```
bool Run (circuit c)
{
//start clock = time
G0, NOT, w1, C
G1, AND2, w2, A, w1
500,A, 1
800, B, 1
1300, C, 1
```

```
Arrinputs[4];
Arroperators[3];
Arrinputs[0] = A;
Arroperators[1] = &;
```

1. All gates' inputs are 0
   isChanged() //checks when an input is changed if the output of the logic gate has changed

This is inside a while loop:

2. Traverse usedGates
   a. Check if you have all the inputs from cirInputs in cir_Input_Names **AT THE current time**.
      i. As time progresses you will check more inputs.
      ii. Once you find a working gate APPLY the expression **then add to cirInputs.**
3. You must have at least one working logic gate (initially) that will provide an output.
   a. Add this output as an input to cirInputs with its time, Name, Logic Value
      i. The time is the time of the inputs that made it work + the gate's propagation delay.
4. After each new Addition to cirInputs we must loop again.

Once the loop is over then you should have a final output and the simulation over.

Note: each element you add to cirInputs will cause a change in the waveform for bonus for the sub outputs.

```
}

A:0 B:0,1 C:0,1,0,1
--------------------
A:1 B:0,1 C:0,1,0,1
```

```
}
bool operator(vector<string> exp)
{
vector<bool> output;
Char operator;
for(int i = 0; i<exp.size(); i++)
{
      if(i%2 != 0)
      {
            if(exp[i]==’&’)
                  Operator = &;
                  Output.push_back(exp[i-1] & exp[i+1]);
            Else if (exp[i]==’|’)
            ……

      }

      Return sum;
```

```
1,0,1

Bool Sum = 1; //x*1 = x

for(int i=0;i<size-2;i++){
//all of this is inside the switch case
sum=v[i]&v[i+1]
}

Sum = sum & (1 & 0 )
Sum =  sum & (0 & 1)
```