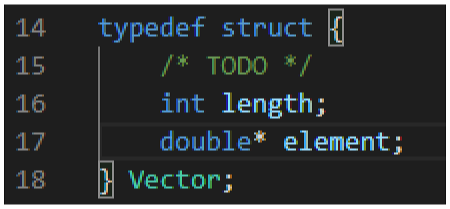
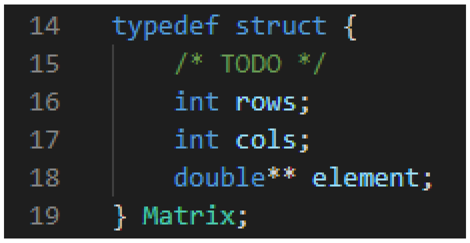
02/11/20
Data Structures & Dynamic Memory Allocation

A struct{} is a collection of variables.
A enum{} assigns names to a set of integers.


```
typedef struct {
    /* TODO */
    int length;
    double* element;
} Vector;
```

```
typedef struct {
    /* TODO */
    int rows;
    int cols;
    double** element;
} Matrix;
```

malloc() requests memory from the operating system and returns it to the program.
calloc() requests memory, assigns zeros to all the positions and returns it to the program.
free() gives back memory to the operating system that had been dynamically assigned to the program.

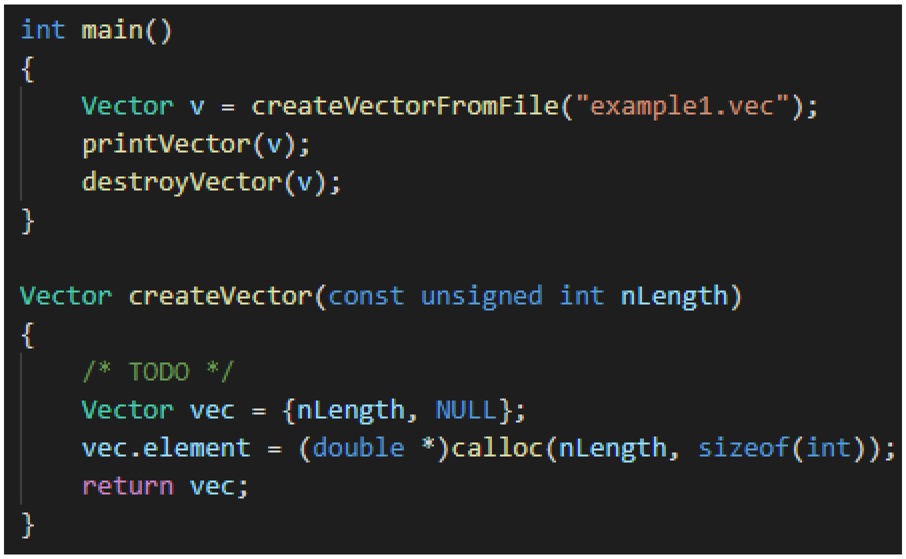
Using the tool make (from MinGW) and a makefile with several terminal commands, you can compile a project from multiple source files.

int main(int argc, char* argv[])

argc is the number of strings on the command line (the first string is always the program name).
argv is a vector of pointers to character strings (argv[0] is the address of the string containing the program's name).

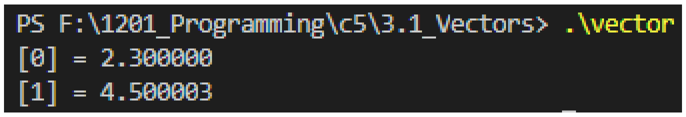
04/11/20

Having defined the data structure for a vector in <vector.h> for my preparation I created a main function to call the other functions. I also wrote the createVector() function that dynamically assigns memory for the vector. After this I ran the code and it read the vectors from the example file.



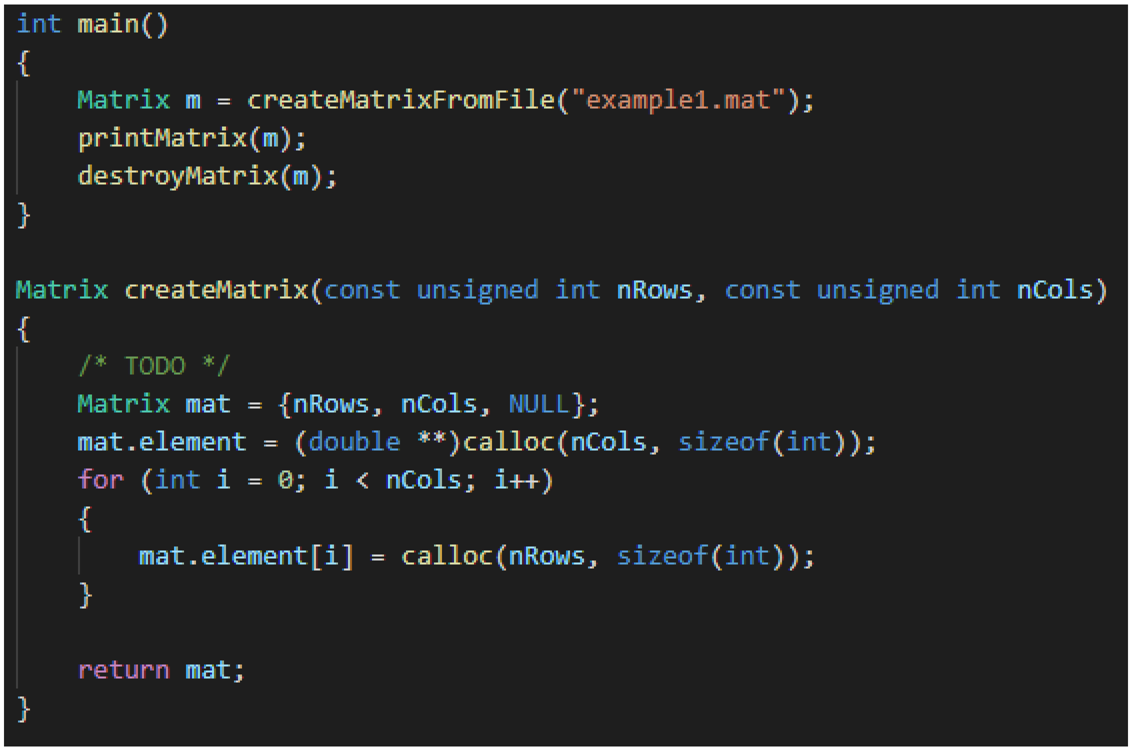
```c
int main()
{
    Vector v = createVectorFromFile("example1.vec");
    printVector(v);
    destroyVector(v);
}

Vector createVector(const unsigned int nLength)
{
    /* TODO */
    Vector vec = {nLength, NULL};
    vec.element = (double *)calloc(nLength, sizeof(int));
    return vec;
}
```



```
PS F:\1201_Programming\c5\3.1_Vectors> .\vector
[0] = 2.300000
[1] = 4.500003
```

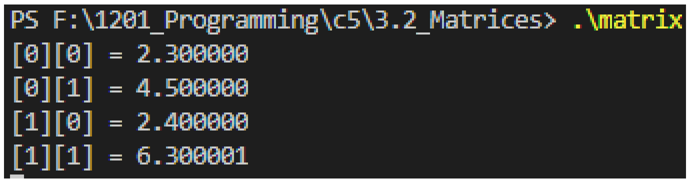
Having defined the data structure for a matrix in <matrix.h> for my preparation I created a main function to call the other functions. I also wrote the createMatrix() function that dynamically assigns memory for the matrix. After this I ran the code and it read the matrices from the example file.



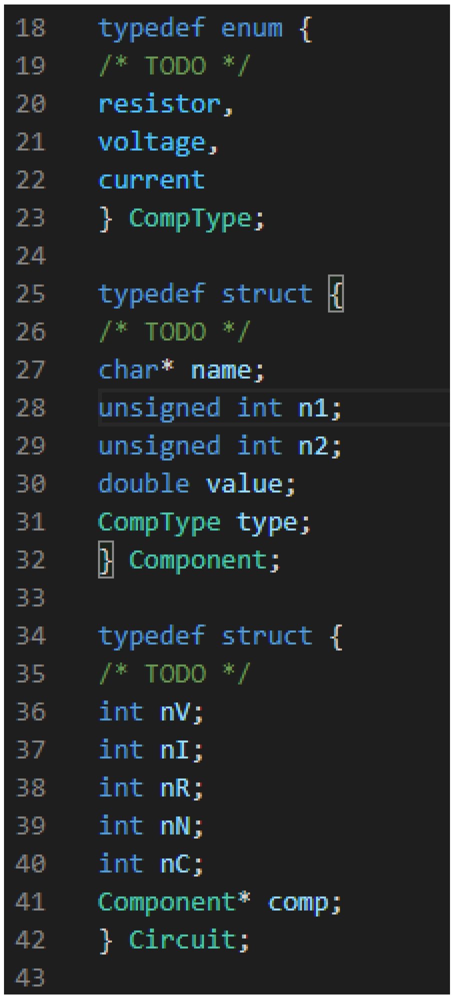
```c
int main()
{
    Matrix m = createMatrixFromFile("example1.mat");
    printMatrix(m);
    destroyMatrix(m);
}

Matrix createMatrix(const unsigned int nRows, const unsigned int nCols)
{
    /* TODO */
    Matrix mat = {nRows, nCols, NULL};
    mat.element = (double **)calloc(nCols, sizeof(int));
    for (int i = 0; i < nCols; i++)
    {
        mat.element[i] = calloc(nRows, sizeof(int));
    }

    return mat;
}
```


```
PS F:\1201_Programming\c5\3.2_Matrices> .\matrix
[0][0] = 2.300000
[0][1] = 4.500000
[1][0] = 2.400000
[1][1] = 6.300001
```

In <circuit.h> I created the structure definitions for all of my data types. I was able to ascertain what each structure needed by looking at what occurred in the provided code of circuit.c.

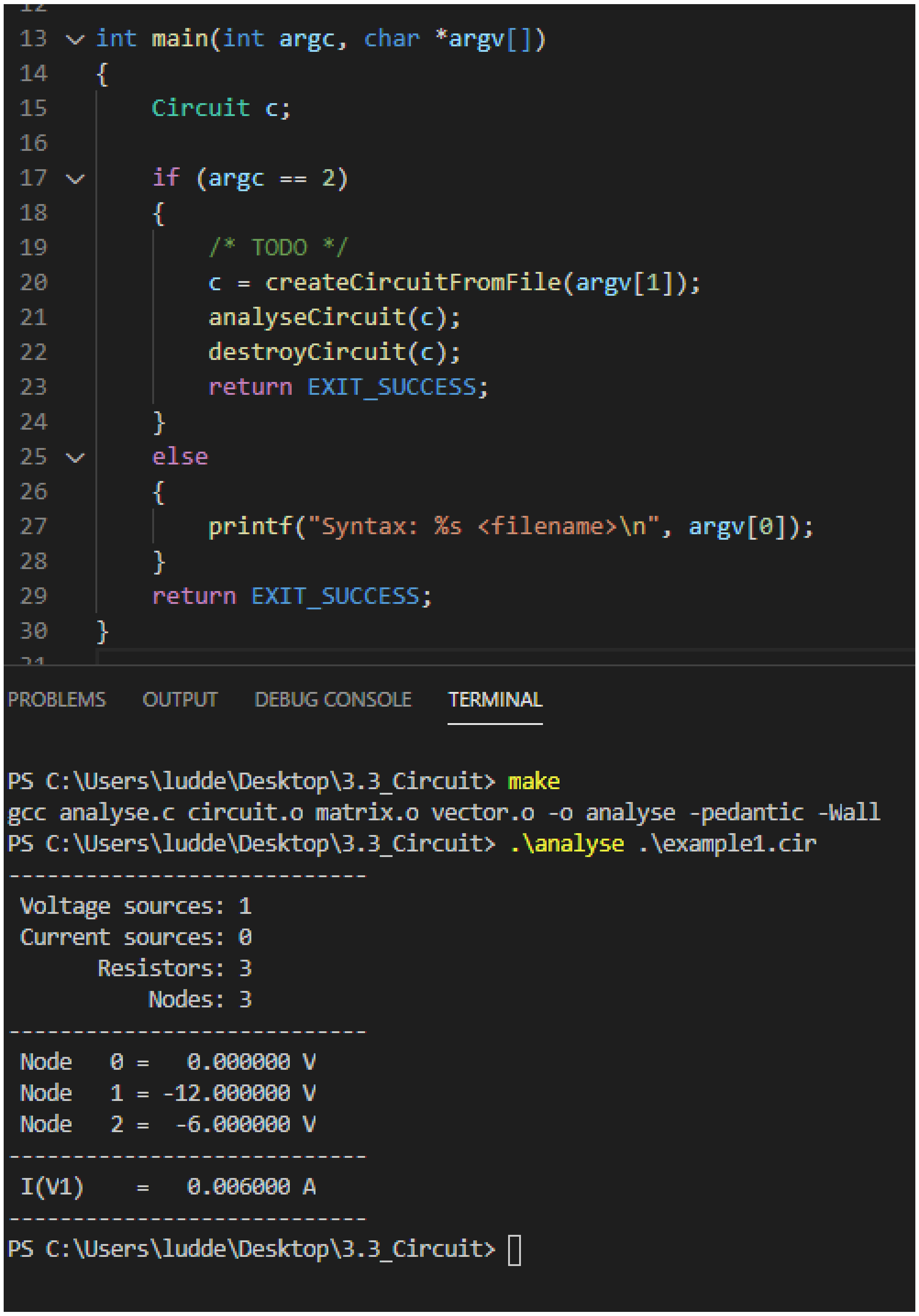

```c
typedef enum {
/* TODO */
resistor,
voltage,
current
} CompType;

typedef struct {
/* TODO */
char* name;
unsigned int n1;
unsigned int n2;
double value;
CompType type;
} Component;

typedef struct {
/* TODO */
int nV;
int nI;
int nR;
int nN;
int nC;
Component* comp;
} Circuit;

```

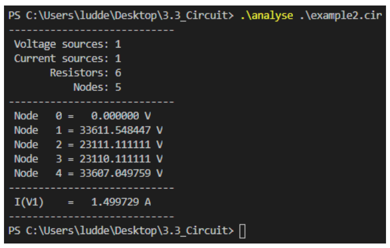
I then used the makefile to compile all my files together so that they would run as one program. However, when running ana



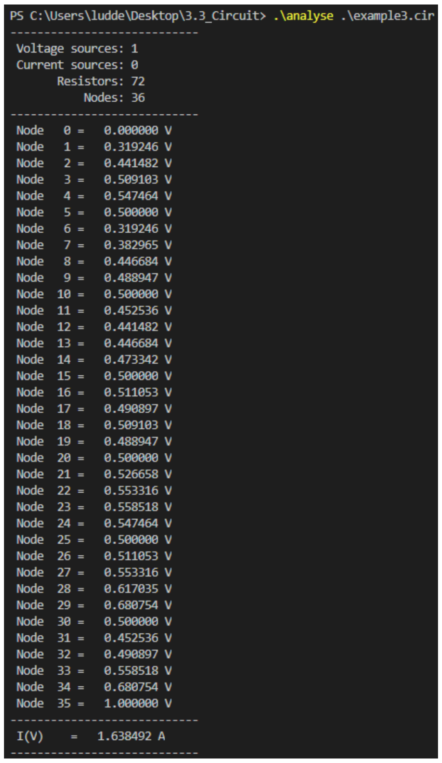
```c
int main(int argc, char *argv[])
{
    Circuit c;

    if (argc == 2)
    {
        /* TODO */
        c = createCircuitFromFile(argv[1]);
        analyseCircuit(c);
        destroyCircuit(c);
        return EXIT_SUCCESS;
    }
    else
    {
        printf("Syntax: %s <filename>\n", argv[0]);
    }
    return EXIT_SUCCESS;
}
```

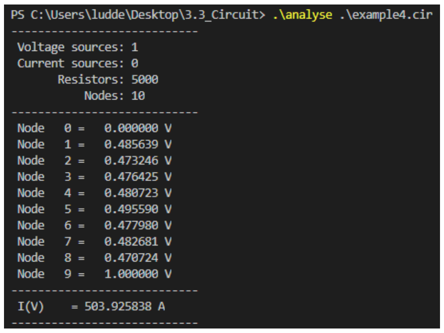
```
PS C:\Users\ludde\Desktop\3.3_Circuit> make
gcc analyse.c circuit.o matrix.o vector.o -o analyse -pedantic -Wall
PS C:\Users\ludde\Desktop\3.3_Circuit> .\analyse .\example1.cir
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 3
           Nodes: 3
----------------------------
 Node   0 =   0.000000 V
 Node   1 = -12.000000 V
 Node   2 =  -6.000000 V
----------------------------
 I(V1)    =   0.006000 A
----------------------------
PS C:\Users\ludde\Desktop\3.3_Circuit> 
```



```
PS C:\Users\ludde\Desktop\3.3_Circuit> .\analyse .\example2.cir
----------------------------
 Voltage sources: 1
 Current sources: 1
       Resistors: 6
           Nodes: 5
----------------------------
 Node   0 =   0.000000 V
 Node   1 = 33611.548447 V
 Node   2 = 23111.111111 V
 Node   3 = 23110.111111 V
 Node   4 = 33607.049759 V
----------------------------
 I(V1)    =   1.499729 A
----------------------------
PS C:\Users\ludde\Desktop\3.3_Circuit> 
```

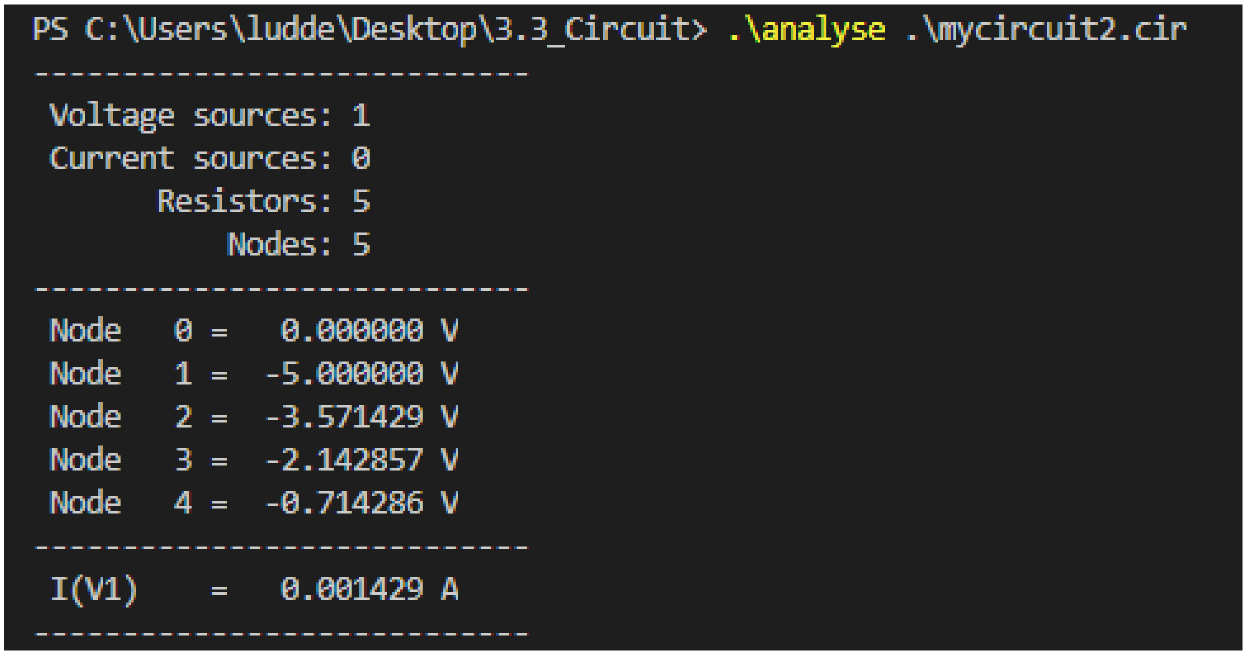


```
PS C:\Users\ludde\Desktop\3.3_Circuit> .\analyse .\example3.cir
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 72
           Nodes: 36
----------------------------
 Node   0 =   0.000000 V
 Node   1 =   0.319246 V
 Node   2 =   0.441482 V
 Node   3 =   0.509103 V
 Node   4 =   0.547464 V
 Node   5 =   0.500000 V
 Node   6 =   0.319246 V
 Node   7 =   0.382965 V
 Node   8 =   0.446684 V
 Node   9 =   0.488947 V
 Node  10 =   0.500000 V
 Node  11 =   0.452536 V
 Node  12 =   0.441482 V
 Node  13 =   0.446684 V
 Node  14 =   0.473342 V
 Node  15 =   0.500000 V
 Node  16 =   0.511053 V
 Node  17 =   0.490897 V
 Node  18 =   0.509103 V
 Node  19 =   0.488947 V
 Node  20 =   0.500000 V
 Node  21 =   0.526658 V
 Node  22 =   0.553316 V
 Node  23 =   0.558518 V
 Node  24 =   0.547464 V
 Node  25 =   0.500000 V
 Node  26 =   0.511053 V
 Node  27 =   0.553316 V
 Node  28 =   0.617035 V
 Node  29 =   0.680754 V
 Node  30 =   0.500000 V
 Node  31 =   0.452536 V
 Node  32 =   0.490897 V
 Node  33 =   0.558518 V
 Node  34 =   0.680754 V
 Node  35 =   1.000000 V
----------------------------
 I(V)    =   1.638492 A
----------------------------
```



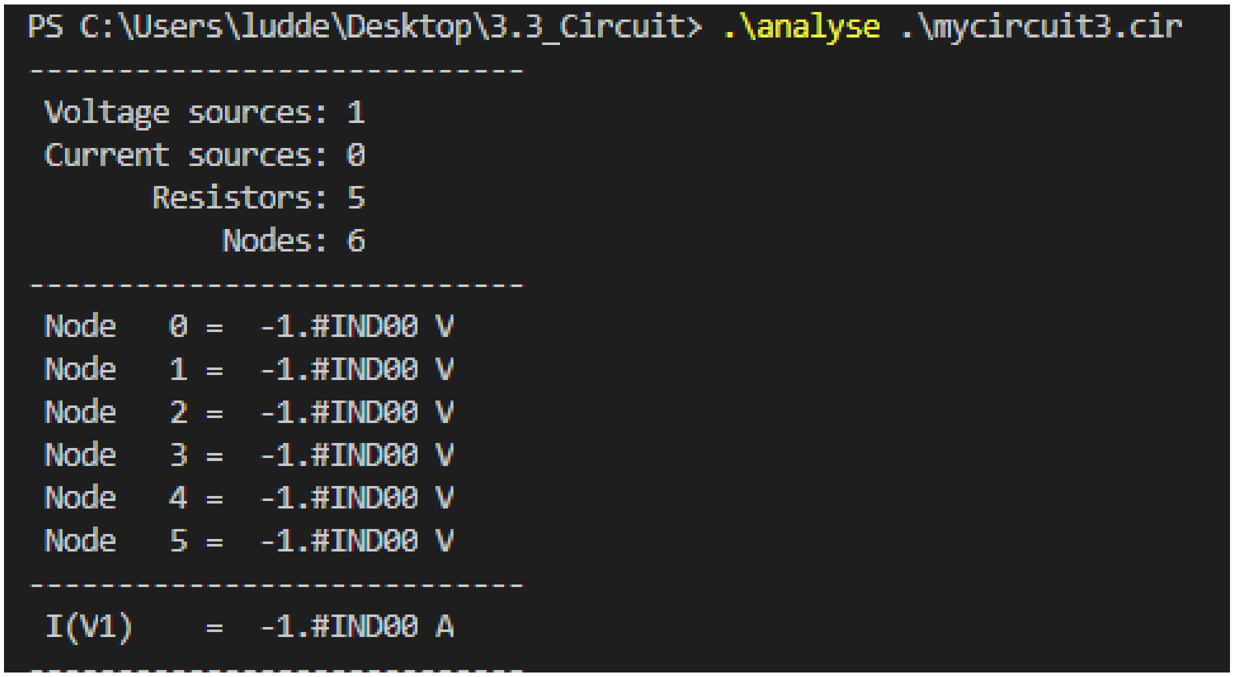
```
PS C:\Users\ludde\Desktop\3.3_Circuit> .\analyse .\example4.cir
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 5000
           Nodes: 10
----------------------------
 Node   0 =   0.000000 V
 Node   1 =   0.485639 V
 Node   2 =   0.473246 V
 Node   3 =   0.476425 V
 Node   4 =   0.480723 V
 Node   5 =   0.495590 V
 Node   6 =   0.477980 V
 Node   7 =   0.482681 V
 Node   8 =   0.470724 V
 Node   9 =   1.000000 V
----------------------------
 I(V)    = 503.925838 A
----------------------------
```

I was able to write a netlist in notepad++ having different voltage sources and resistor values by connecting the nodes together.



```
PS C:\Users\ludde\Desktop\3.3_Circuit> .\analyse .\mycircuit2.cir
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 5
           Nodes: 5
----------------------------
 Node   0 =   0.000000 V
 Node   1 =  -5.000000 V
 Node   2 =  -3.571429 V
 Node   3 =  -2.142857 V
 Node   4 =  -0.714286 V
----------------------------
 I(V1)    =   0.001429 A
----------------------------
```

In my next circuit I was able to break the program by not connecting all the nodes together and causing a divide by zero error in the programme. This stopped it from being able to correctly calculate the voltage over the resistors.



```
PS C:\Users\ludde\Desktop\3.3_Circuit> .\analyse .\mycircuit3.cir
----------------------------
 Voltage sources: 1
 Current sources: 0
       Resistors: 5
           Nodes: 6
----------------------------
 Node   0 =  -1.#IND00 V
 Node   1 =  -1.#IND00 V
 Node   2 =  -1.#IND00 V
 Node   3 =  -1.#IND00 V
 Node   4 =  -1.#IND00 V
 Node   5 =  -1.#IND00 V
----------------------------
 I(V1)    =  -1.#IND00 A
```