HIGH-LEVEL PROGRAMMING I

Intro to Algorithms

by Prasanna Ghali

Outline

- What is an Algorithm?
- Algorithm characteristics
- Elements of an algorithm
- Expressing algorithms
- Pseudocode
- Flowcharting
- Examples

What is an Algorithm?

- Step-by-step method for solving a problem
- Similar in concept to a cook's recipe

Algorithm Example: Recipe for Brownies

½ cup butter
1 cup sugar
2 eggs
1 tsp vanilla extract
½ cup cocoa
½ cup flour

1. If butter not soft, then melt butter
2. Blend melted buffer and sugar until mixture has creamy consistency
3. Add eggs and vanilla; stir
4. Add cocoa and flour; mix until well blended
5. Pour into greased round glass cake pan
6. Microwave for 9-10 minutes

Examples of Algorithms

- Putting together IKEA furniture
- Going from airport to hotel in new city
- Solving Rubik's cube
- Playing board games such as Chess, Go, …
- Solving a maze
- Choosing a TV cable plan
- Choosing a cell phone subscription plan

Oldest Algorithm?

- First documented algorithm was presented by Greek mathematician Euclid in 300 B.C. to compute Greatest Common Divisor (GCD)
  1. Let A and B be integers such that $A > B \geq 0$
  2. If $A = 0$, then GCD is B and algorithm ends
  3. If $B = 0$, then GCD is A and algorithm ends
  4. Write quotient remainder format: $A \leftarrow B \cdot Q + R$
  5. Set $A \leftarrow B$ and $B \leftarrow R$
  6. Go to 2

Formal Definition

- Procedure to solve well-specified problem
- Must have following characteristics
  - Input
  - Output
  - Precision
  - Finiteness
  - Uniqueness
  - Generality

Algorithm Elements (1/4)

- Mid-60s, mathematicians proved that any logic problem, no matter how complicated, can be constructed using one or more of only three *structures* or *elements*
  - Think of an element as *basic unit of logic*
- What are these three elements?
  - Sequence
  - Selection
  - Iteration

Algorithm Elements (2/4)

- Euclid's Greatest Common Divisor (GCD) algorithm
  1. Let A and B be integers such that $A > B \geq 0$
  2. If $A = 0$, then GCD is B and algorithm ends
  3. If $B = 0$, then GCD is A and algorithm ends
  4. Write quotient remainder format: $A \leftarrow B \cdot Q + R$
  5. Set $A \leftarrow B$ and $B \leftarrow R$
  6. Go to 2

Algorithm Elements (3/4)

- Elements an be combined in infinite ways
  - Every element has single entry and single exit point
  - Elements can be stacked or connected to one another only at their entry or exit points
  - Any element can be nested within another element

Algorithm Elements (3/3)

- We'll add two additional elements
  - Input
  - Sequence
  - Selection
  - Iteration
  - Output

Expressing Algorithms

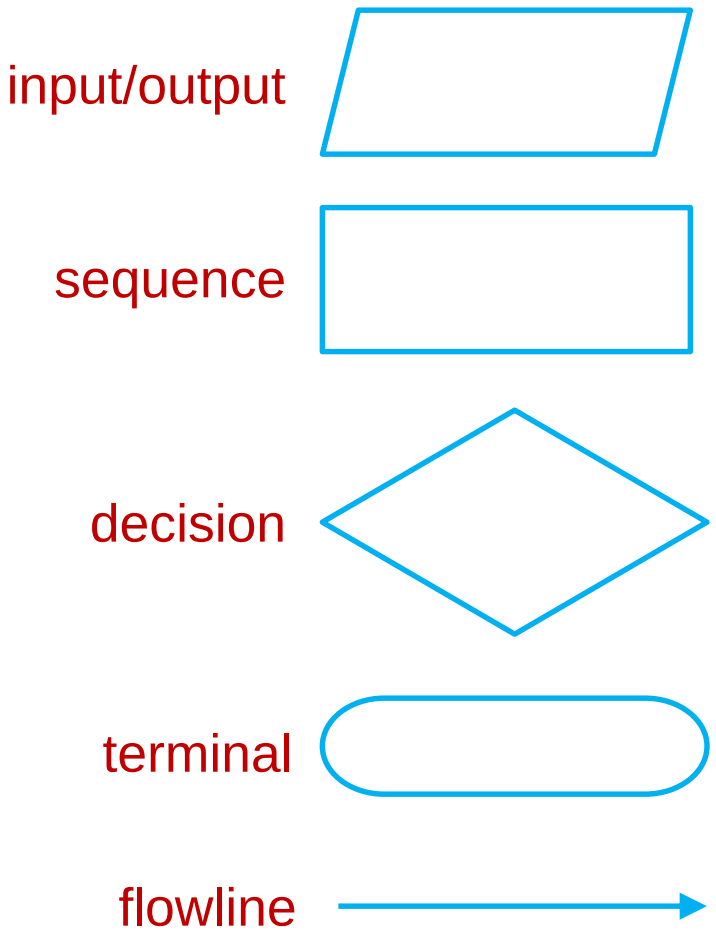
- Need notation for expressing algorithms
- Options range from natural to programming languages
- Two tools commonly used
  - Pseudocode
  - Flowchart

Pseudocode

- *Pseudo* means false
- *Code* a problem means to put it into programming language
- *Pseudocode* means false code
- Sentences that appear to be written in programming language but don't necessarily follow syntax rules of any specific language

Flowcharting

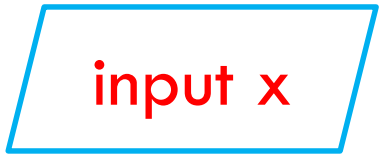
- Combination of graphical symbols that represent logical flow of data thro' solution



Pseudocode: Input

- Can write reading of input from user as:

1: read x

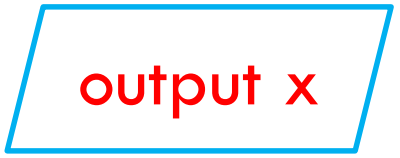


1: read a, b, c

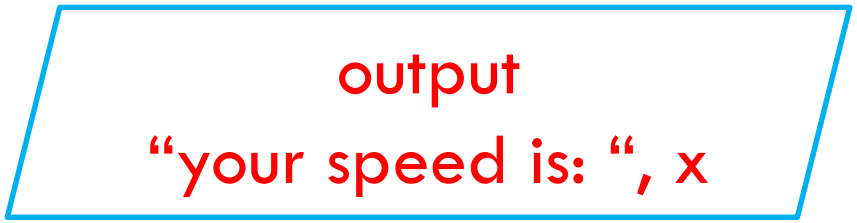
Pseudocode: Output

- Writing of output to user represented as:

1: print x



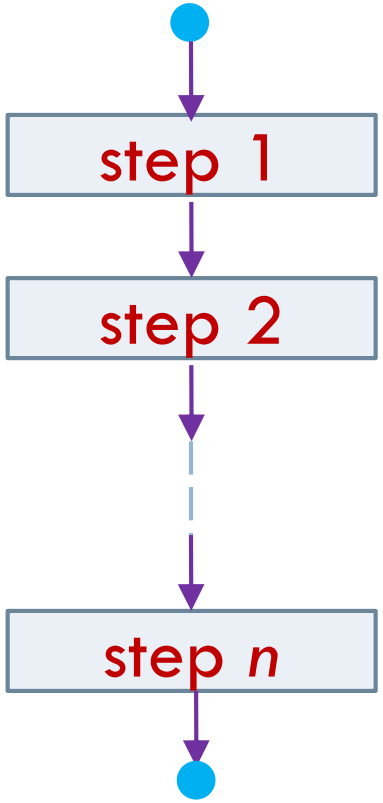
1: print “your speed is: “, x



Sequence (1/2)

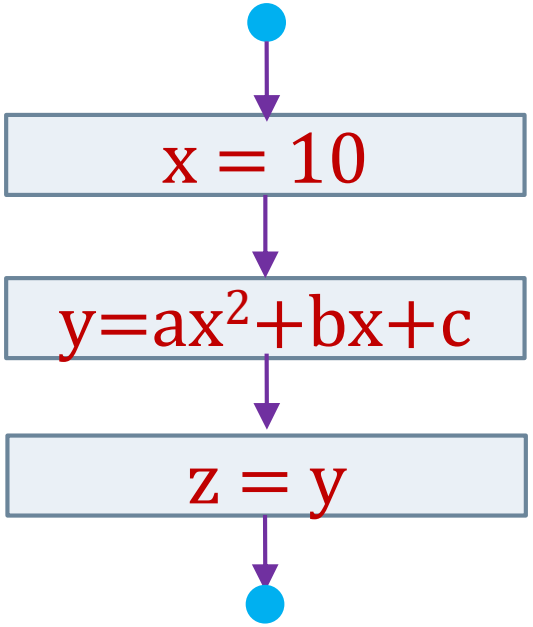
- Computational statements executed in order, one after another
- Once you start series of steps in sequence, you must continue step by step until sequence ends

1: step 1
2: step 2
3: ...
⋮
n: step *n*



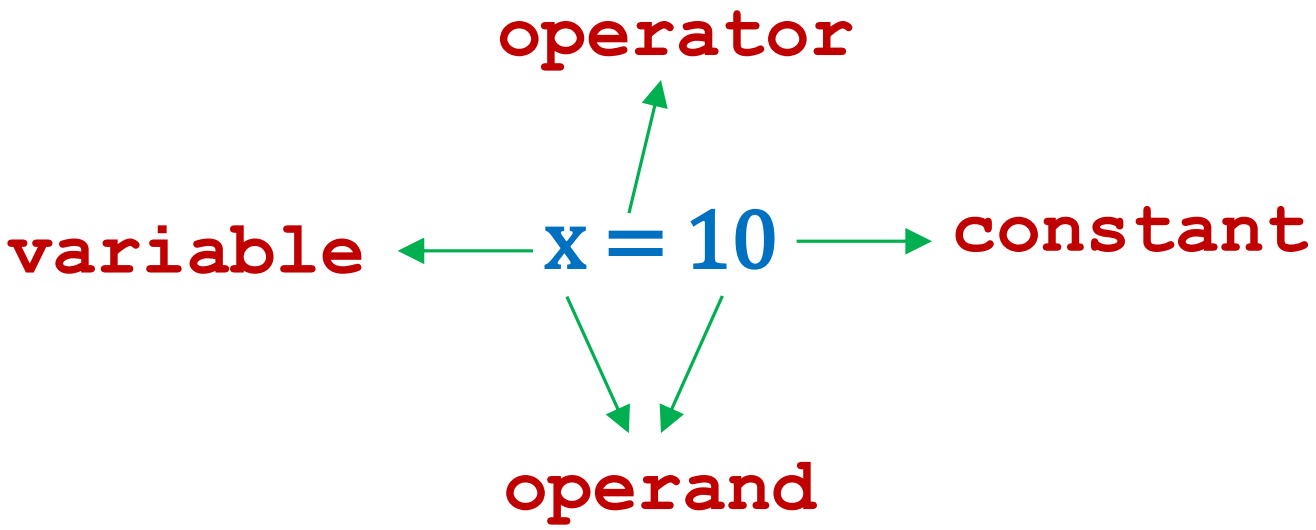
Sequence (2/2)

1: $x = 10$
2: $y = ax^2+bx+c$
3: $z = y$



Statements, Expressions, … (1/4)

- *Statement* made up one or more *expressions*
- *Expression* consists of one or more *operands* and zero or more *operators*
- Operand can be *constant* or *variable*



Statements, Expressions, … (2/4)

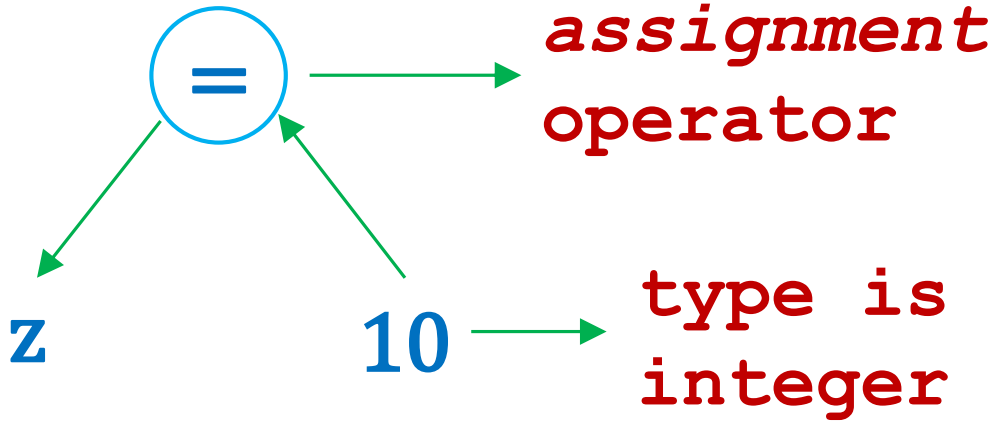
- How many expressions in following statements?

3 ← 1: x = 10
5 ← 2: y = x*x
7 ← 3: z = x + y * y

Statements, Expressions, … (3/4)

- Every *expression* evaluates to a *value* of certain *type*
- *Type* represents set of *values* and set of *operations* that can be applied on these values

1: z is integer
2: z = 10



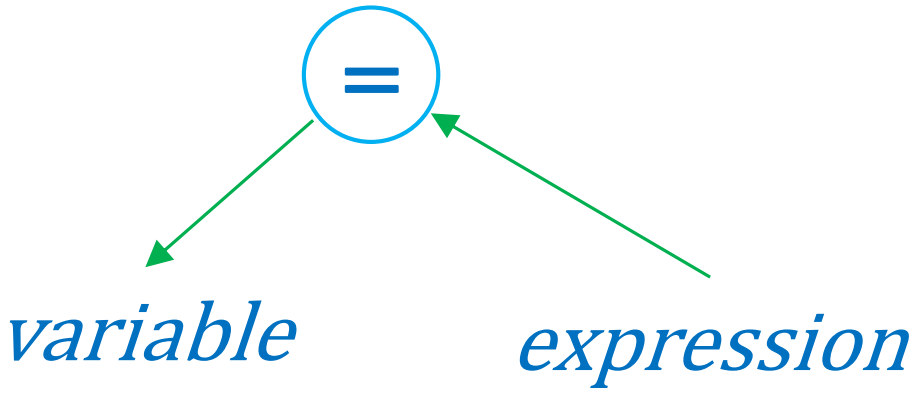
Statements, Expressions, … (4/4)

- Numbers without fractions (such as 10, 33, …) are called *integers*
- Digital systems cannot represent infinite set of fractional values on real number-line
- Instead, they represent a finite set of *floating-point* values with each floating-point value being an approximation of real values

Assignment Operator

variable = *expression*

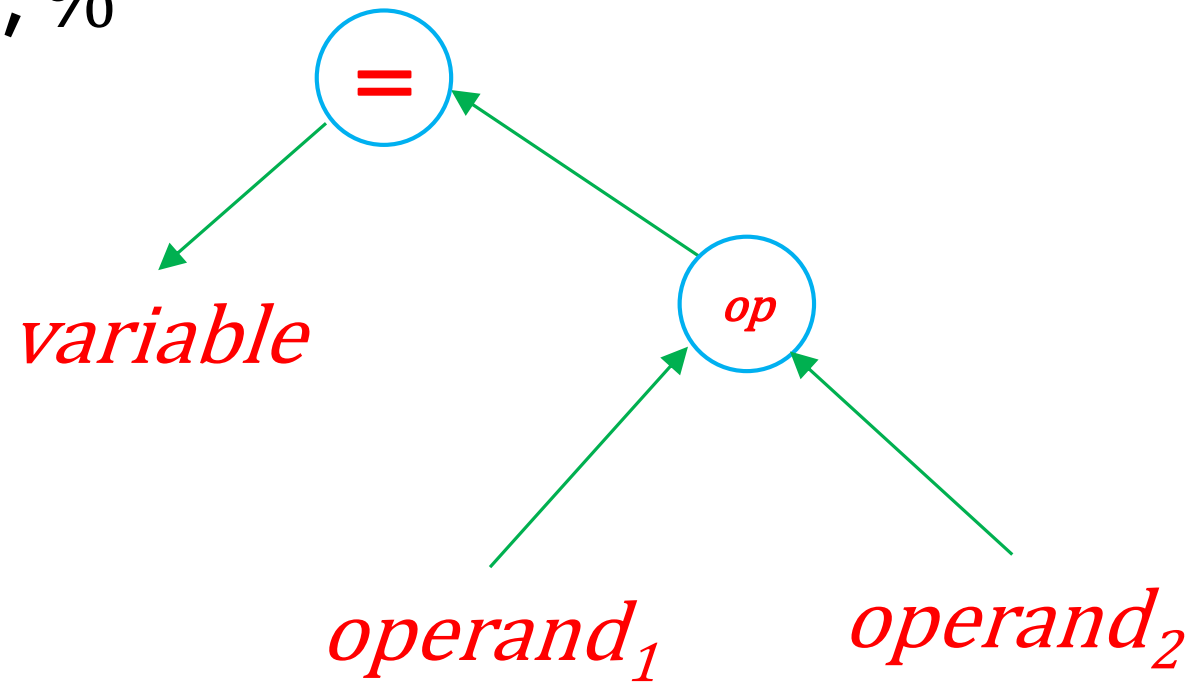
- Copy value of *expression* into *variable*, or
- Replace current value of *variable* by value of *expression*



Arithmetic Operators

variable = *operand*$_1$ ***operator*** *operand*$_2$

- +, -, *, /, %



Relational Operators

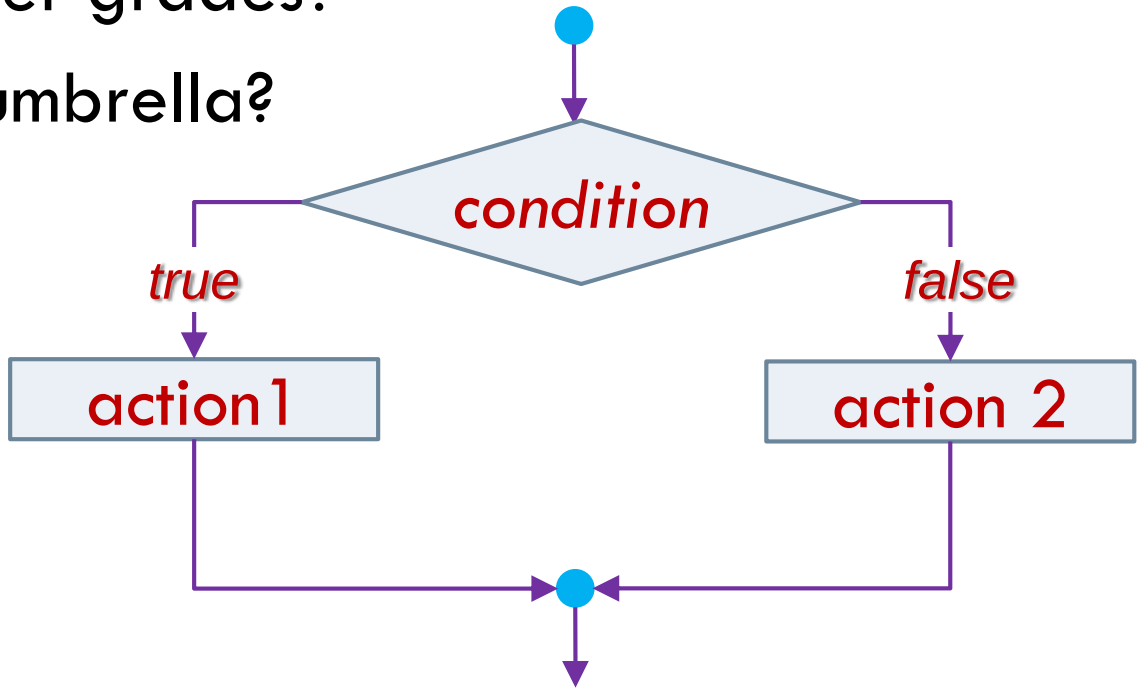
- Used to compare two operands or values or expressions
- ==, !=, >, >=, <, <=
- Expression involving relational operators always evaluates to either 1 (*true*) or 0 (*false*)
  - Value 0 means *false*
  - Any non-zero value means *true*

Logical Operators

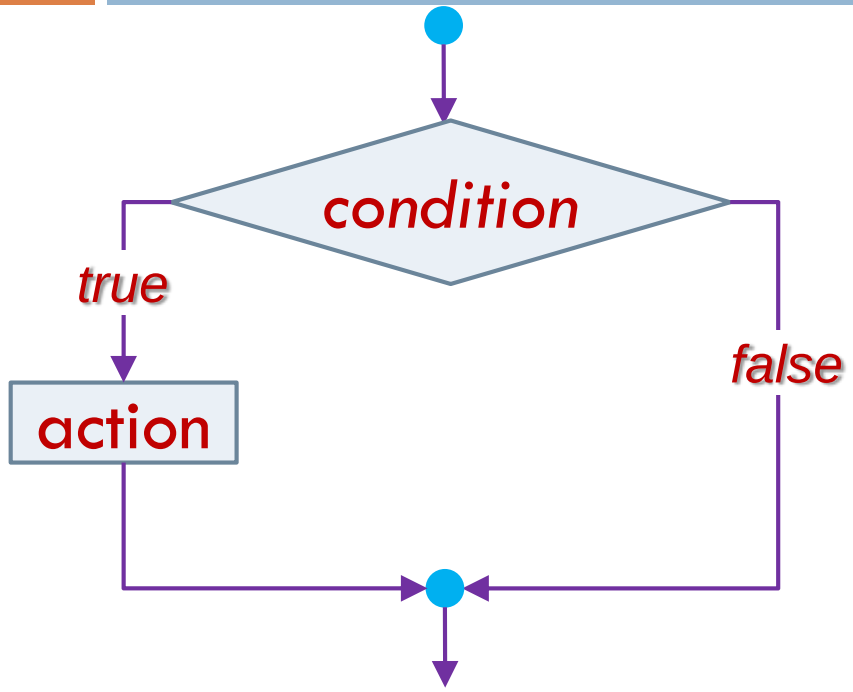
- Used to control sequencing of steps; combine simple relational expressions into more complex expressions
- &&, ||, !
- Expression involving logical operators always evaluates to either 1 (*true*) or 0 (*false*)
- a = x < y || x > z
- b = x >= y && x <= z

Selection Element (1/4)

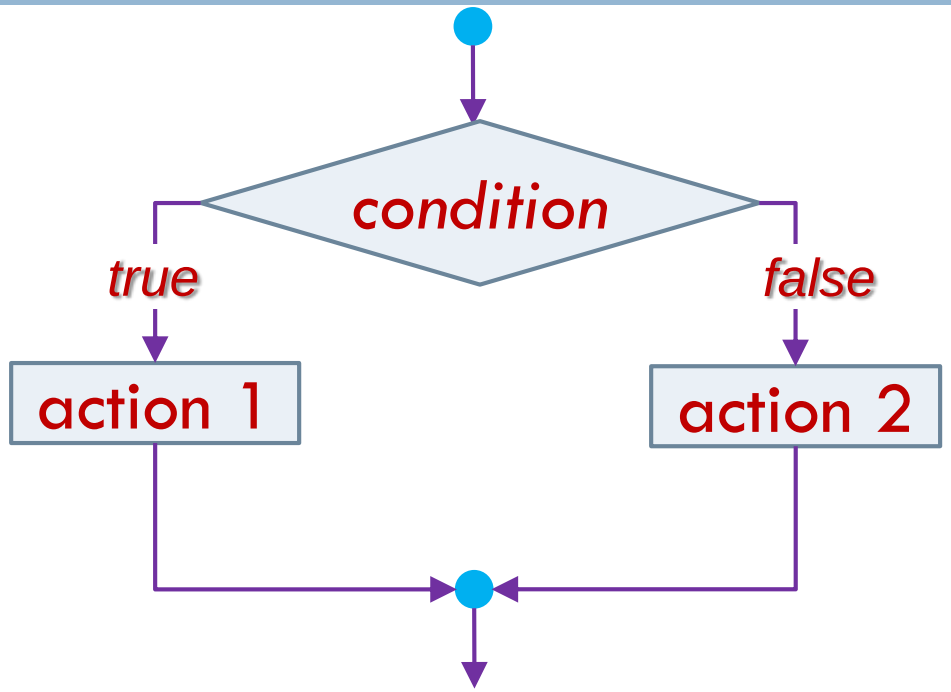
- Alters sequential flow of control
- One of two courses of action is taken based on result of evaluation of *condition*
- How to assign letter grades?
- Should I take an umbrella?
- Overtime pay?
- Cable billing



Selection Structure (2/4)

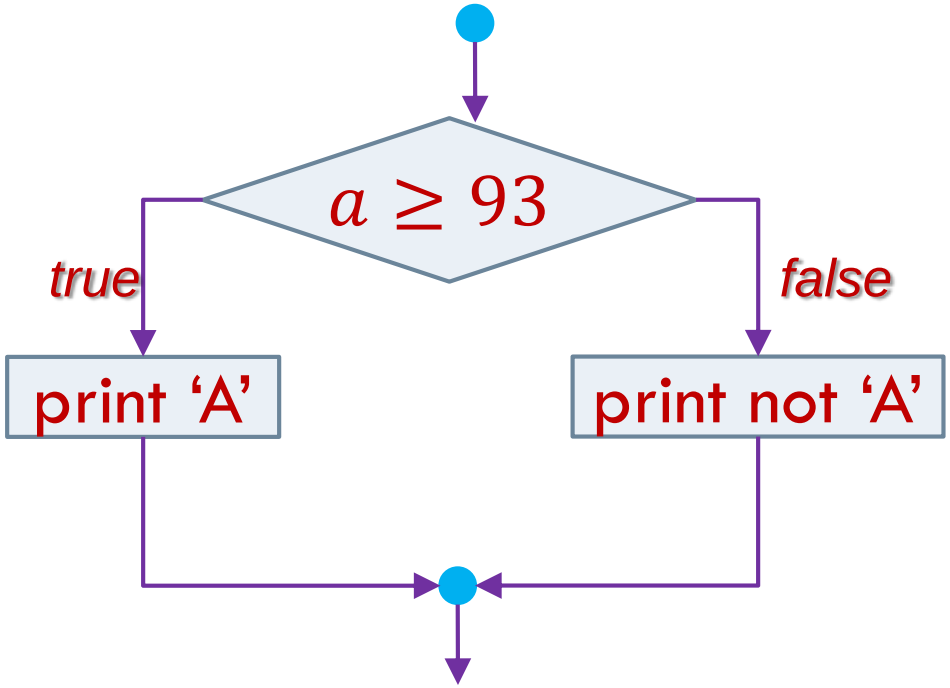



```
if (condition) then
   action
endif
```



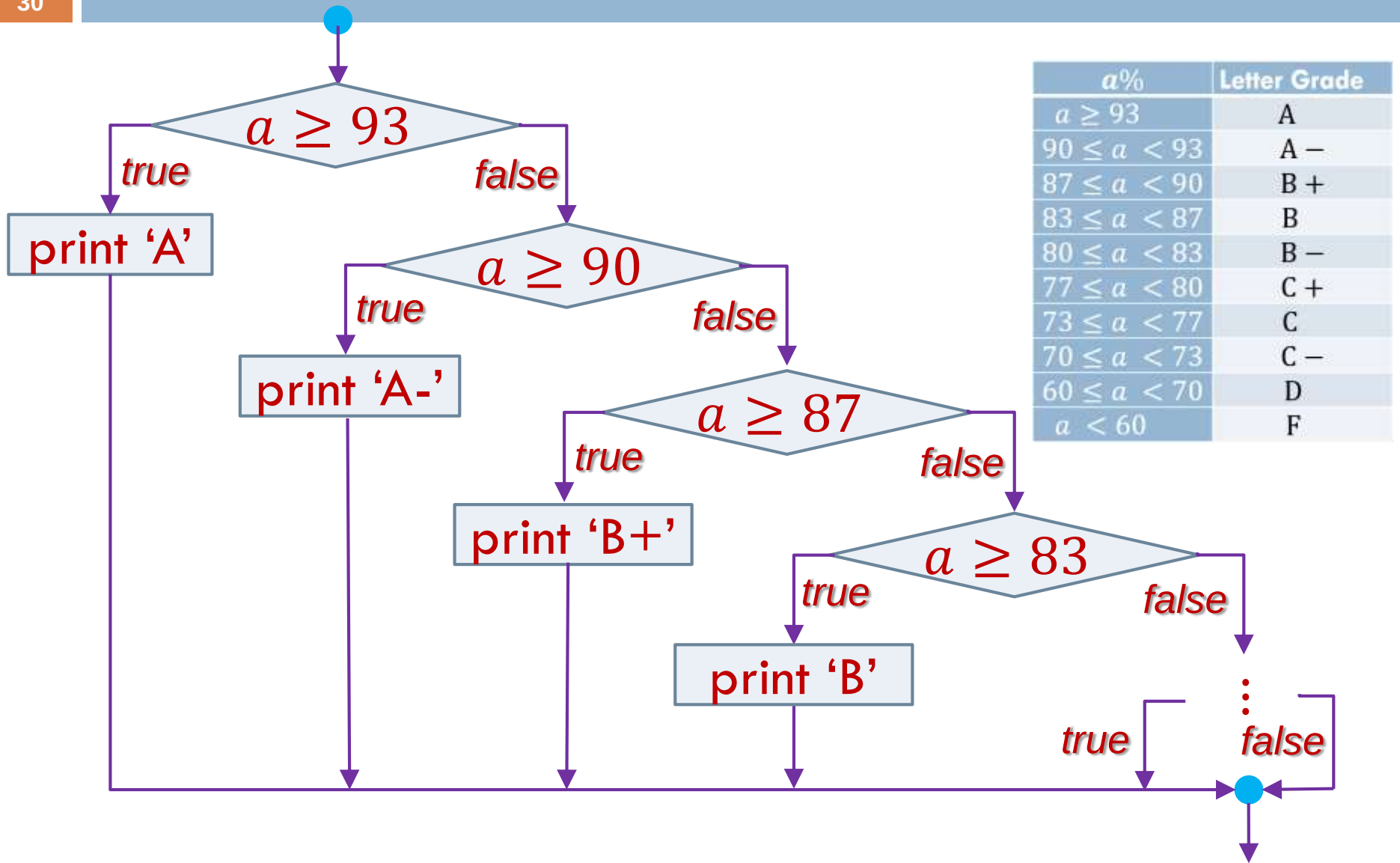

```
if (condition) then
   action 1
else
   action 2
endif
```

Selection Structure (3/4)



a%	Letter Grade
$a \geq 93$	A
$90 \leq a < 93$	A −
$87 \leq a < 90$	B +
$83 \leq a < 87$	B
$80 \leq a < 83$	B −
$77 \leq a < 80$	C +
$73 \leq a < 77$	C
$70 \leq a < 73$	C −
$60 \leq a < 70$	D
$a < 60$	F

Selection Structure (4/4)

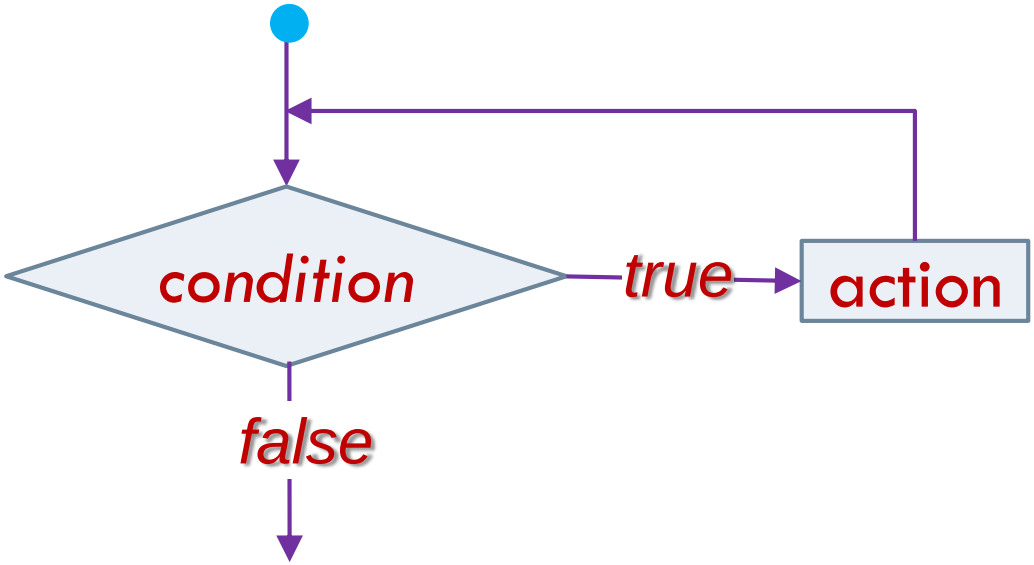


a%	Letter Grade
$a \geq 93$	A
$90 \leq a < 93$	A −
$87 \leq a < 90$	B +
$83 \leq a < 87$	B
$80 \leq a < 83$	B −
$77 \leq a < 80$	C +
$73 \leq a < 77$	C
$70 \leq a < 73$	C −
$60 \leq a < 70$	D
$a < 60$	F

Iteration Element

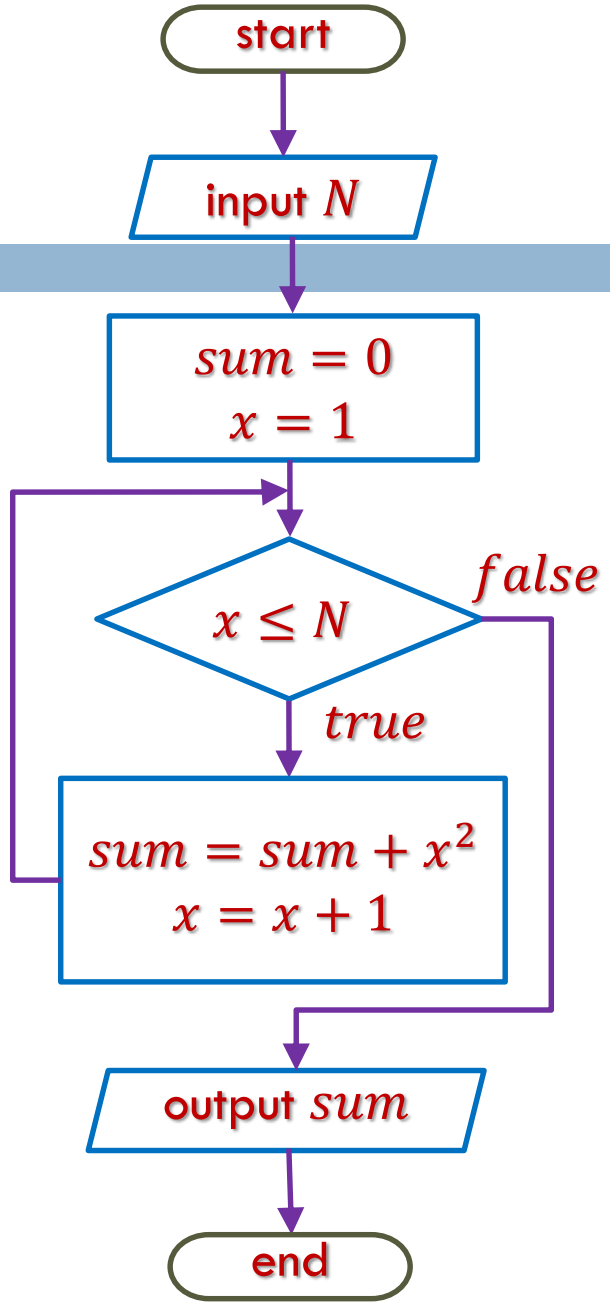
- Repeat actions *while* a *condition* remains *true*

```
while (condition)
   action
endwhile
```

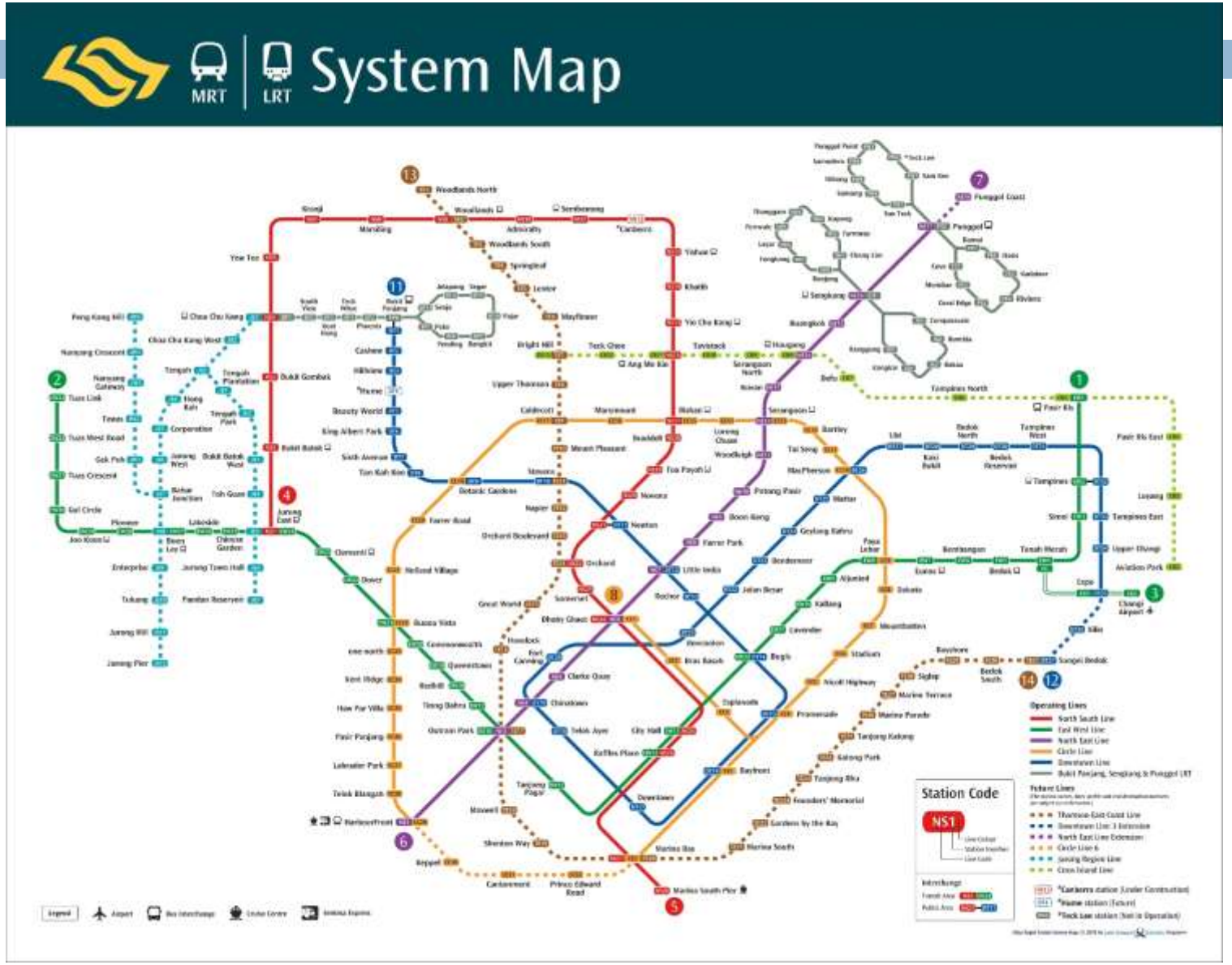


First Algorithm

- Compute $\sum x^2 \ \forall \ x \in [1, N]$
  - 1: read N
  - 2: $sum = 0$
  - 3: $x = 1$
  - 4: while $(x <= N)$
  - 5: $sum = sum + x * x$
  - 6: $x = x + 1$
  - 7: endwhile
  - 8: print sum



Traveling on a Subway



Algorithm: Subway

- **Input:** Source station A and destination station B
- **Output:** Directions to go from A to B using maximum of single transfer
- **if** A and B are on same line L then
- travel on line L from A to B
- **else** [A and B are on different lines]
- find all pairs of lines (L_i, L_j) s.t A is on L_i & B is on L_j and two lines have common station
- **if** there is only one such pair then
- take L_i to a station also on L_j, get off, and then take L_j to station B
- **else** [more than one possible pair of them]
- use pair (L_i, L_j) which requires fewest stops
- **endif**
- **endif**

Summary

- What is an Algorithm?
- Algorithm characteristics: inputs, outputs, precision, finiteness, uniqueness, generality
- Elements of an algorithm: sequence, selection, iteration
- Expressing algorithms: Pseudocode, flowcharting
- Sequence: operator, operand, expression, variable, constant, type, integer and floating-point values, assignment operator, arithmetic operator, relational operators
- Examples
- Things to do: Read handout, complete assignment