CSc 106: The Practice of Computer Science

Algorithms

What is a computer?

A **computer** is a “device” that …

- Accepts input
- Processes data
- Stores data
- Produces output

- A computer executes machine instructions.

What is a computer?

A **computer system** is made up of:

- Hardware:
  - Pre-defined functionality, hard-wired logic, difficult to change once designed.
- Software:
  - Controlling hardware
  - Logic and functionality easy to change
- Peripherals

What's an Algorithm?

- A finite set of unambiguous instructions performed in a prescribed sequence to achieve a goal, especially a mathematical rule or procedure used to compute a desired result. Algorithms are the basis for most computer programming.
  - The American Heritage® Science Dictionary
- An algorithm is a specification of a behavioral process. It consists of a finite set of instructions that govern behavior step-by-step.
  - Shackelford, Russell L. in *Introduction to Computing and Algorithms*

Baking cookies

- How does one bake sugar cookies?
  (what is the "bake sugar cookies" algorithm?)
  - Mix the dry ingredients.
  - Cream the butter and sugar.
  - Beat in the eggs.
  - Stir in the dry ingredients.
  - Set the oven for the appropriate temperature.
  - Set the timer.
  - Place the cookies into the oven.
  - Allow the cookies to bake.
  - Mix the ingredients for the frosting.
  - Spread frosting and sprinkles onto the cookies.
  - ...

Characteristics of a Good Algorithm

- Definiteness
  - specifies the sequence of events and the details of each step
  - Unambiguous
- Finiteness
  - Must stop
- Input
- Output
- Effectiveness
  - Doable with expected output

Challenges with recipes?

- Precision of terms (need a well defined language)
- Expressiveness of terms (need abstractions)
- An art or a science?

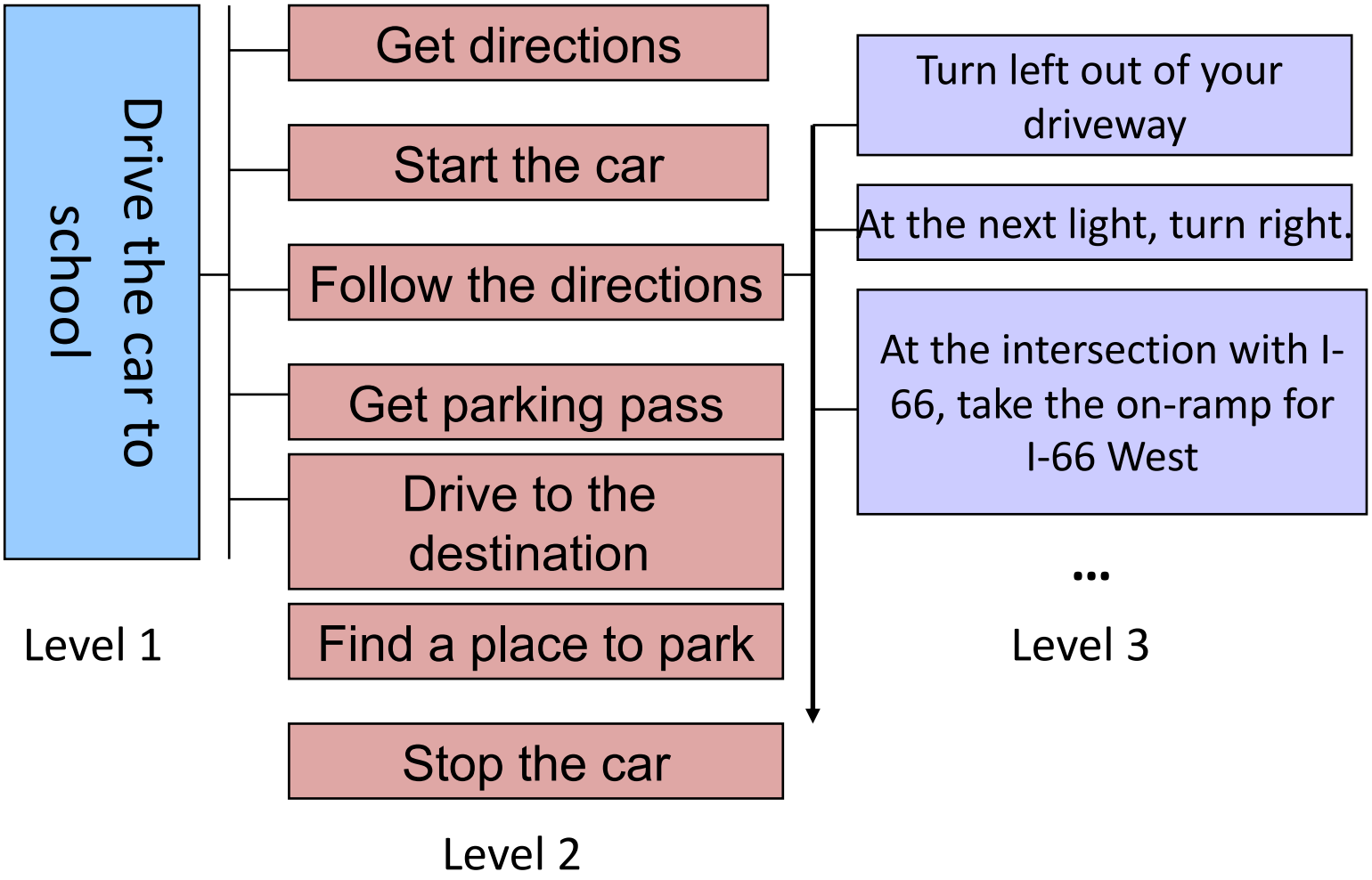
Algorithm Creation

- analyze the problem
  - identify the input(s)
  - identify the output(s)
  - articulate the purpose of the algorithm in terms of the input(s) and output(s)
  - articulate any assumptions
- work through example(s) of the problem
- design/implement an algorithm solution
- test your solution

Top down design

- Instead of approaching a problem and worrying about each and every thing you must do to solve the problem, you can begin to look at the major steps. After the major steps, you can begin to fill in how you would accomplish the major step.
- That fill in may lead to the need for additional levels to fill in those details, etc.

Diagrammatically



Refining our informal definition of an Algorithm

- Every algorithm has an ***input*** (or ***problem instance***) and an ***output***.

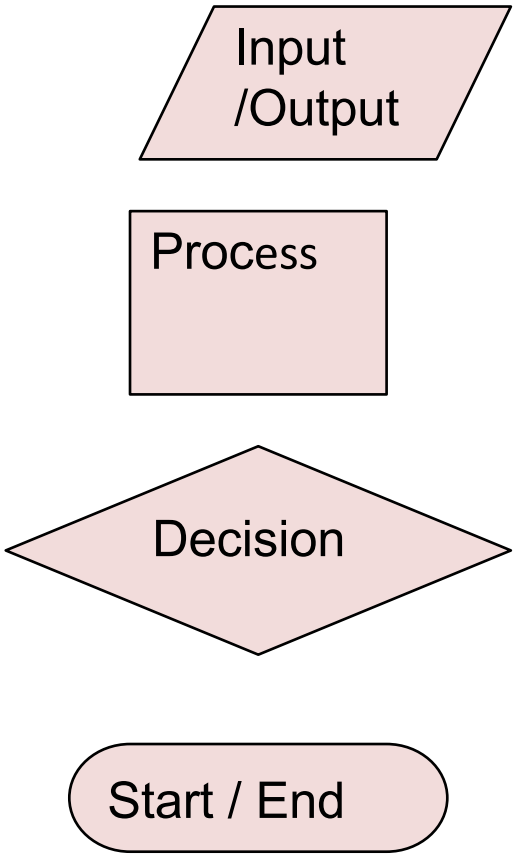
- If an algorithm A finishes its work on an input x in a finite time, then we say that the algorithm A ***halts*** on x.

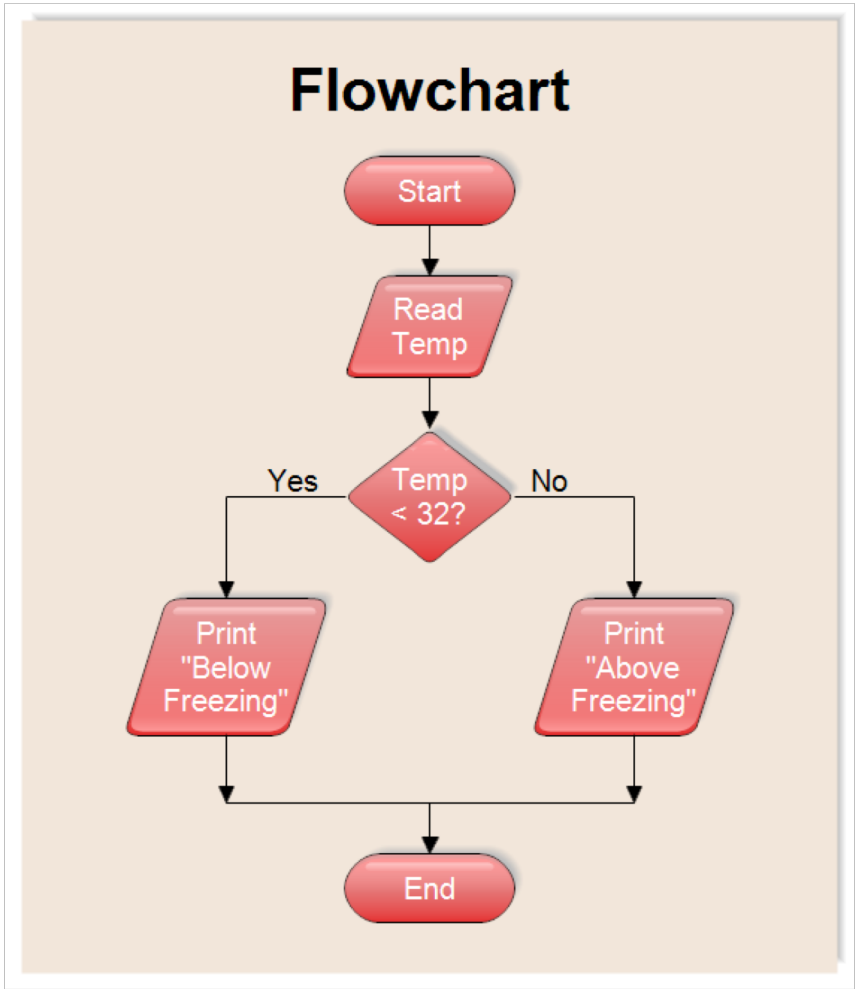
Formal Definition:

A is an algorithm solving a given problem/task if

- For any input, A halts
- For any input, A produces the correct result

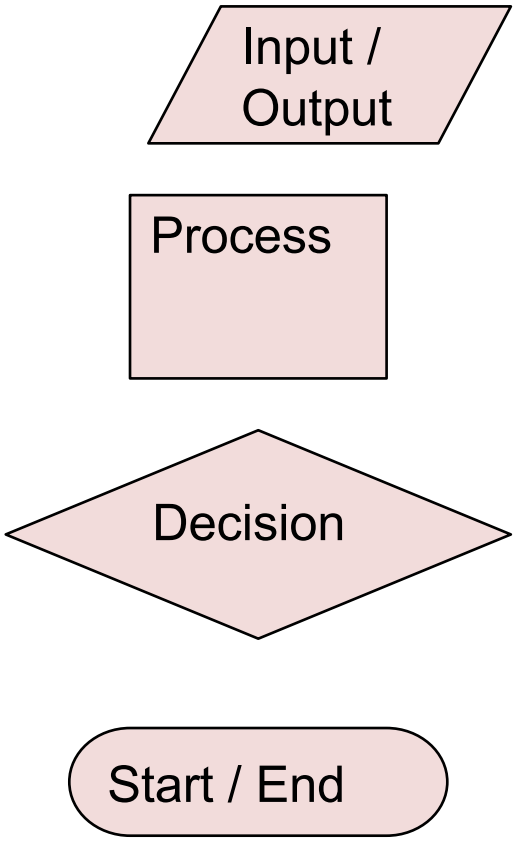
Flowcharting



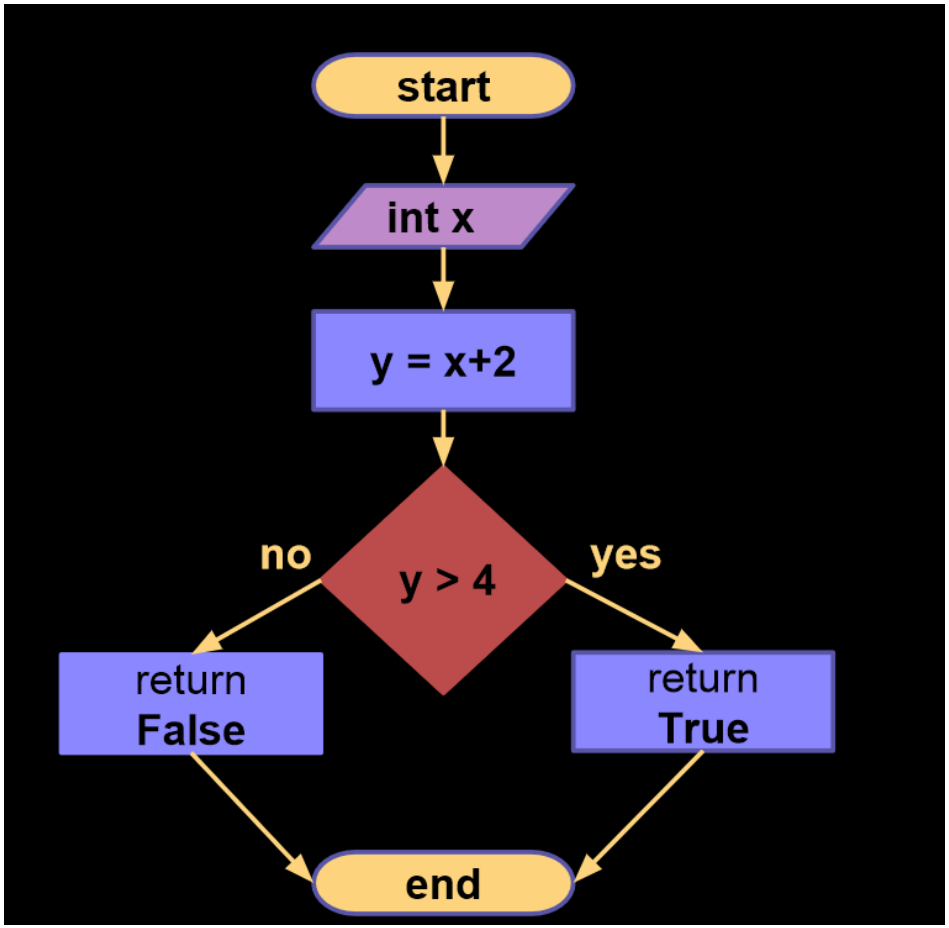


Is x + 2 greater than 4….

Recall our building blocks…



A flowchart….



Challenge…

The problem…

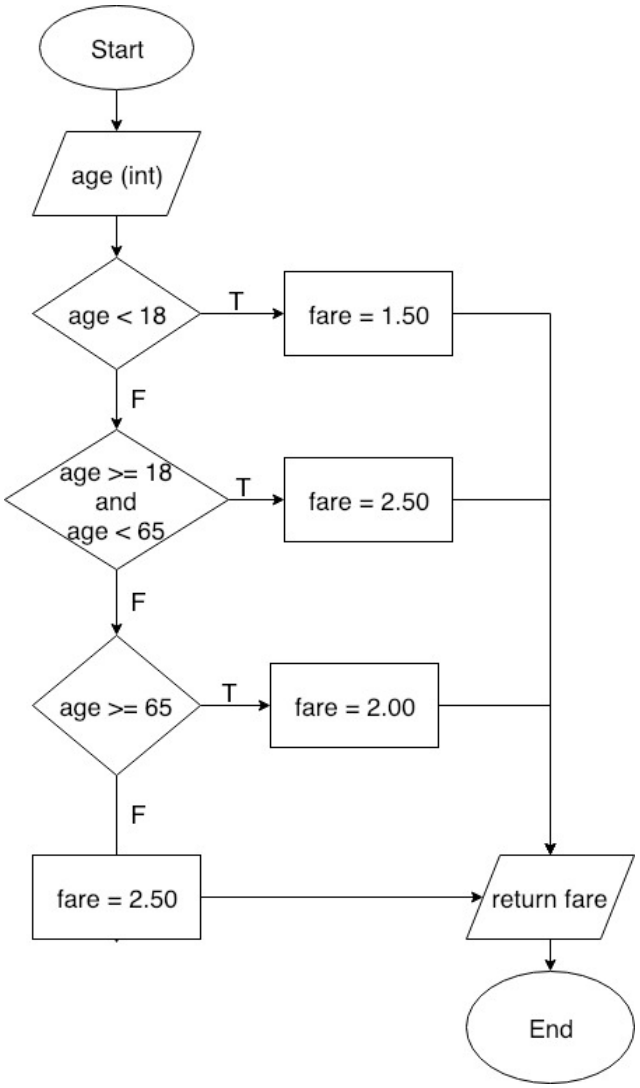
- You have been asked to design a program that determines the cost of riding the bus based on the age of the rider.
  If rider is less than 18, the cost is $1.50.
  If the rider is 65 or older, the cost is $2.00.
  For all other riders, the cost is $2.50.

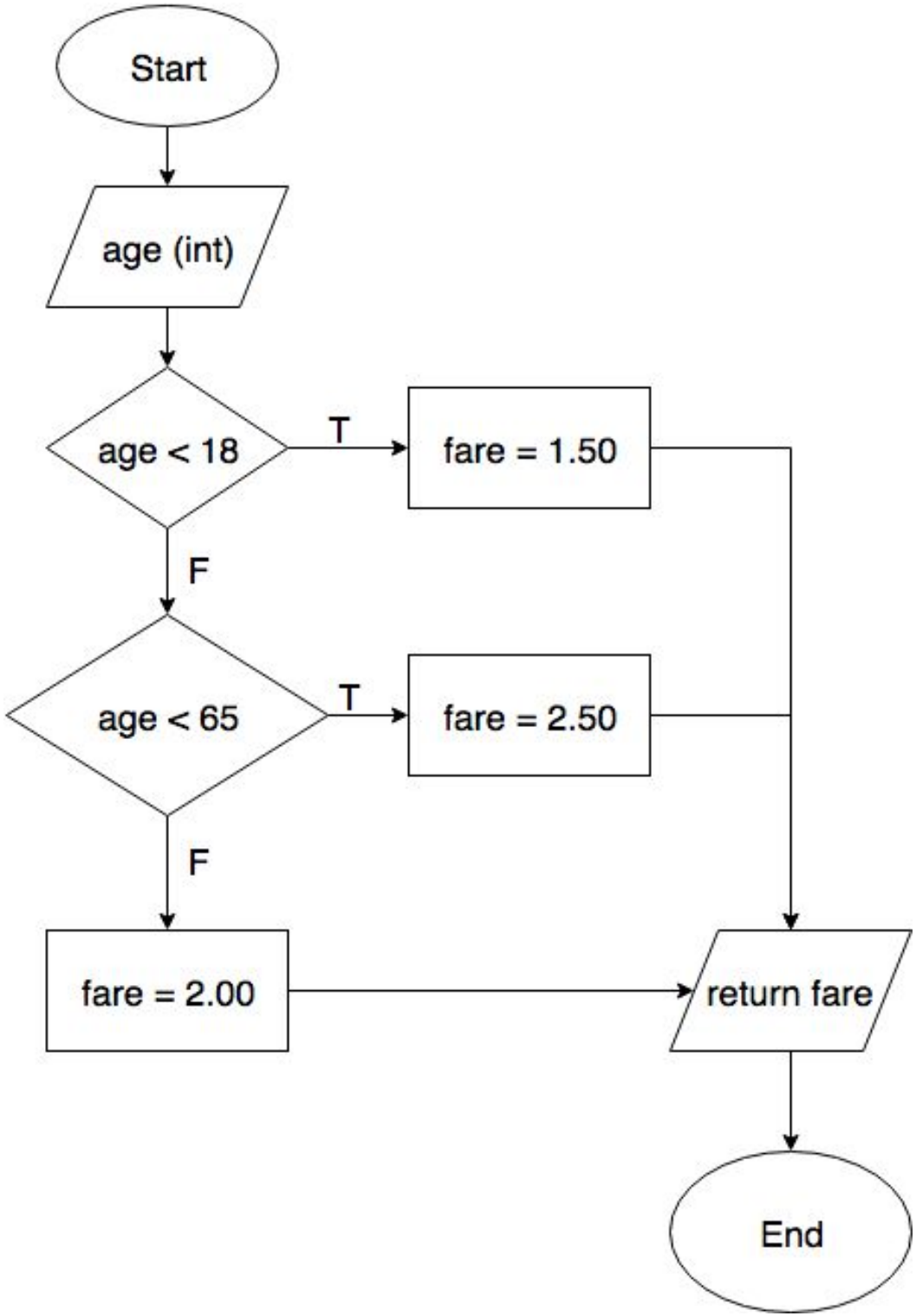
Our process…

- analyze the problem
  - identify the input(s)
  - identify the output(s)
  - articulate the purpose of the algorithm in terms of the input(s) and output(s)
  - articulate any assumptions
- work through example(s) of the problem
- design/implement an algorithm solution
- test your solution

Is this correct?

- input: age (integer value)
- output: price (floating point number)
- purpose: given the age of the rider, return the price of the bus fare
- examples
  - age 17 -> 1.50
  - age 18 -> 2.50
  - age 64 -> 2.50
  - age 65 -> 2.00
  - age 100 -> 2.00




Start
age (int)
age < 18
T
fare = 1.50
F
age < 65
T
fare = 2.50
F
fare = 2.00
return fare
End

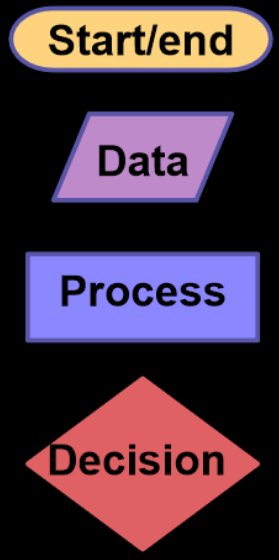
Pseudocode

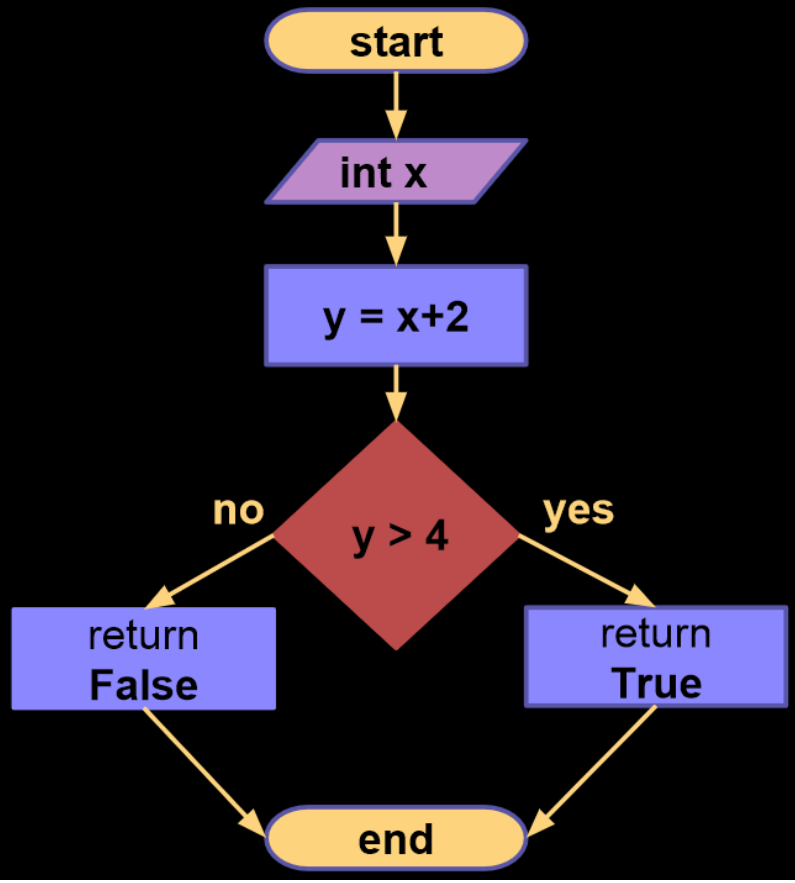
- Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm.
  - Wikipedia
  - http://en.wikipedia.org/wiki/Pseudocode

Pseudocode…

A bit of geometry:

```
isXplus2GreaterThan4(int x)
    y = x+2
    if y > 4
        return True
    else
        return False
```





next challenge…

- write a pseudocode representation of our algorithm for determining the cost of a bus fare

is this correct?

```
getFare (age)
        fare

        if (age < 18)
                fare = 1.50
        if (age < 65)
                fare = 2.50
        else
                fare = 2.00

        return fare
```

solution

```
getFare (int age)
        fare

        if (age < 18)
                fare = 1.50
        else if (age < 65)
                fare = 2.50
        else
                fare = 2.00

        return fare
```

Challenge…

The problem…

- The real roots of the quadratic equation are given by the formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

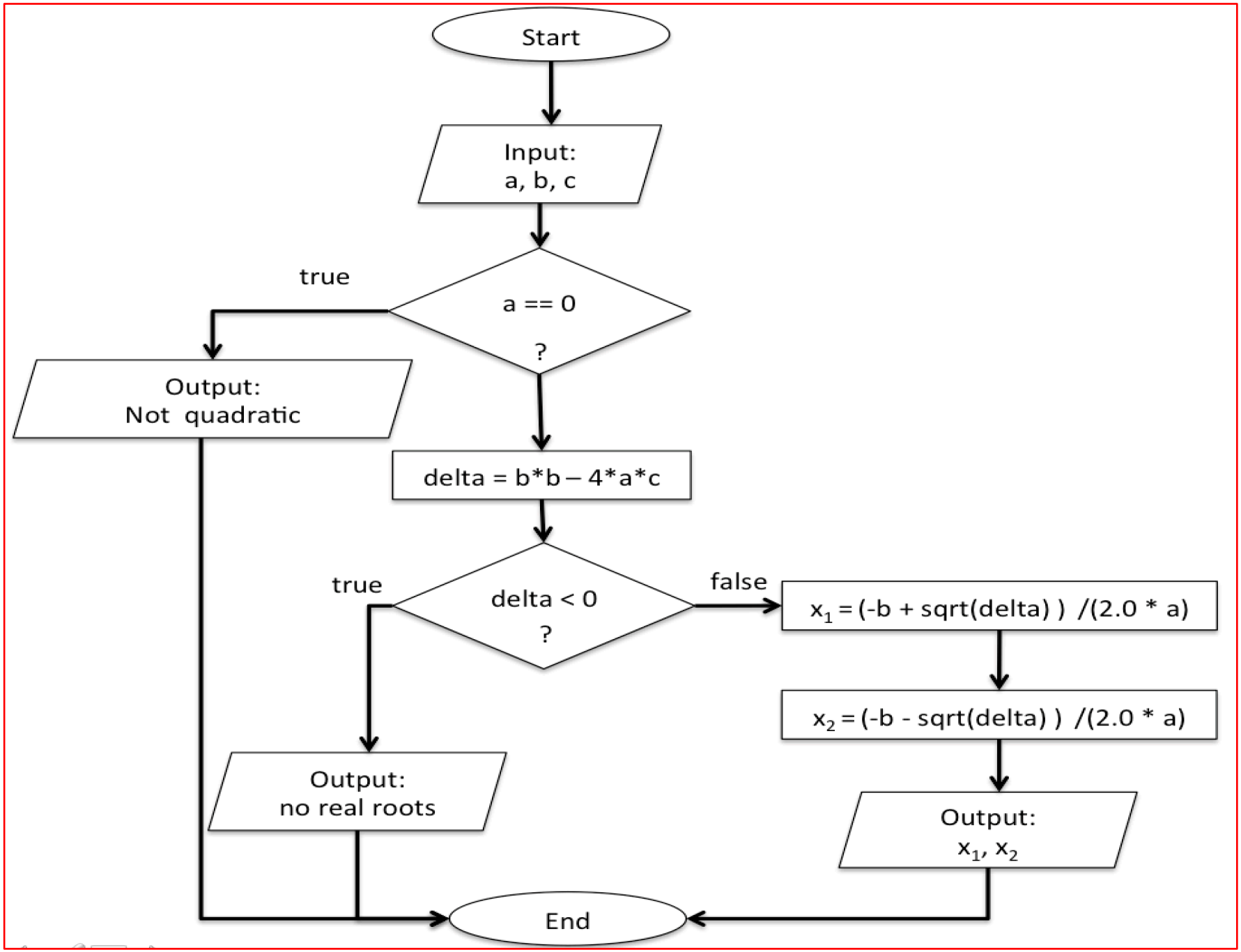
- assuming that $b^2 - 4ac \geq 0$
- There are no real roots when $b^2 - 4ac < 0$
- Design a program that prints the roots of the quadratic equation, if they exist, otherwise print “no real roots”
- NOTE: You can assume there is an operation for computing the square root of a number. You cannot assume there is a ± operator.

Our process…

- analyze the problem
  - identify the input(s)
  - identify the output(s)
  - articulate the purpose of the algorithm in terms of the input(s) and output(s)
  - articulate any assumptions
- work through example(s) of the problem
- design/implement an algorithm solution
- test your solution

Analyzing the problem

- input: a, b, c (float)
- purpose: calculate and print the roots of the quadratic equation using a, b, c, prints error message if not able to calculate
- examples
  - a, b, c = 0 -> not quadratic, division by zero
  - a, b, c = 1 -> no real roots
  - b=2, a, c = 1 -> x1 = (-2 + sqrt(0)) / 2 x2 = (-2 - sqrt(0)) / 2
  - b=6, a=5, c=1 -> x1 = (-6 + sqrt(16)) / 10 x2 = (-6 - sqrt(16)) / 10


Start
Input:
a, b, c
true
a == 0
?
Output:
Not quadratic
delta = b*b – 4*a*c
true
delta < 0
?
false
x1 = (-b + sqrt(delta)) /(2.0 * a)
x2 = (-b - sqrt(delta)) /(2.0 * a)
Output:
no real roots
Output:
x1, x2
End

ToDos…

- Labs start this week! Make sure you are registered and attending!
- Assignment 1 is up
  - Get started