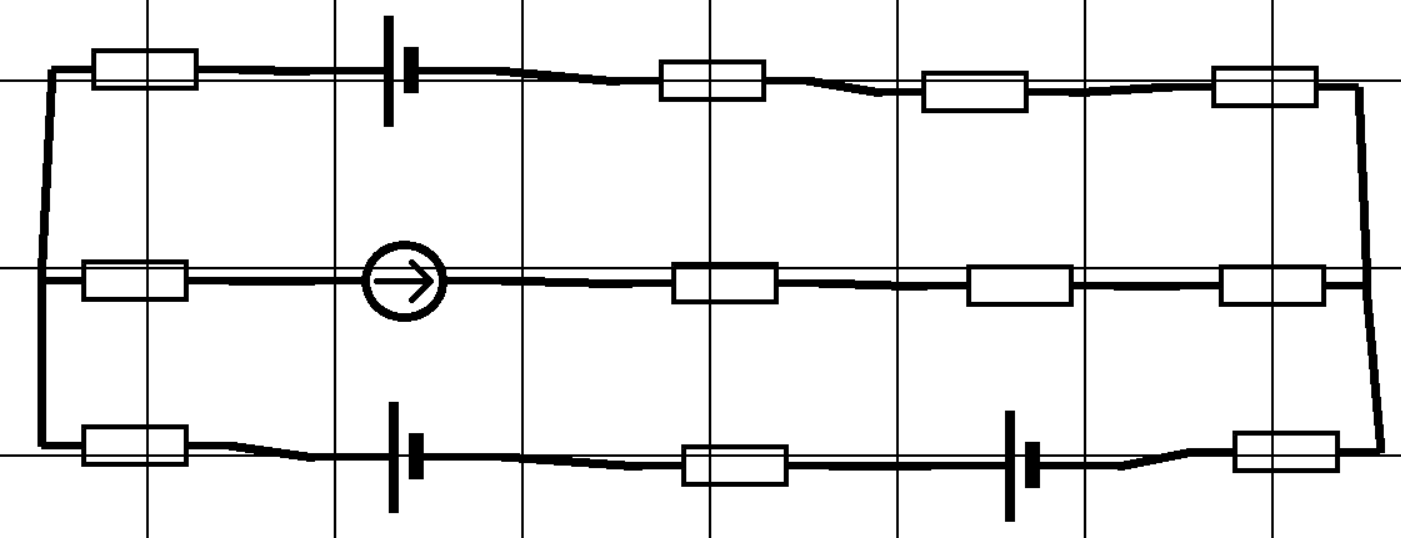
To check if using the sparse matrix optimalisation actually saves time when solving a circuit, I will create five different circuit and solve them first with the sparse matrix optimalisation and then without out. Both times, time the program took to finish will be measured.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test No.** | **Objective number being tested** | **Test description** | **User input** | **Outcome** |
| 1 | 6. | Time taken to solve an optimised circuit with 3 components. | User creates a circuit and presses build button |  |
| 2 | 6. | Time taken to solve a normal circuit with 3 components. | User creates a circuit and presses build button |  |
| 3 | 6. | Time taken to solve an optimised circuit with 6 components. | User creates a circuit and presses build button |  |
| 4 | 6. | Time taken to solve a normal circuit with 6 components. | User creates a circuit and presses build button |  |
| 5 | 6. | Time taken to solve an optimised circuit with 10 components. | User creates a circuit and presses build button |  |
| 6 | 6. | Time taken to solve a normal circuit with 10 components. | User creates a circuit and presses build button |  |
| 7 | 6. | Time taken to solve an optimised circuit with 15 components. | User creates a circuit and presses build button |  |
| 8 | 6. | Time taken to solve a normal circuit with 15 components. | User creates a circuit and presses build button |  |
| 9 | 6. | Time taken to solve an optimised circuit with 25 components. | User creates a circuit and presses build button |  |
| 10 | 6. | Time taken to solve a normal circuit with 25 components. | User creates a circuit and presses build button |  |

Circuit 1: Circuit 2:

Circuit 3: Circuit 4:

Circuit 5

