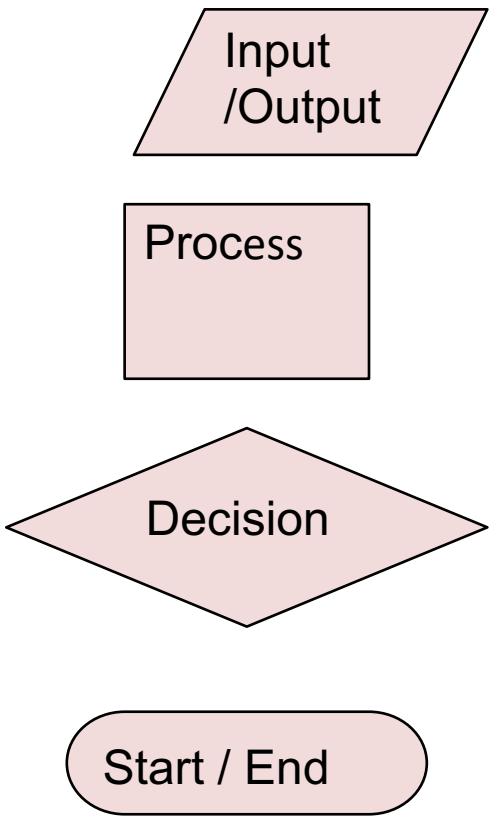
CSc 106: The Practice of Computer Science

Algorithms continued

Recall…

- 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
  - Precision of terms (need a well defined language)
  - Expressiveness of terms (need abstractions)

Flowchart shapes and problem solving process…

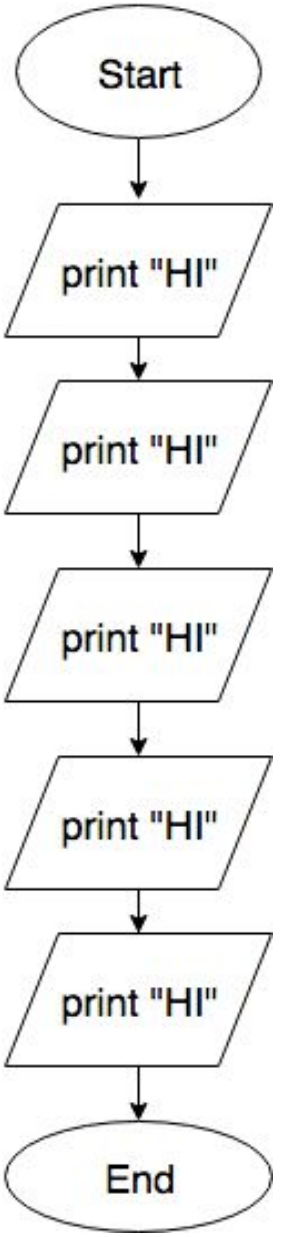


- analyze the problem
  - identify the input(s)
  - identify the output(s)
  - articulate the purpose of the algorithm in terms of the input(s) and output(s)
  - articulate any assumptions
- work through example(s) of the problem
- design/implement an algorithm solution
- test your solution

Problem

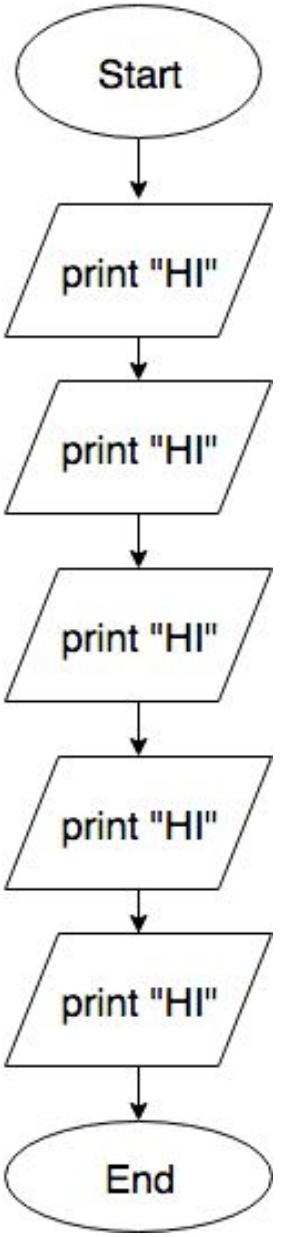
- You have been asked to design a program that will print the word “HI” five times.
  Your friend came up with an algorithm represented in the following flowchart.

- Does it meet the criteria for a good algorithm?
  - definite, finite, input, output, effective



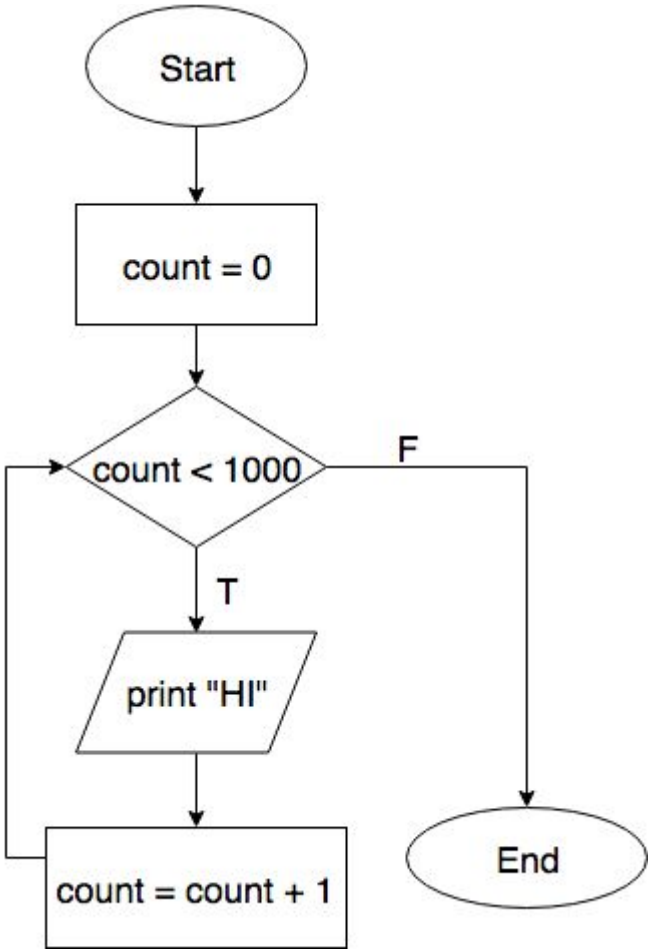
Problem

- You have been asked to design a program that will print the word “HI” five times.
  Your friend came up with an algorithm represented in the following flowchart.
- The specification changes. Your program now must print the word “HI” 1000 times.
- What do you think of the expressiveness of this set of instructions?
- What abstraction would you like to be able to use?



the count-driven loop

- abstraction supported in many languages
  - for-each
  - for loop



clicker

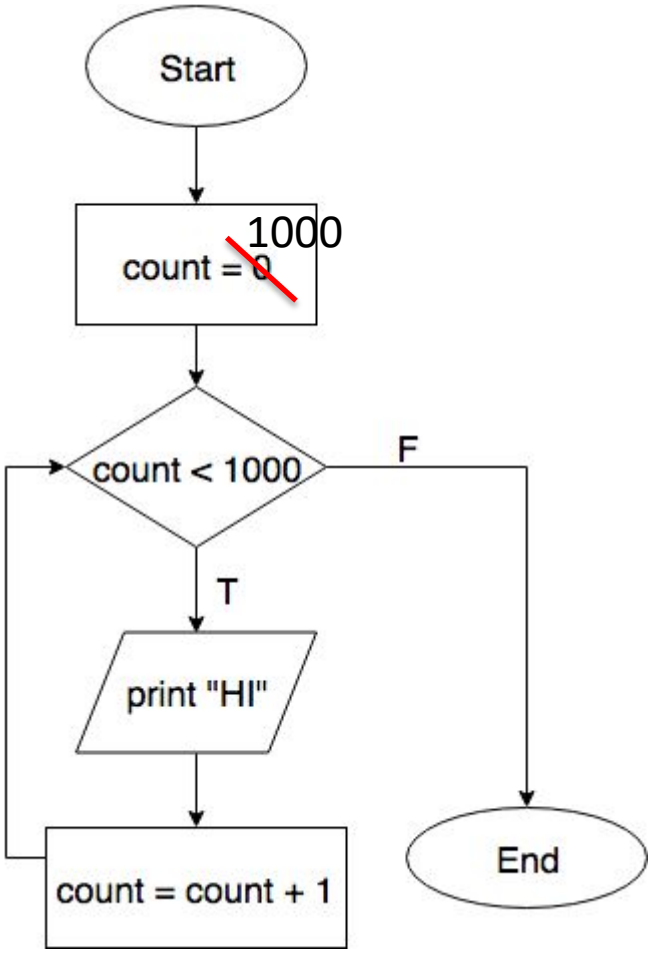
- What would happen if count was initialized to 1000 instead of 0?

A) the output wouldn't change

B) "HI" would only print once

C) "HI" would not print at all

D) it would printing "HI" forever



clicker

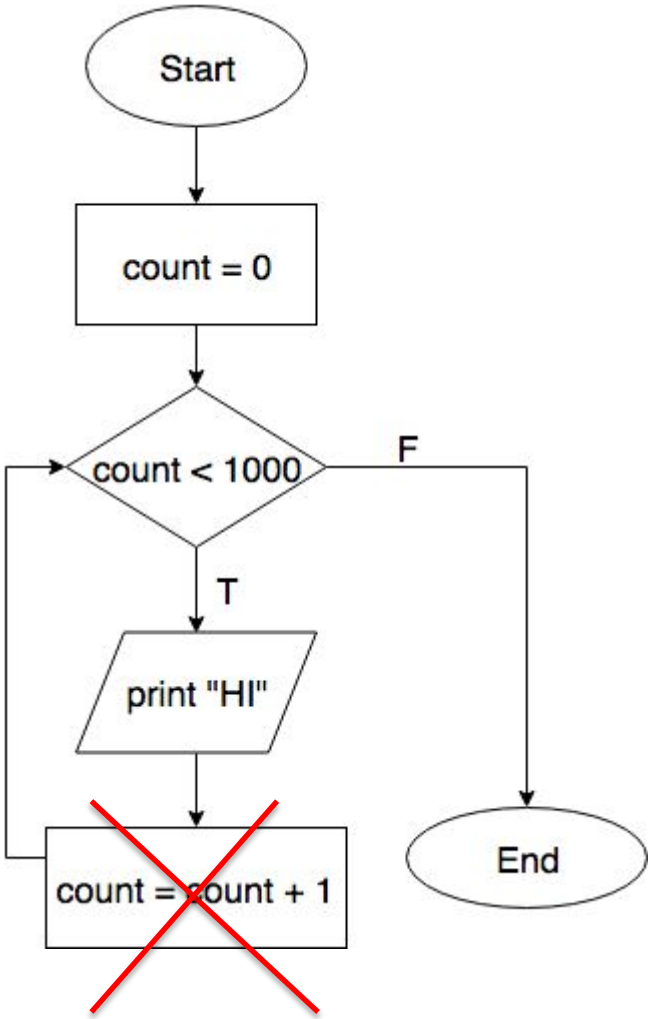
- What would happen if we removed the instruction: count = count + 1

A) the output wouldn't change

B) "HI" would only print once

C) "HI" would not print at all

D) it would printing "HI" forever



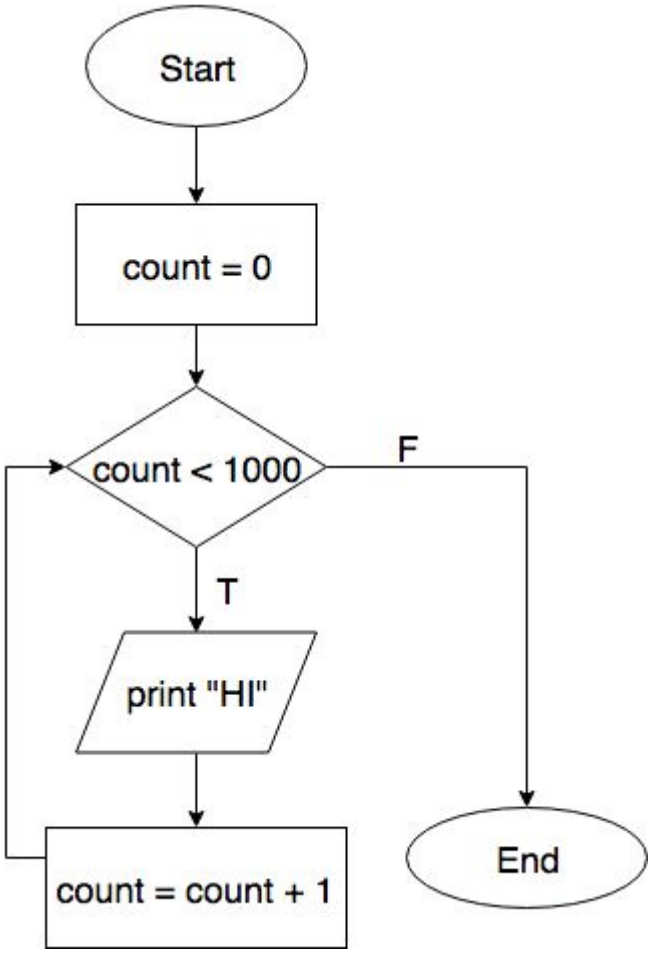
Formal Definition:

A is an algorithm solving a given problem/task if

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

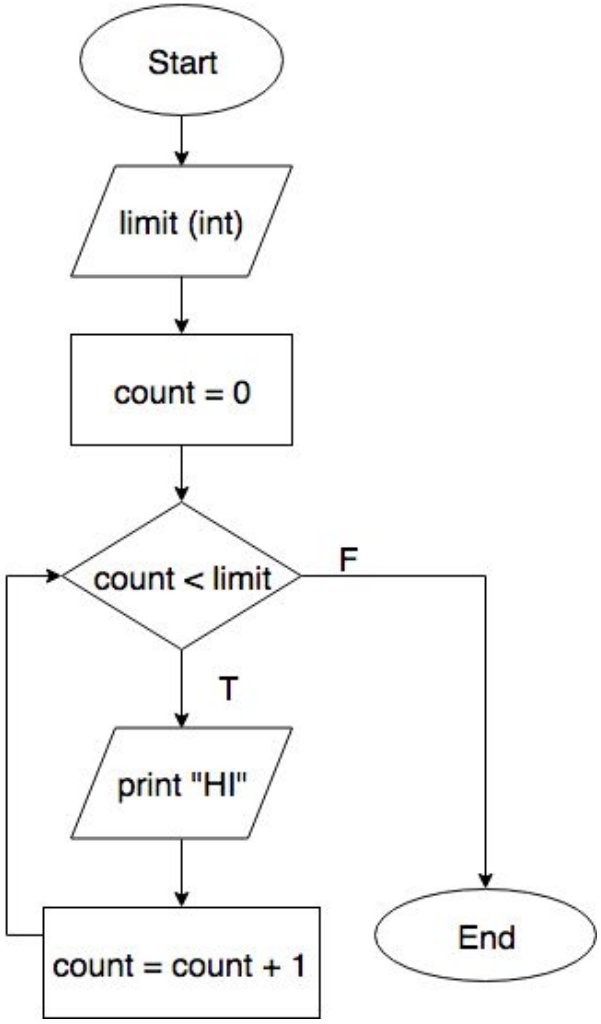
Problem

- The specification changes again. Your program now must take as input the number of times to print "HI"
- How does your flowchart change?



Problem

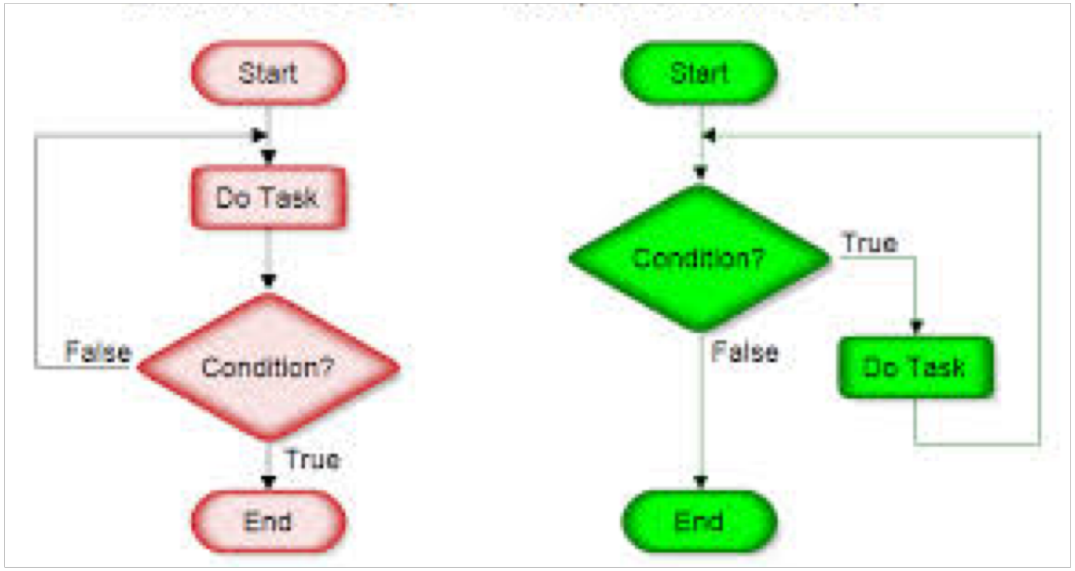
- The specification changes again. Your program now must take as input the number of times to print "HI"
- How does your flowchart change?
- Convert to pseudocode



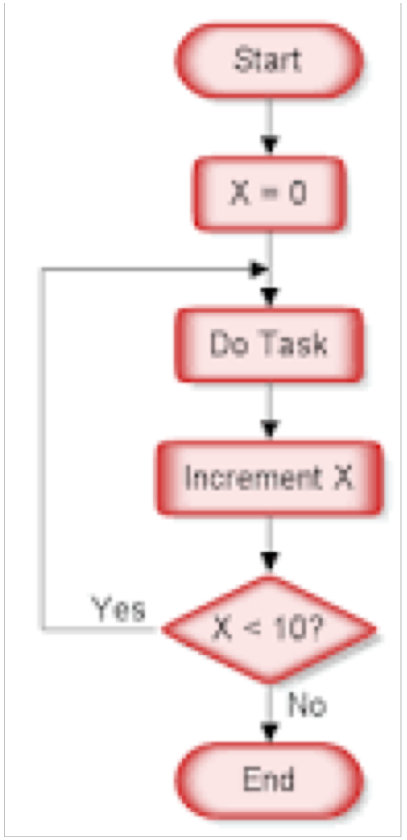
pseudocode

```
printHi (limit)
        count = 0
        for (count = 0 up to limit (not including))
                print “HI”
```

Repeating processes



Repeat until a condition is met
While loops



Repeat until a condition is met and that condition is count based:
For loop

Challenge…

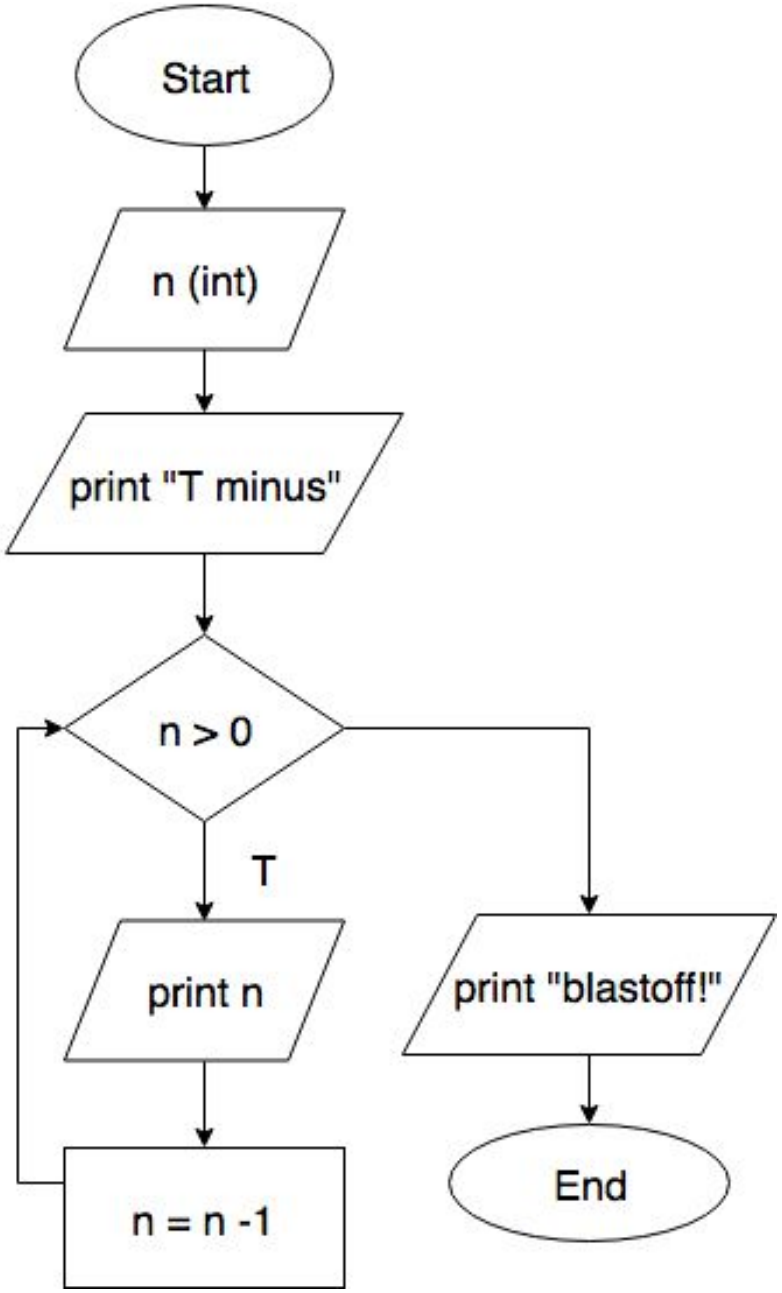
The problem…

- You have been asked to design a program that takes a number and prints a count down.
- For example, if the number is 5, it should print:
  T minus 5 4 3 2 1 blastoff!

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

- input: n (integer)
- output: a countdown (string)
- purpose: print a countdown from n to 1
- assume: n > 0
- examples
  - n 1 -> T minus 1 blastoff!
  - n 3 -> T minus 3 2 1 blastoff!



pseudocode

```
countDown (n)
        print “T minus”

        for (n down to 1 (including))
                print n

        print "blastoff!”
```

Challenge…

The problem…

- You are asked to design a program that allows the user to enter ages for an undetermined number of bus riders.
  The program should return the total bus fare for all riders.
  The user will repeatedly enter the ages, one at a time, and finally enters the value -1 to indicate that there are no more ages to enter.

from last lecture

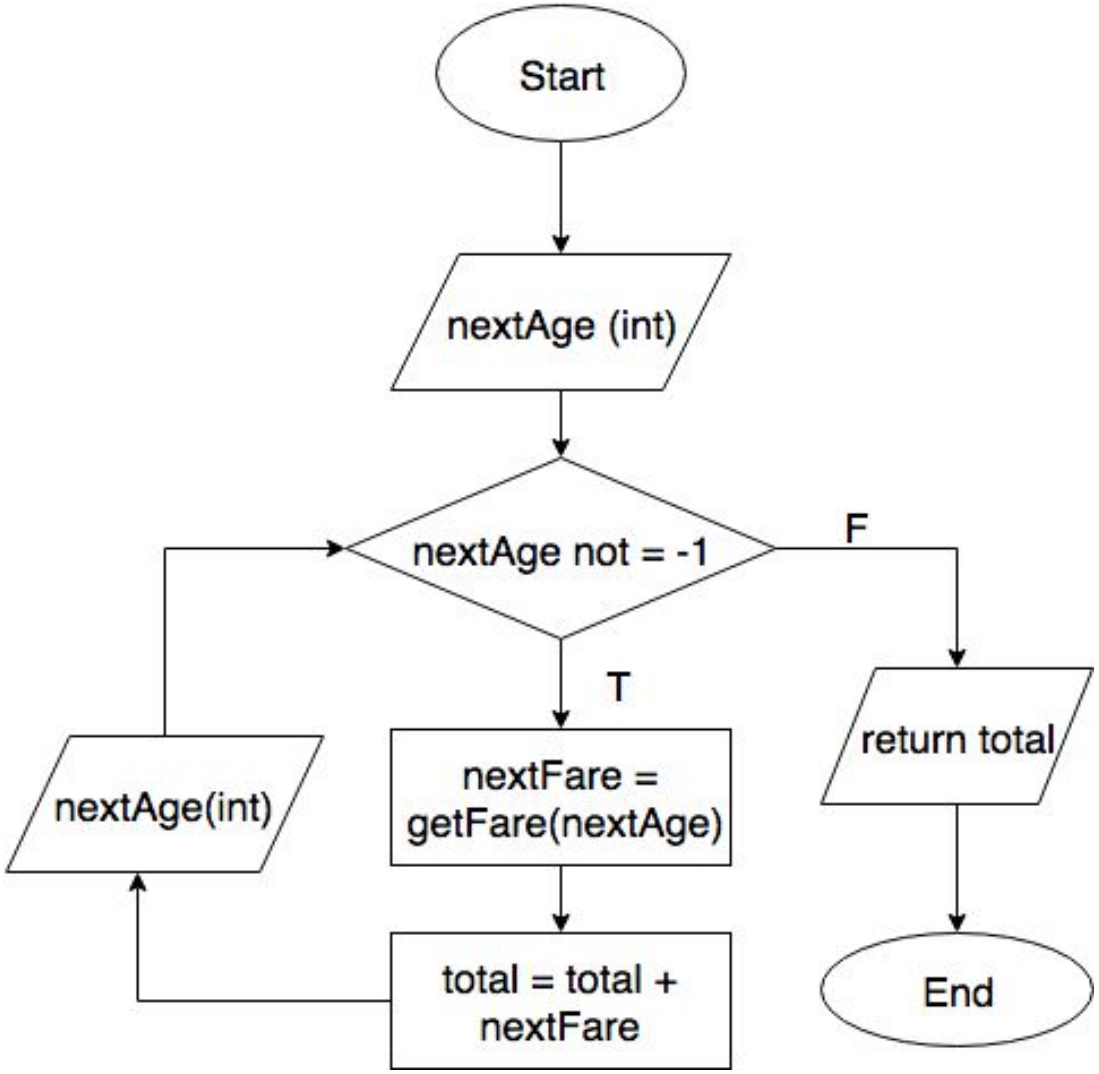
```
getFare (int age)
        fare

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

        return fare
```

The process

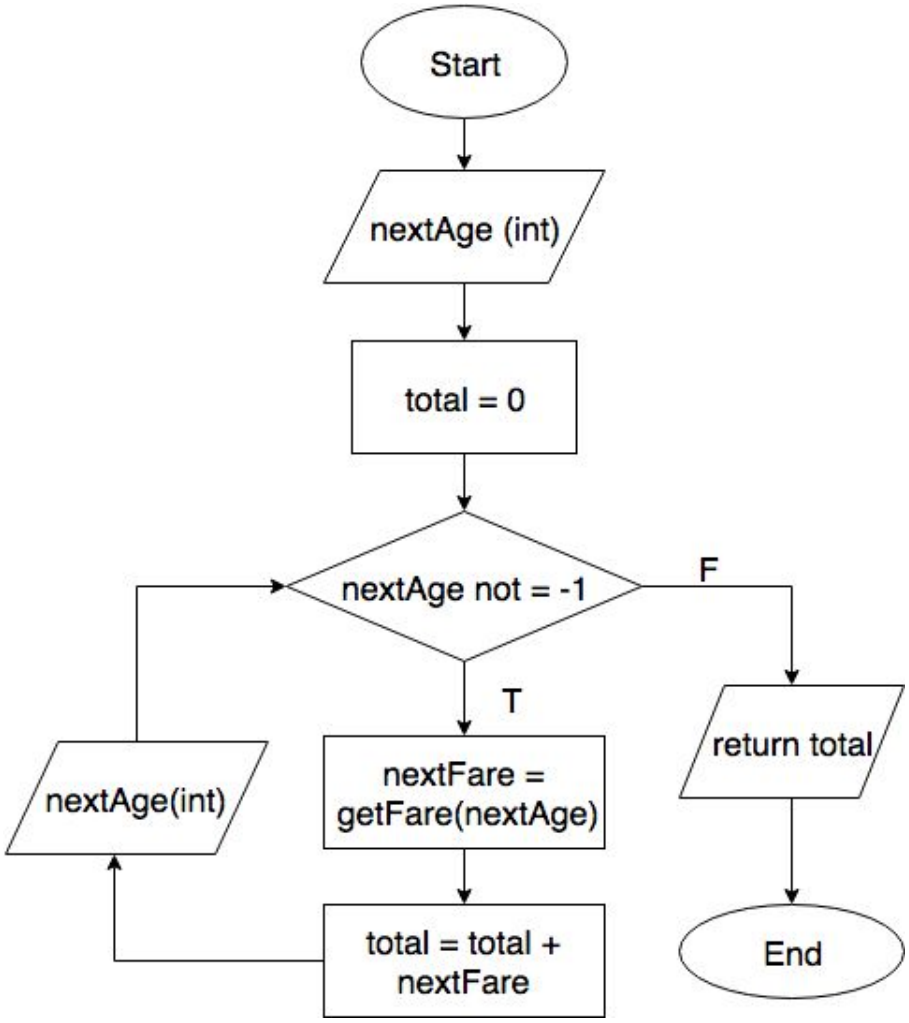
- input: nextAge (integer)
- output: total bus fare (float)
- purpose: returns the total bus fare for every age entered until a -1 is entered
- examples
  - user enters: -1 -> 0
  - user enter: 17, 66, -1 -> fare for age 17 + fare for age 66 -> 1.50 + 2.00 = 3.50

- 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 a good algorithm?



Convert to pseudocode



pseudocode

```
getTotalFare ( )
        nextAge from user
        total = 0
        while (nextAge != -1)
                nextFare = getFare(nextAge)
                total = total + nextFare
                nextAge from user

        return total
```