ITI 1120 Fall 2012 - Assignment 3

Available: Sunday, Oct 27, 2012
Due: Sunday Nov 11, 2012, 11:59 pm (midnight)

Instructions

This assignment is to be done INDIVIDUALLY. Follow the instructions in the lab manual for submitting assignments through the Virtual campus. The following are specific instructions for this assignment:

1) For question 1, provide a Word file A3Q1.doc file with the algorithms developed for the question.
2) For question 2, provide a Word file A3Q2.doc file with the algorithms developed for the question.
3) For question 3, submit the Java files A3Q3.java and Circuit.java. The Java code should also be inserted in the files A3Q3.doc for commenting.
4) Zip all the .doc, .java, and .class files in a3_xxxxxx.zip, where xxxxxx is your student number, and submit it through the Virtual Campus.

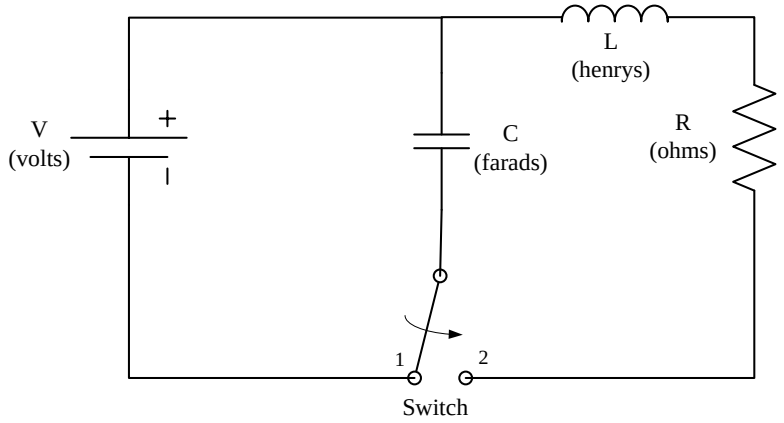
Your algorithms should be developed using the format used in class. In the Java code, use the coding structures presented in class. Do not use other structures, such as the switch and break statements.

Marking Scheme (total 100 marks)

- **Regulations and Standards: 5 marks**
- Question 1: 30 marks
- Question 2: 25 marks
- Question 3: 40 marks

Exploring an R-L-C electric circuit

Consider the following electrical circuit that contains one source and three passive electric circuit elements, the capacitor C, , the inductor L, and the resistor R. The voltage source V (a battery) provides the energy that will charge capacitor when the switch is in position 1 which connects the capacitor to the battery.



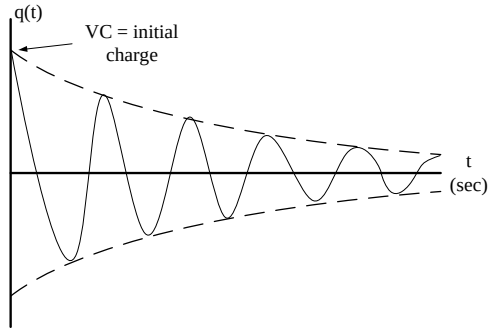
The following table provides reasonable values for each of the circuit components:

Component	Unit	Minimum	Maximum
V	volts	4	15
C	farads	$1 \text{ X } 10^{-9}$	$1 \text{ X } 10^{-7}$
L	henrys	$1 \text{ X } 10^{-3}$	$1 \text{ X } 10^{-1}$
R	ohms	5	10

When the switch is thrown from the voltage source to the resistor (capacitor is fully charged), the charge *q(t)* of the capacitor varies with time as given by the following equation:

$$q(t) = VCe^{-\frac{R}{2L}t} \cos\left\{t\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}\right\}$$

The function *q(t) oscillates and decreases over time* as shown in the following graph.



The objective of this assignment is to develop a program that allows the generation of values for q(t) given values for each of the four components. The following page shows a sample output from the program. Use it as a guideline for developing your software.

Sample Output

```
ITI1120 Winter 2012, Assignment 3, Question 3
Name: Gilbert Arbez, Student# 81069665
Enter a value for V (4 to 15): 10
Enter a value for C (1e-9 to 1e-7): 1e7
Bad value: 1.0E7
Enter a value for C (1e-9 to 1e-7): 1e-7
Enter a value for L (1e-3 to 1e-1): 1
Bad value: 1.0
Enter a value for L (1e-3 to 1e-1): 0.1
Enter a value for R (5 to 10): 50
Bad value: 50.0
Enter a value for R (5 to 10): 10
-----------------------------------------
Voltage Source V = 10.0 volts
Capacitor C = 1.000E-07 farads
Inductor L = 1.000E-01 henrys
Resistor R = 10.0 ohms
-----------------------------------------
Reset component values? n
Run a simulation? y
Enter a maximum time (sec): 1.0
Enter a time step (sec): 0.05
-----------------------------------------
Time t      Q(t)
0.000E000   1.000E-006
5.000E-002   -7.279E-008
1.000E-001    3.859E-009
1.500E-001   -7.128E-011
2.000E-001   -1.562E-011
2.500E-001    2.755E-012
3.000E-001   -2.957E-013
3.500E-001    2.449E-014
4.000E-001   -1.573E-015
4.500E-001    6.397E-017
5.000E-001    1.287E-018
5.500E-001   -6.183E-019
6.000E-001    8.135E-020
6.500E-001   -7.676E-021
7.000E-001    5.694E-022
7.500E-001   -3.117E-023
8.000E-001    7.007E-025
8.500E-001    1.080E-025
9.000E-001   -2.044E-026
9.500E-001    2.248E-027
1.000E000   -1.896E-028
-----------------------------------------
Run a simulation? n
Do you want to quit (y/n)? y
Program Terminated
```

Question 1 – <u>Design of the User Interface</u>

Appendix A provides two algorithms for interacting with the user; the main algorithm and the algorithm *getCircuitComp*. The main algorithm interacts with the user as follows.

- Request from the user the circuit component values by calling the algorithm *getCircuitComp*. This is done at the very start of the program and within the outer loop of the algorithm.
- The outer loop of the algorithm allows the user to repeat two steps. The first is to prompt the user to reset the values of the components. To reset the component values, the algorithm *getCircuitComp* is called. The second step is to simulate the operation of the circuit as described in the next point.
- An inner loop is used to simulate the operation of the circuit. To perform this simulation, the user is asked to provide a time length and a time step for simulation. These values are used to generate a time array using the algorithm *genTimeArr*, which is then passed to the *genQArray* algorithm to create and fill a q value array. Results can then be displayed using the *displayQFunction*. These called algorithms are developed in Question 2. The inner loop prompts the user to run another simulation and thus allows the user to run a number of simulations for different time values before resetting the component values.

Note also that the values of the components (battery, capacitor, inductor, and resistor) are all stored in an array. The four constants VIX, CIX, LIX, and RIX are defined globally so that all algorithms can use these constants as indices into the component value array.

The *getCircuitComp* algorithm is used to prompt the user for values of each component in the circuit. You will notice that for each such value, there is a pattern in the steps taken to interact with the user:

- Setup a loop that monitors the value of flag (a Boolean variable)
- Prompt the user for a value (and store in the array)
- Check the value provided by the user
- If the value is valid (within expected range), set flag to false (to break out of loop)
- If the value is not valid, print an error message.

Note that these algorithms are rather complex. It should then be possible to identify tasks in each algorithm that can be captured in other algorithms. Then the original algorithms can call these new algorithms and consequently the original algorithms will become simpler.

a) Expand the main algorithm into the following algorithms:
   i. An algorithm that prompts the user to reset the component values. This algorithm is given the reference to the existing component array. First the current values are displayed (see the example output). If the user chooses to reset the component values, a reference to a new array (created with a call to the *getCircuitComp* algorithm) is returned, otherwise the reference to the original array is returned.
   ii. An algorithm that allows the user to run one or more simulations.
   iii. Update the *main* algorithm to make calls to the above new algorithms.

b) Expand the *getCircuitComp()* algorithm into the following algorithms.
   i. Define an algorithm that receives as givens a prompt string, a minimum value and a maximum value. The algorithm prompts the user until it obtains a valid value. The result of the algorithm is the valid value read in from the user.
   ii. Revise the algorithm *getCircuitComp* to make the necessary calls the algorithm developed above.

Question 2 – Design of the Circuit Software

The circuit software consists of an algorithms for simulating the operation of the circuit.

a) Develop an algorithm, *computeQ*, that returns the value for the function q(t) for a given time and given component values provided in the component values array.
b) Develop an algorithm, *genTimeArray*, that generates an array of time values given a maximum length of time (*tMax*) and a time step (*tStep*). Thus the array will contain the values t_0, t_1, t_2, …, t_{max}, where $t_0 = 0$, and $t_i = t_{i-1}$+tStep for i=1, 2, 3, ….
c) Develop an algorithm, *genQArray*, that calculates the values of q(t) for a given array of time values (see b) and a given array of circuit component values . The algorithm shall save the values of the charge q(t) in an array; the reference to the array is the results of the algorithm. Use the algorithm from part a above to compute the values of q.
d) Develop an algorithm, *displayQFunction*, that displays the contents of time and the q value arrays. See the example output for the format.

Question 3 - Implementation and Testing

a) Translate the algorithms developed in question 1 to Java methods and place them into the class A3Q3, stored in the file A3Q3.java.
b) Translate the algorithms developed in question 2 to Java methods and store them in the class Circuit, stored in the file Circuit.java. Define the constants VIX, CIX, LIX and RIX in this class outside the methods using the following syntax (shown for VIX):

```
public static final int VIX = 0;  // Index to the battery value in
                                  // the components value array
```

The keyword *final* makes the variable *VIX* a constant with the value 0. These constants will be available to all methods in the *Circuit* class. As well, the constants can be accessed from outside the class using the expressions like *Circuit.VIX* (this is similar to the constant *Math.PI* available in the *Math* class).

Appendix A - Algorithms for Question 1

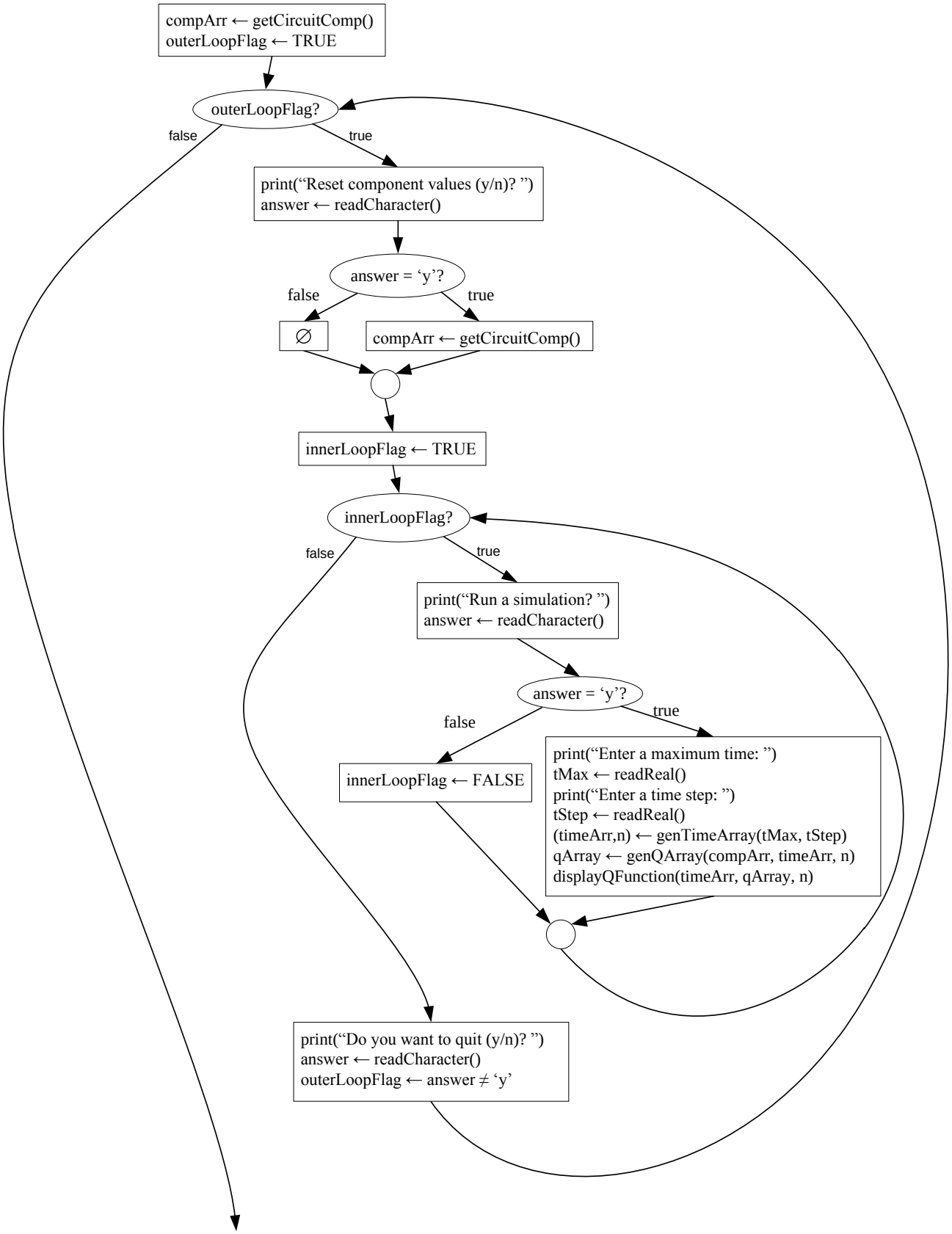
Global constants, these constants are accessible to all algorithms

VIX	*(constant 0 – index for voltage value in component value array)*
CIX	*(constant 1 – index of capacitor value in component value array)*
LIX	*(constant 2 – index of inductance value in component value array)*
RIX	*(constant 3 – index of resistance value in component value array)*

GIVENS: *(none)*
RESULTS: *(none)*
INTERMEDIATES:

compArr	*(reference to component value array)*
outerLoopFlag	*(flag to control outer loop)*
innerLoopFlag	*(flag to control inner loop)*
answer	*(character to record y/n answers from user)*
tMax	*(variable for maximum time of simulation)*
tStep	*(time step for simulation)*
timeArr	*(reference to time array)*
qArray	*(reference to q value array)*
n	*(number of elements in the arrays referenced by timeArr and qArray)*

HEADER:
main()
BODY: (see next page)


compArr ← getCircuitComp()
outerLoopFlag ← TRUE
outerLoopFlag?
false
true
print("Reset component values (y/n)? ")
answer ← readCharacter()
answer = 'y'?
false
true
∅
compArr ← getCircuitComp()
innerLoopFlag ← TRUE
innerLoopFlag?
false
true
print("Run a simulation? ")
answer ← readCharacter()
answer = 'y'?
false
true
innerLoopFlag ← FALSE
print("Enter a maximum time: ")
tMax ← readReal()
print("Enter a time step: ")
tStep ← readReal()
(timeArr,n) ← genTimeArray(tMax, tStep)
qArray ← genQArray(compArr, timeArr, n)
displayQFunction(timeArr, qArray, n)
print("Do you want to quit (y/n)? ")
answer ← readCharacter()
outerLoopFlag ← answer ≠ 'y'

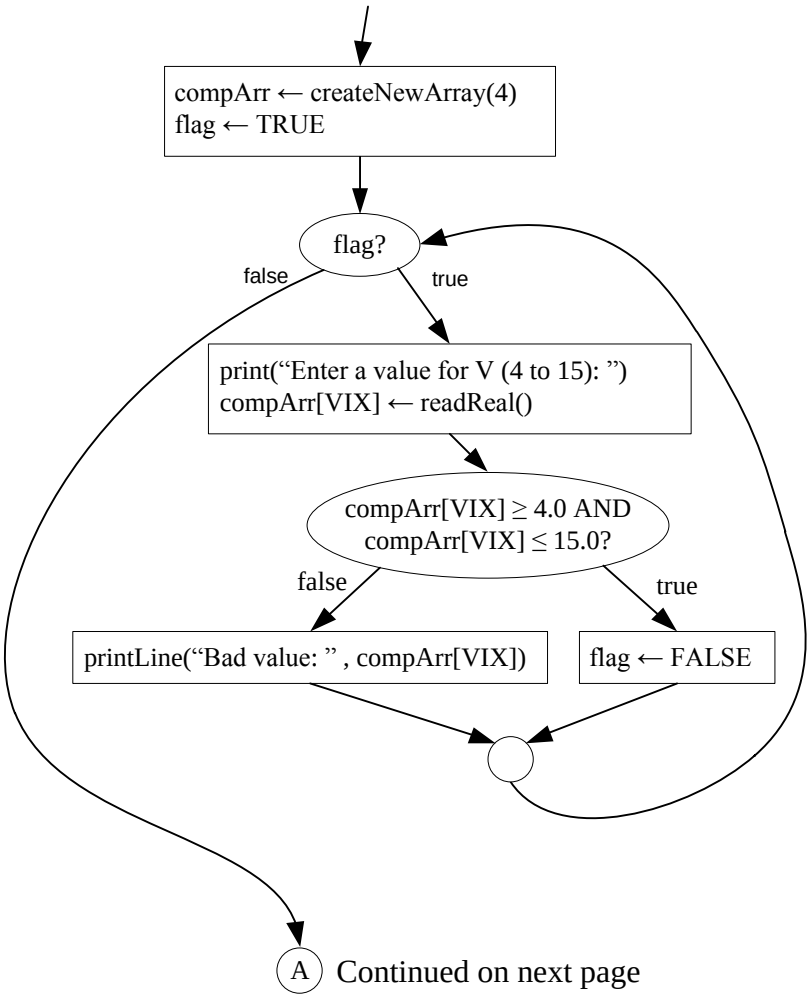
GIVENS: *(none)*
RESULTS:
compArr *(Reference to component array with values)*

INTERMEDIATES:
flag *(flag to monitor when user has entered a valid value)*

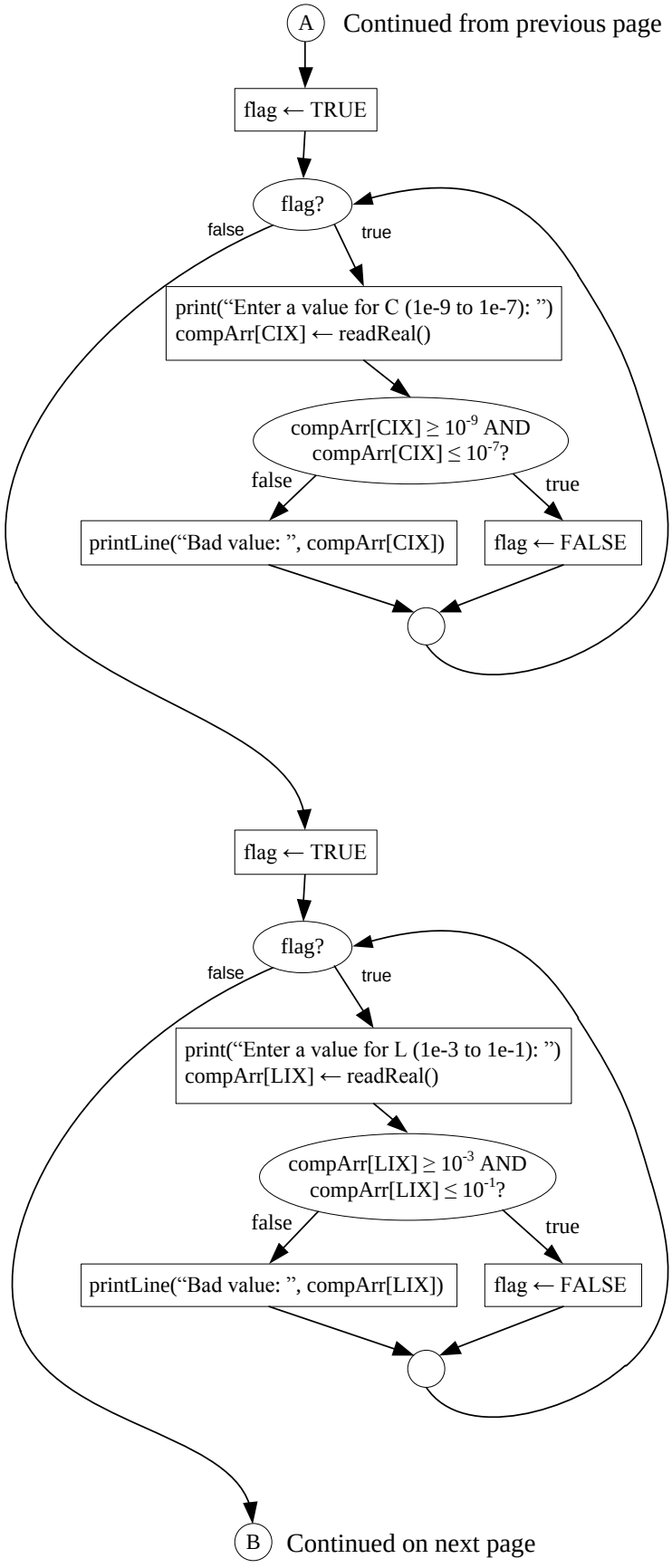
CONSTRAINTS:
compArr[VIX] has values ranging between 4 and 15 volts
compArr[LIX] has values ranging between 10^{-3} and 10^{-1} henrys
compArr[CIX] has values ranging between 10^{-9} and 10^{-7} farads
compArr]RIX] has values ranging between 5 and 10 ohms

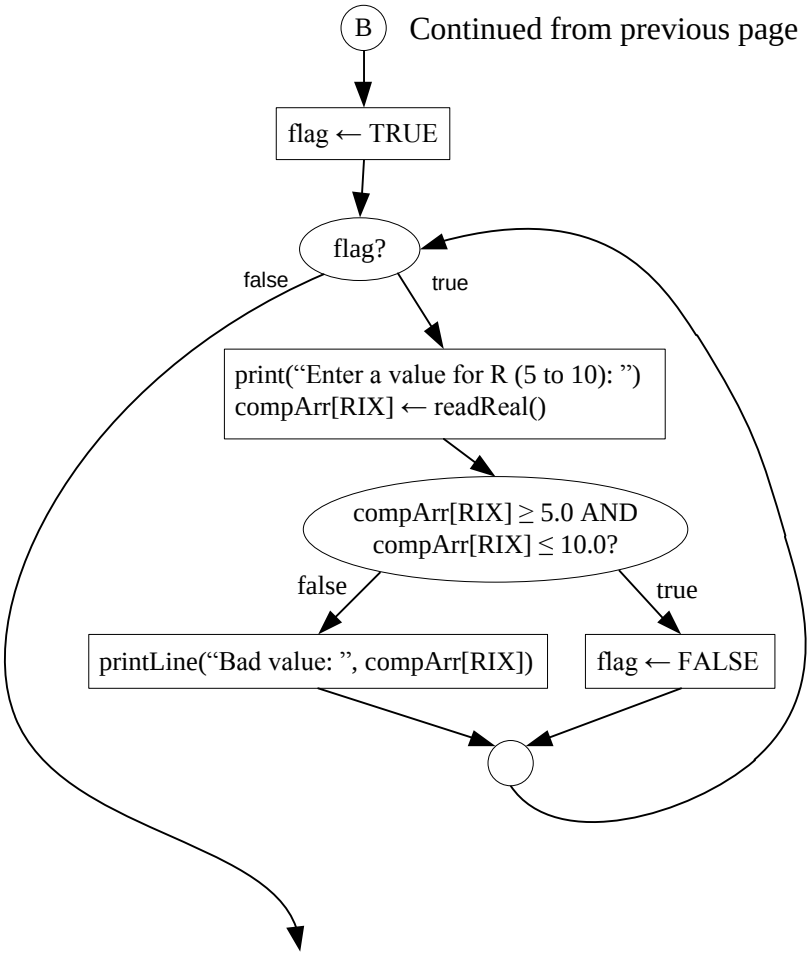
HEADER:
compArr ← getCircuitComp()

BODY:



Continued on next page


A
Continued from previous page
flag ← TRUE
flag?
false
true
print("Enter a value for C (1e-9 to 1e-7): ")
compArr[CIX] ← readReal()
compArr[CIX] ≥ 10^-9 AND
compArr[CIX] ≤ 10^-7?
false
true
printLine("Bad value: ", compArr[CIX])
flag ← FALSE
flag ← TRUE
flag?
false
true
print("Enter a value for L (1e-3 to 1e-1): ")
compArr[LIX] ← readReal()
compArr[LIX] ≥ 10^-3 AND
compArr[LIX] ≤ 10^-1?
false
true
printLine("Bad value: ", compArr[LIX])
flag ← FALSE
B
Continued on next page


B
Continued from previous page
flag ← TRUE
flag?
false
true
print(“Enter a value for R (5 to 10): ”)
compArr[RIX] ← readReal()
compArr[RIX] ≥ 5.0 AND
compArr[RIX] ≤ 10.0?
false
true
printLine(“Bad value: ”, compArr[RIX])
flag ← FALSE