Firstly go through the equations and identify if we have any unrestricted variables

and then write down the equations in the form ax+bx+c= 0, the canonical form

Simplex Algorithm:

- Formulate the Objective Function
- Write the Constraints (Inequalities)
-

Input from user:
1. Enter objective func and convert to maximization
2. Enter the number of eqns
3. Enter each eqn in canonical format
4. Select constraint for each eqn
5. formulate the final matrix

Flowchart SIMPLEX steps

Problem (Max/Min) convert to max

convert into standard form

write matrix

check last row for ve coefficients

pivoting

Pivoting: 2,3,4,1

What we need:
1. adding/subtracting the equations for pivoting
2. finding the standard form of eqns
3. finding the smallest variable in the objective func
4. finding the smallest coeff in that variable
5.

Standard form:

what we need:
1. slack variables
2. artificial variables
3. write the general matrix
4.

This a single function that performs Simplex Linear Optimization for LP programming

This example is for maximising an Objective Function. This can be converted into a minimizing problem by multiplying a negative one across the constraint equations

function simplex_optimizer(x1,x2,a1,a2,b1,b2,s1,s2,p1,p2)
x1,x2 -> Coeff of variable in the objective function
a1,a2 -> Coeff of variable x1 in the first and second objective function respectively
b1,b2 -> Coeff of variable x2, x1

Write the objective function in standard form get the inputs Z, x1, x2..

Check the conditions of equations to solve (i.e) Restricted, redundent etc..

Add or subtract the slack variables to bring the conditional inequalities to standard form

Using a method of pivoting eliminate the coefficient with negative value in order to maximize or minimize the objective function

Perform mathematical operations (multiply, divide, subtract, add) solve each set of equation by finding the departing variable

Ask for the number of variables
Ask for the coefficient of the variables
Ask for the inequations
Ask for the right hand of the inequation
Ask for the function to optimise
Ask for maximisation or minimisation
Use :
value = input(text)

OR

split(string, [character to split the string])
To split a string with particular characters (+, -, /, * for example)

Test restriction of variables (artificial variables)

Number of contraints= Number of slack variables

Add slack variables

convert multiple equations into a matrix with an additionnal column which contains numbers or strings corresponding to basic variables

```
index=findLessValue([1 2 4 0])
assert(isequal(index,4),'not equel')

function i=findLessValue(a)
 %return the position (i) of the less value in the array (a)
end
```

```
M=[2,3,4;2,3,4;3,4,5]
array = extractLine(M,2)
assert(isequal(array,[2,3,4]),'not equel')
fucntion array=extractLine(M,index)
%To extract a whole line from a bigger matrix

end
```

Test least values

For each function :

assert(test, message if false)

```
Divide the equation of the current pivot by the coefficient of the pivot variable
The least result is in the pivot line
assert(, 'Index not corresponding to the matrix size')
function newM=divideEquation(M,equationIndex,pivotIndex)

 %M is the matrix which contains all the equations
 %Divide the equation of the current pivot by the coefficient of the pivot variable
 %returns the modified matrix
end
```

pivot with the correct line and correct column

```
function v = maxmiseObjEqn(Objeqn,z)
% Maxmise the objective equation
end
```

```
add or substract corresponding rows
assert(<length(), 'Index not corresponding to the array length')
assert(isequal(length(pivotLine),length(operationLine)),'lengths are not matching ')
function newLine=(pivotLine,pivotIndex,operationLine)
% newLine, pivotLine, operationLine are arrays
% this function chooses to add or substract according to the pivot
%returns the new line after the operation
```

```
test if all entries in the bottom row are zero or positive
flag = isPositive([1 -2 -1 0 2])
assert(~flag)
flag = isPositive([1 2 1 0 2])
assert(flag)
function bool=isPositive(a)
 %returns true if all the values in the
```

```
write the solution according to final matrix and the corresponding basic variables


function result=simplexResult(M)
 %result is an array and M the matrix which contains the equations
 %this function associates variables and to values (1, 2 , ...,xn , 1 ,       S2 ,..., n )
 %returns an array with the solution
end

function optimum=optimumValue(M)
 %M is the matrix which contains all the equations
 %returns the optimum value
end
```

To improve : Ask for the operation between variables (+, -, *, /)
Might be usefull if the problem gets more complex (x1*x2 <= 50 for example)

```
v=mathFun([1 2 3 4]);
assert(isequal(v,[1 3 4]),'not equel')

function v = mathFun(vinput)
%mathFun This function will plus x to y
% Detailed explanation goes here
v=[1 3 4];
end



inputMatrix = [ -10 20 22;...
                5 10 49;...
                1 0 4;...
                1 -4 0]
inequality = [1;1;-1]
```