DESIGN & ANALYSIS OF ALGORITHMS

CMSC 142, LEC0

Instructor Information

Ara Abigail Ambita
Instructor
09458913320 (Globe)

Official email address:
aeambita@up.edu.ph

Email address for class submissions:
abbyambita@gmail.com

Consultation hours:

- MTH 12:00 pm - 5:00 pm

Remote Learning Setup?

- Where are the materials uploaded?
  - UPV LMS
- Can I take the course fully remote? No
- Do I need to attend live class? Yes

Zoom meeting for the rest of the semester:

Course Requirements

COURSE REQUIREMENTS

1. 3 long exams 30%
2. Final exams 20%
3. Other Requirements 10%
4. Laboratory 40%

Passing Grade: 60%

These percentages may be adjusted if necessary to ensure fair assignment of course grades. Assignments may be weighted individually depending on their degree of difficulty.

Score	Equivalent
94-100	1.0
90-93	1.25
87-89	1.5
84-86	1.75
80-83	2.0
75-79	2.25
70-74	2.5
65-69	2.75
60-64	3.0
50-59	4.0
Below 50	5.0

How can we reach you for consultation?

Consultation hours:

- MTH 12:00-5:00

For consultation, set an appointment 2-3 days in advance with the following email format:

Subject header: **[CMSC 142] Consultation re: topic**

Indicate the following in the email body:

- topic of concern
- mode of appointment (via zoom or f2f)
- proposed schedule of appointment (date and time)
- your name/s and section

Deadlines

a. Deadlines and deadliest deadlines are absolute. Submissions beyond the deadline have deductions while submissions beyond the deadliest deadlines are unacceptable. Submissions ahead of deadline may have bonus points.
b. Deadlines can be extended for some circumstances (e.g., suspension of classes, memorandum from UP officials).

Cheating

a. A student caught cheating during examinations/quizzes/problem sets/assignments will automatically be given a grade of 5.0 and the case will be forwarded to the Student Disciplinary Tribunal.
b. Always remember that “Cheating also means Quitting”.

What Constitutes a Violation of Academic Integrity?

- Turning in your friend’s code/write-up (obvious).
- Turning in solutions you found on Google with all the variable names changed (should be obvious). This is a copyright violation, in addition to an AI violation.
- Turning in solutions you found on Google with all the variable names changed and 2 lines added (should be obvious). This is also a copyright violation.
- LF commissioner. You may as well not submit the work since you will fail the exams and the course.
- Posting to forums asking someone to solve the problem. Note: Aggregating every [stack overflow answer/result from google| other source] because you "understand it" will likely result in full credit on assignments (if you aren't caught) and then failure on every exam. Exams don't test if you know how to use Google, but rather test your understanding (i.e., can you understand the problems to arrive at a solution on your own). Also, other students are likely doing the same thing and then you will be wondering why 10 people that you don’t know have your solution.

Adapted from: https://cse.buffalo.edu/~epmikida/teaching/sp23/cse250/resources/syllabus.pdf

Other violations that may not be as obvious

- Sending your code to a friend to help them. If another student uses/submits your code/solutions, you are also liable and will be punished.
- Joining a chatroom for the course where someone posts their code/solution once they finish, with the honor code that everyone needs to change it in order to use it.
- Reading your friend's code the night before it is due because you just need one more line to get everything working. It will most likely influence you directly or subconsciously to solve the problem identically, and your friend will also end up in trouble.

Adapted from: https://cse.buffalo.edu/~epmikida/teaching/sp23/cse250/resources/syllabus.pdf

What collaboration is allowed?

- Assignments in this course should be solved individually with only assistance from course staff and allowed resources. You may discuss and help one another with technical issues, such as how to get your compiler running, etc
- **Talking out how you eventually reached the solution from a high level is okay:** "I used a stack to store the data and then looked for the value to return."
- **but explaining every step in detail/pseudocode is not okay:** "I copied the file tutorial into my code at the start of the function, then created a stack and pushed all of the data onto the stack, and finished by popping the elements until the value is found and use a return statement."

The first example is OK but the second is basically a summary of your code and is not acceptable.

Adapted from: https://cse.buffalo.edu/~epmikida/teaching/sp23/cse250/resources/syllabus.pdf

Cheating

- You are expected to do your own work
- Sharing solutions, doing work for, or accepting work from others is cheating
- But you can learn with each other. You can discuss general solution strategies, but must write up the solutions yourself!
- If you discuss any problem with anyone else, you must write their name at the top of your assignment, labelling them as ”collaborators”

Using AI/chatgpt

- The use of AI tools must be properly documented and cited.
- The prompter of any produced output of AI, especially Generative AI, will be responsible for that output and be held accountable
- The use of AI tools is allowed for the following:
  - Finding information on a topic
  - Brainstorming and refining ideas
  - Drafting an outline to organize thoughts
  - Checking grammar and style
  - Summarizing or helping understand course materials
  - Cross-referencing information
  - Helping analyze, debug, or optimize user created code
- The use of AI tools is not allowed for the following:
  - Impersonating you or any other person in classroom contexts such as, but not limited to, using AI tools during in-person exams, class reporting, oral discussions
  - Writing entire code, scripts, sentences, paragraphs or papers to complete class assignments, machine problems, projects, and research papers.
- Specific class policies may be further imposed supplementing the above policies.

Copyright issues

a. You can't distribute any document that has a copyright.
b. Notice from University of the Philippines:
c. Note that the course pack provided to you in any form is intended only for your use in connection with the course that you are enrolled in. It is not for distribution or sale. Permission should be obtained from your instructor for any use other than for what it is intended.

Web Platform

- LMS
- Discord

CMSC 142

- Course Description?

- Design and Analysis of Algorithms

CMSC 142

- Prerequisites?

- CMSC 123

Student Learning Outcomes

- **CO1.** Learn good principles of algorithm design
- **CO2.** Analyse algorithm and estimate their worst-case and average-case behavior (in easy cases)
- **CO3.** Become familiar with fundamental data structures and identify how these data structures can best be implemented become accustomed to the s of algorithms in both functional and procedural styles
- **CO4.** Apply the theoretical knowledge in practice
- **CO5.** Synthesize an efficient algorithms in common engineering design situations

Class Outline

Introduction to Algorithms

- The role of algorithms in computing
- Defining algorithms
- Design and Analysis of Algorithms
- Growth of functions (asymptotic notation, standard notation and common functions)

Class Outline

Sorting and Searching

- Bogosort
- Stoogesort
- Insertion sort
- Selection sort
- Bubble sort
- Shell sort
- Merge sort
- Heapsort

Class Outline

Divide and Conquer algorithms

- Divide and Conquer Algorithms (ex. Counting inversions, finding the median)
- Recurrence relations

Class Outline

-- FIRST LONG EXAM --

Class Outline

Dynamic Programming

- Elements of dynamic programming
- Knapsack Algorithms

Class Outline

Greedy Algorithms

- Introduction to greedy algorithms
- Activity Selection
- Fractional knapsack
- Set cover
- Huffman Encoding

Class Outline

Graphs

- Introduction to graphs
- Graph representations
- Breadth fast search and depth first search algorithms

Class Outline

Minimum Spanning Tree

- Prim's Algorithm
- Kruskal's Algorithm

Class Outline

-- SECOND LONG EXAM --

Class Outline

Single source shortest path algorithms

- Single-source path in DAGs
- Dijkstra's algorithm
- Bellman-Ford Algorithm

Class Outline

Maximum Flow

- Flow networks
- Ford-Fulkerson method
- Maximum bipartite matching

Class Outline

-- THIRD LONG EXAM --

Class Outline

References

- Cormen, T. et al, Introductions to Algorithms 4th Ed., 2022
- Skiena, S., The Algorithms Design Manual 2nd Ed., 2008
- Dasgupta, S. et al, Algorithms, 2006

Class Policies

- A student caught cheating during examinations/quizzes will automatically be given a grade of **5.0** and the case will be forwarded to the Student Disciplinary Tribunal.
- During examinations, a student is only allowed to bring a pen unless specified otherwise.
- During discussions, the use of laptops and other gadgets is not allowed unless used to supplement instruction.

Class Policies

- Attendance will be recorded after a 5 min grace period.
- A student with 3 late records will be considered as 1 absent.
- A student who acquires at most 6 absences will be given a grade of DRP/5.0 depending on his/her current standing.

---- Class Begins ---

Problem 1

- A peasant finds himself on a riverbank with a wolf, a goat, and a head of cabbage. He needs to transport all three to the other side of the river in his boat. However, the boat has room for only the peasant himself and one other item (either the wolf, the goat, or the cabbage). In his absence, the wolf would eat the goat, and the goat would eat the cabbage.

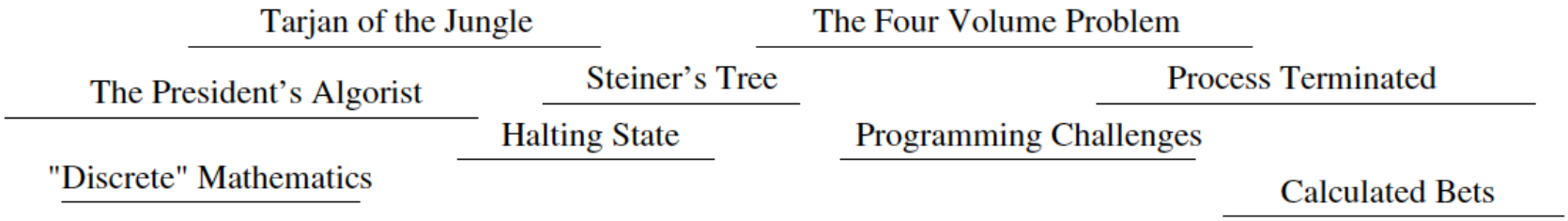
- Solve this problem for the peasant or prove it has no solution. (Note: The peasant is a vegetarian but does not like cabbage and hence can eat neither the goat nor the cabbage to help him solve the problem. And it goes without saying that the wolf is a protected species.)

Problem 2

- Imagine you are a highly-indemand actor, who has been presented with offers to star in n different movie projects under development. Each offer comes specified with the first and last day of filming. To take the job, you must commit to being available throughout this entire period. Thus you cannot simultaneously accept two jobs whose intervals overlap.
- For an artist such as yourself, the criteria for job acceptance is clear: you want to make as much money as possible. Because each of these films pays the same fee per film, this implies you seek the largest possible set of jobs (intervals) such that no two of them conflict with each other.

Problem

- For example, consider the available projects in We can star in at most four films, namely “Discrete” Mathematics, Programming Challenges, Calculated Bets, and one of either Halting State or Steiner’s Tree.



- You (or your agent) must solve the following algorithmic scheduling problem.

A Strategy: Earliest Job First

EarliestJobFirst(I)

Accept the earlest starting job j from I which does not overlap any previously accepted job, and repeat until no more such jobs remain.

War and Peace

_____ _____ _____ _____ _____

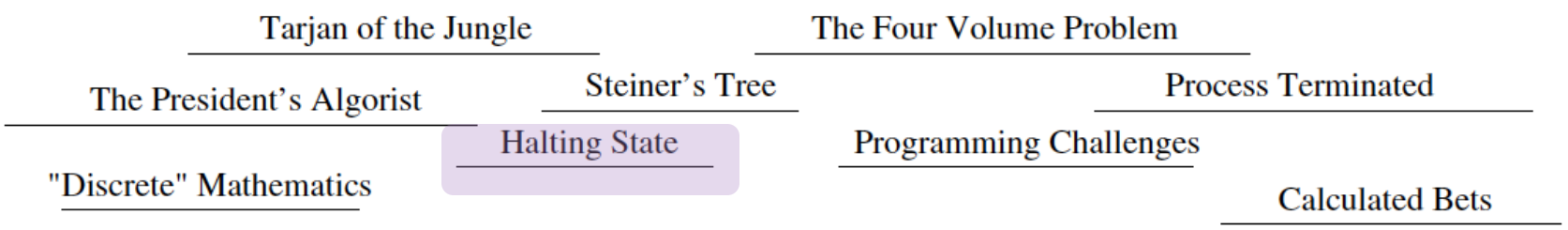
Another strategy: Shortest Job First

ShortestJobFirst(I)

While $(I \neq \emptyset)$ do

Accept the shortest possible job j from I.

Delete j, and any interval which intersects j from I.



An Optimal Strategy

OptimalScheduling(I)

While $(I \neq \emptyset)$ do

Accept the job j from I with the earliest completion date.

Delete j, and any interval which intersects j from I.

- Sort intervals by **end time**.
- Select the first movie that finishes earliest.
- Keep selecting the next earliest finishing movie that **does not overlap** with the previously selected one.

You were given a **problem**

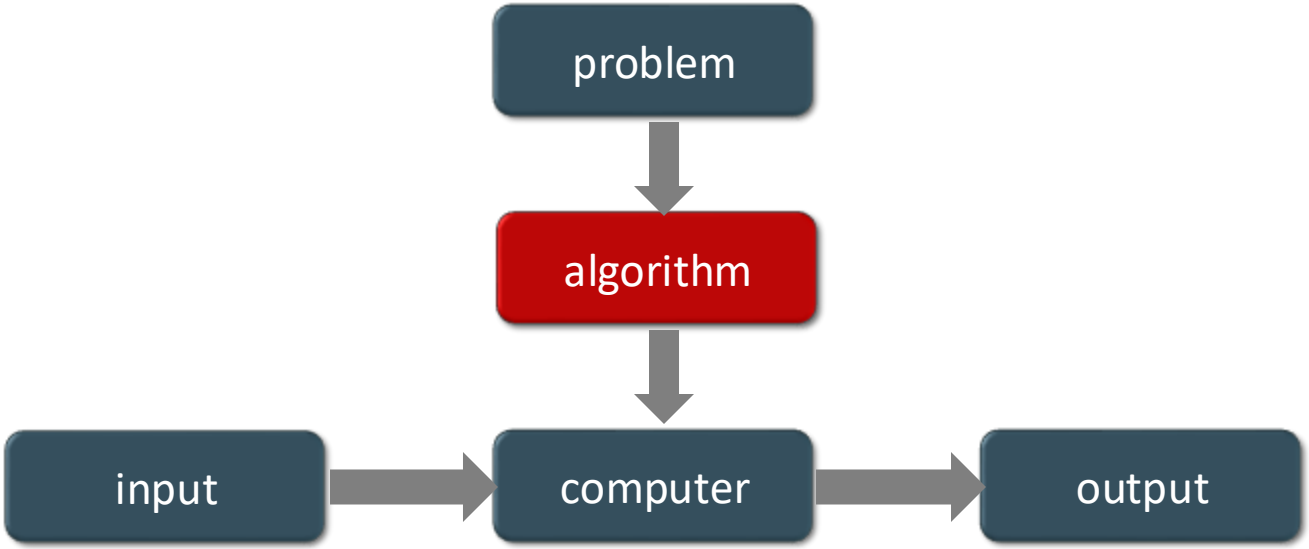
We were also given a set of **inputs** (jobs with their starting and completion times)

Our desired **output** are the set of movies that we can take as an actor

What did we do given these inputs and outputs?

Algorithms

- Sequence of computational steps that transform input into output



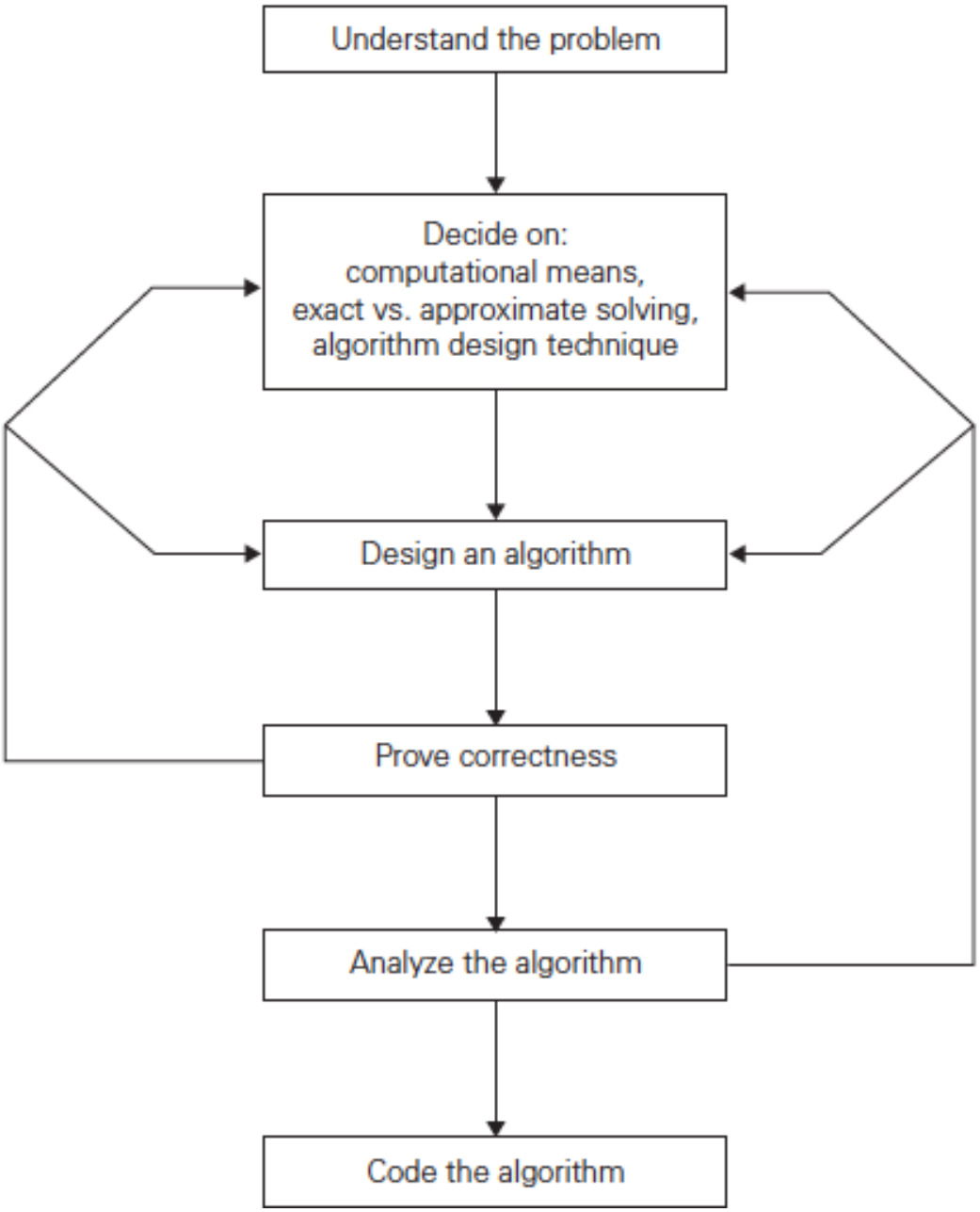
Algorithms

Heart of Computer Science

We explored multiple algorithms

Why do we have to study algorithms?

- Challenging: good for the brain
- Makes you a better programmer
- It is easy to write a (naive) algorithm, but cleverness is needed to make fast and efficient algorithms


Understand the problem
Decide on:
computational means,
exact vs. approximate solving,
algorithm design technique
Design an algorithm
Prove correctness
Analyze the algorithm
Code the algorithm

Design of Algorithms

- Design is a creative process
- Don't necessarily have to invent new algorithm when designing
- Understand **which algorithm design techniques fit the problem** at hand the best

Analysis of Algorithms

- The theoretical study of computer program performance (running time) and resource usage (memory)
- We study algorithms with respect to performance because:
  - It helps us understand **scalability**
  - It often draws the line between **feasible** and **impossible**
  - It provides a vocabulary for talking about **program behavior** (e.g. **fast**, **slow**)
  - It often generalizes to other **computer resources** (e.g. too much time = too much bandwidth)
  - It is the currency of computing (**tradeoffs**)
- Some factors that could be more important than performance: correctness, maintainability, functionality, user-friendliness, programmer time, simplicity, extensibility, reliability

When designing an algorithm

- Is it correct? (Does it solve the problem)
- How long does it take? Is it reasonable? (Analysis)
- Can we do better?

Properties of an Algorithm

1. **INPUT:** The Algorithm should be given zero or more input.

2. **OUTPUT:** At least one quantity is produced. For each input the algorithm produced value from specific task.

3. **DEFINITENESS:** Each instruction is clear and unambiguous.

4. **FINITENESS:** If we trace out the instructions of an algorithm, then for all cases, the algorithm terminates after a finite number of steps.

5. **EFFECTIVENESS:** Every instruction must very basic so that it can be carried out, in principle, by a person using only pencil & paper.

How can we describe an algorithm?

An algorithm can be described in the following ways:

1. Natural language like English
2. Graphic representation called a flowchart
3. Pseudo-code Method
4. Programming Language

How to Write an Algorithm

Step-1:start
Step-2:Read a,b,c
Step-3:if a>b
if a>c
print a is largest
else
if b>c
print b is largest
else
print c is largest
Step-4 : stop

Step-1: start
Step-2: Read a,b,c
Step-3:if a>b then go to step 4
otherwise go to step 5
Step-4:if a>c then
print a is largest otherwise
print c is largest
Step-5: if b>c then
print b is largest otherwise
print c is largest
step-6: stop

We often confuse programs with algorithms.

What's the difference between the two?

Algorithm vs Program

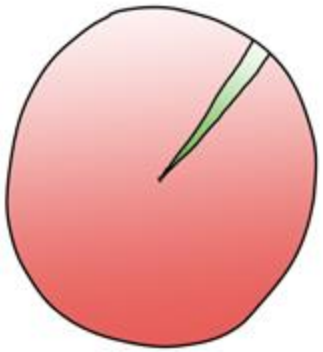
Algorithm

1. At design phase
2. Natural language
3. Person should have domain knowledge
4. Analyze

Program

1. At implementation phase
2. Written in any programming language
3. Programmer
4. Testing

THE USAGE OF PSEUDOCODE IN REAL LIFE



- DESCRIBING AN ALGORITHM
- A TOOL THAT FRESHMAN COMPUTER SCIENCE STUDENTS THAT JUST STARTED TO LEARN PROGRAMMING USES TO EXPRESS THEIR DUMB ACTIONS

Pseudo-Code Conventions

1. Comments begin with **//** and continue until the end of line.
2. Blocks are indicated with matching braces { and }.
3. An identifier begins with a letter. The data types of variables are not explicitly declared.
4. There are two Boolean values **TRUE** and **FALSE**.

Logical Operators

AND, OR, NOT

Relational Operators

<, <=,>,>=, =, !=

5. Assignment of values to variables is done using the assignment statement.

<Variable>:= <expression>;

7. The following looping statements are employed.

For, while and repeat-until While Loop:

```
while < condition > do
 {
     <statement-1>
          ..
          ..
   <statement-n>
 }
```

For Loop:

```
for variable: = value-1 to value-2 step do
{
        <statement-1>
                    .
                    .
                    .
        <statement-n>
}
```

8. A conditional statement has the following forms.

if <condition> then <statement>

if <condition> then <statement-1>
else <statement-1>

Example

Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

Pseudocode & Algorithm

Pseudocode:

- *Input a set of 4 marks*
- *Calculate their average by summing and dividing by 4*
- *if average is below 50*

  Print “FAIL”

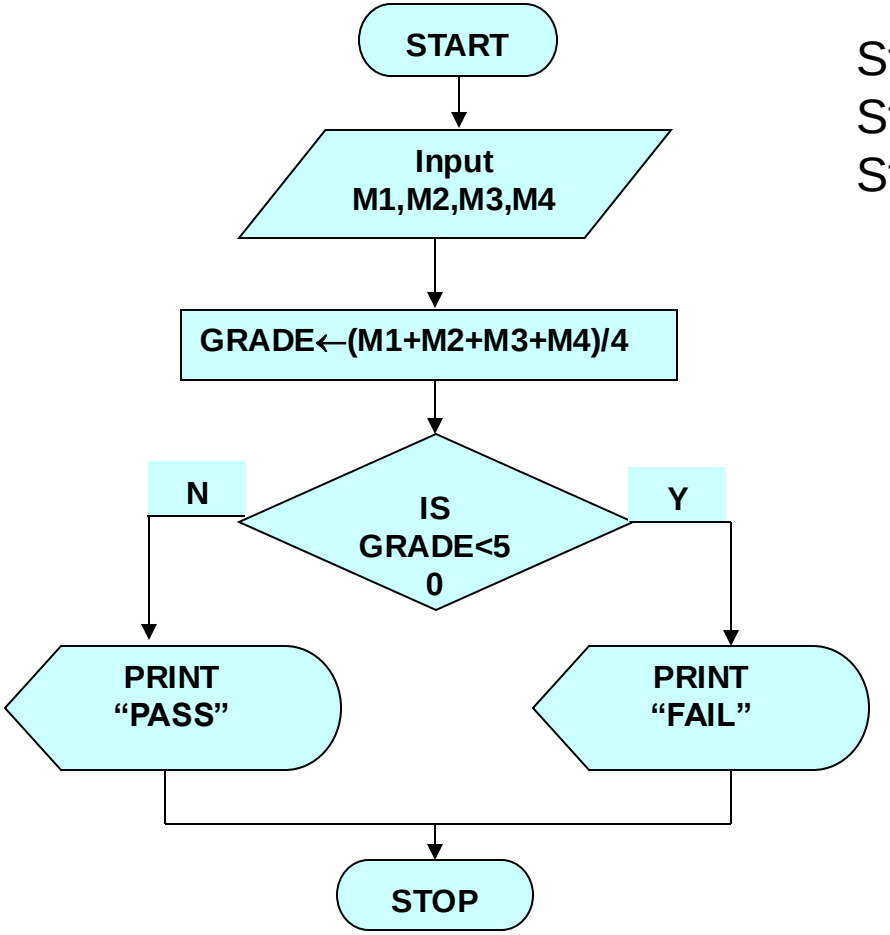
  else

  Print “PASS”

Pseudocode & Algorithm

- Detailed Algorithm
- Step 1: Input M1,M2,M3,M4
  Step 2: GRADE ← (M1+M2+M3+M4)/4
  Step 3: if (GRADE < 50) then
  Print “FAIL”
  else
  Print “PASS”
  endif

Example



Step 1: Input M1,M2,M3,M4
Step 2: GRADE ← (M1+M2+M3+M4)/4
Step 3: if (GRADE <50) then
Print “FAIL”
else
Print “PASS”
endif

Okay… going back…

Activity: Design an algorithm

Group Activity

- Form a group composed of 3 members
- Write your algorithms in a document file and convert it into pdf. Submit your files through the classroom on or before class time. No more extensions.
- Algorithms should be written in two ways:
  - (a) Pseudocode
  - (b) List of steps

3 Problems

- Find the nth number in the Fibonacci Sequence
- Find the GCD of two numbers