



CMSC 11: Introduction to Computer Science

Jaderick P. Pabico <jppabico@uplb.edu.ph>
Institute of Computer Science, CAS, UPLB

Get ready for a quiz



- On a ½ sheet of paper write:
 - Your name
 - Today's date (21 February 2008)
 - CMSC 11
 - Lab Section
- UP System's rules on academic honesty is in effect!

Quiz



- Explain what the following assembly code do

JZ 321



Quiz Answer

- Explain what the following assembly code do

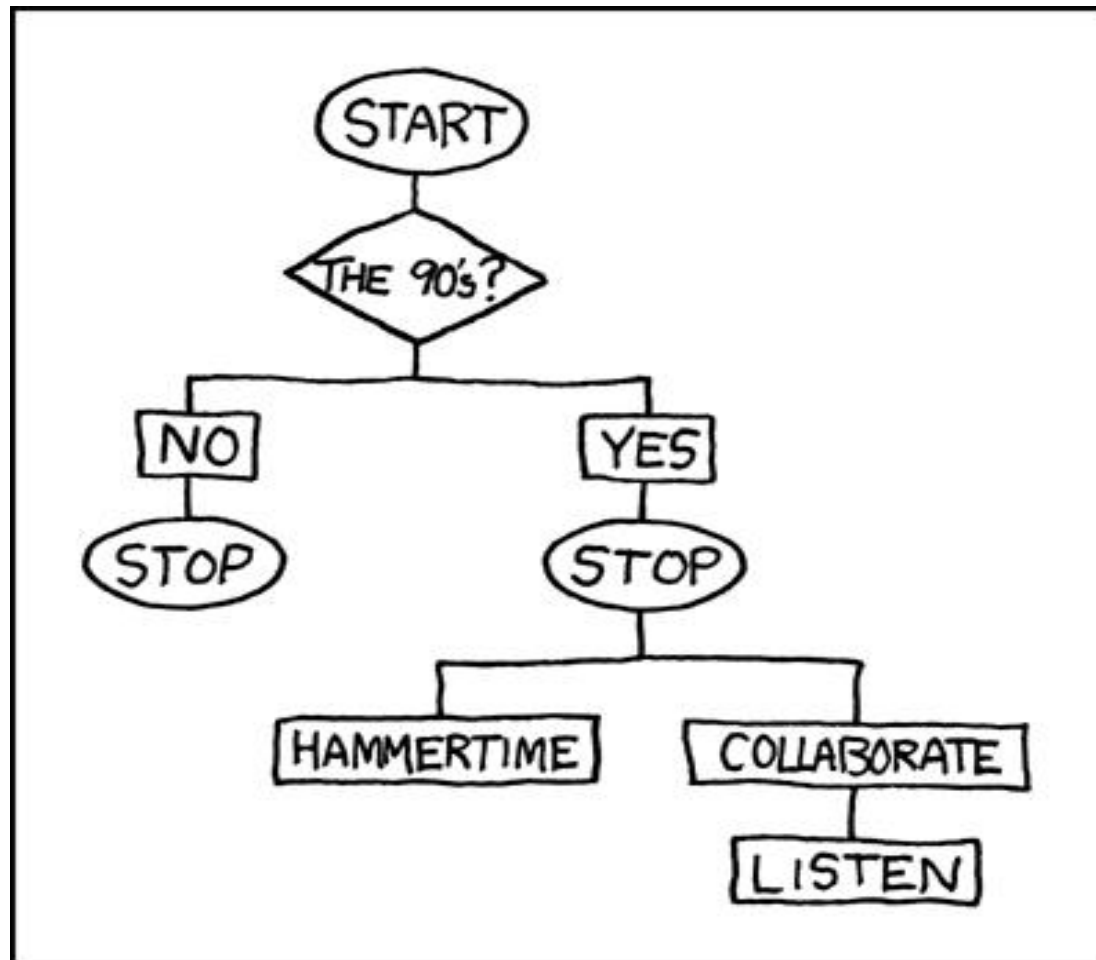
JZ 321

- If accumulator is zero, control writes 321 to program counter, else control increments program counter by 1.

What's up today?



- Flowcharts



Flowchart

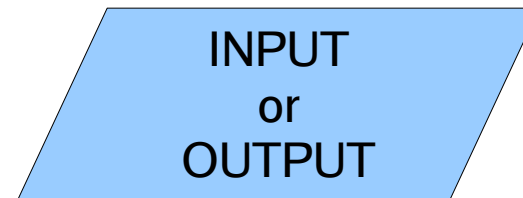
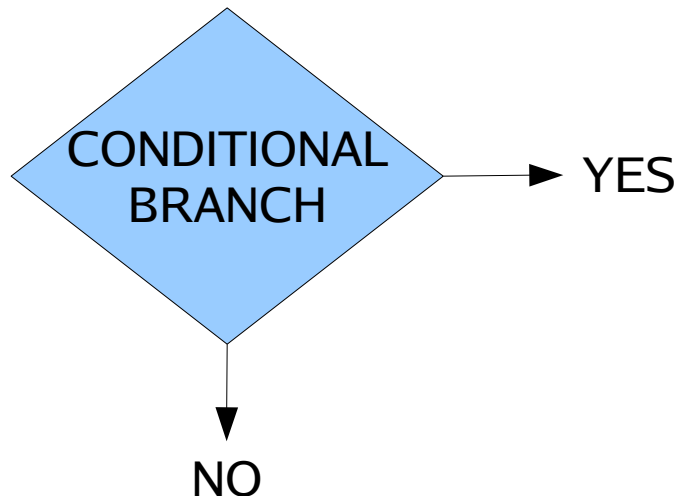
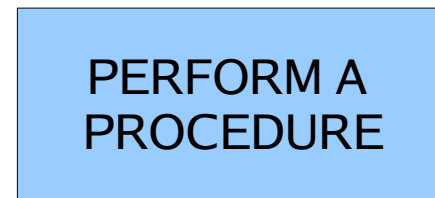


- Some standard symbols are used to make algorithms easier to follow
- Each step is represented by a specially shaped box
- The shape indicates what type of step is to be executed.
- There are four possibilities...

Flowchart




- Here are the four possibilities:



Flowchart



- The flow of the algorithm is represented by arrows 
- And when all the symbols are combined, it's a flow chart

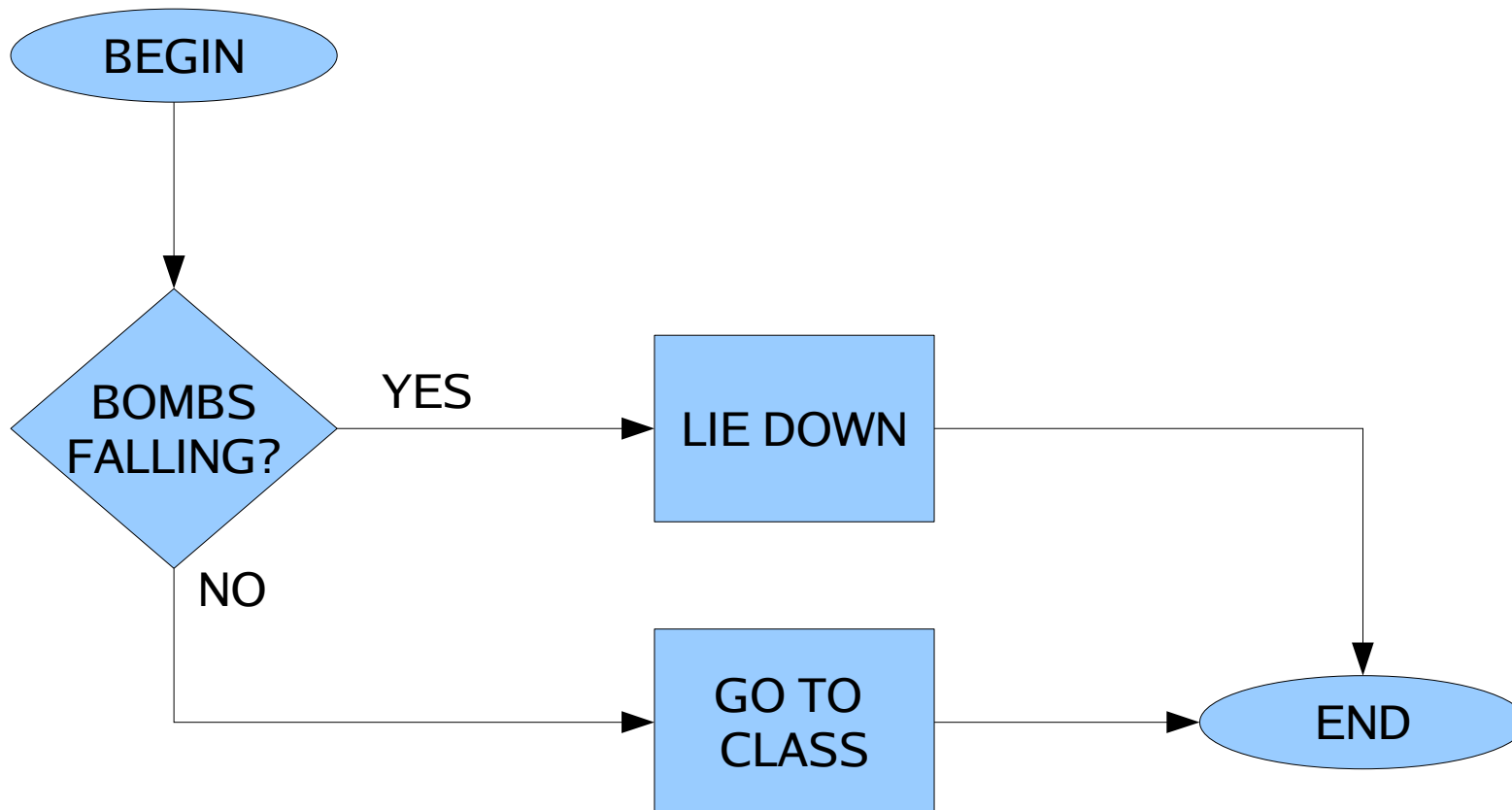


Go with the flow!

Flowchart



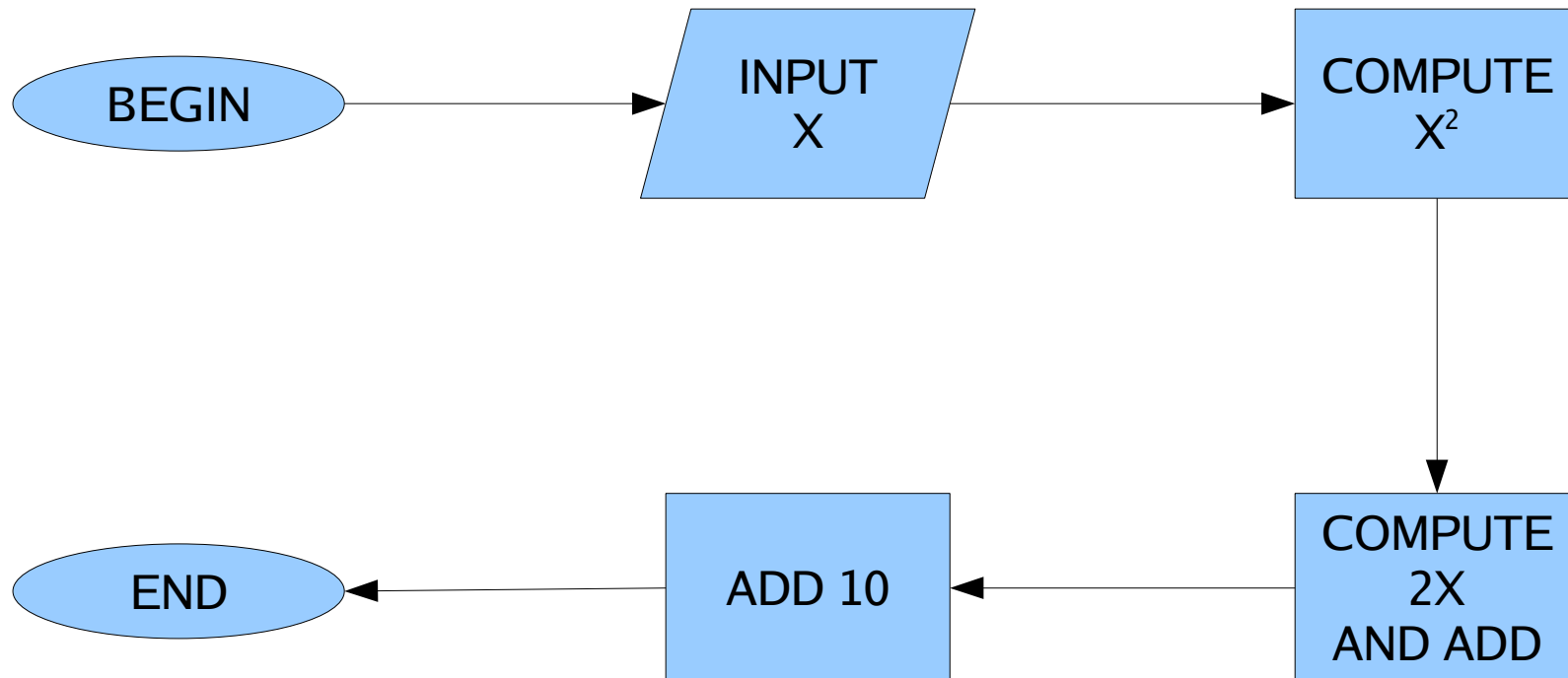
- Here is the flow chart of the falling bombs example several meetings ago:



Flowchart



- Here is another one:



Flowchart



- One common thing about the two algorithms (flowcharts) is that...
- the flow proceeds in one direction...
- from start to finish.
- However, it is also possible for the flow of algorithms to jump forward or backward.
- How?



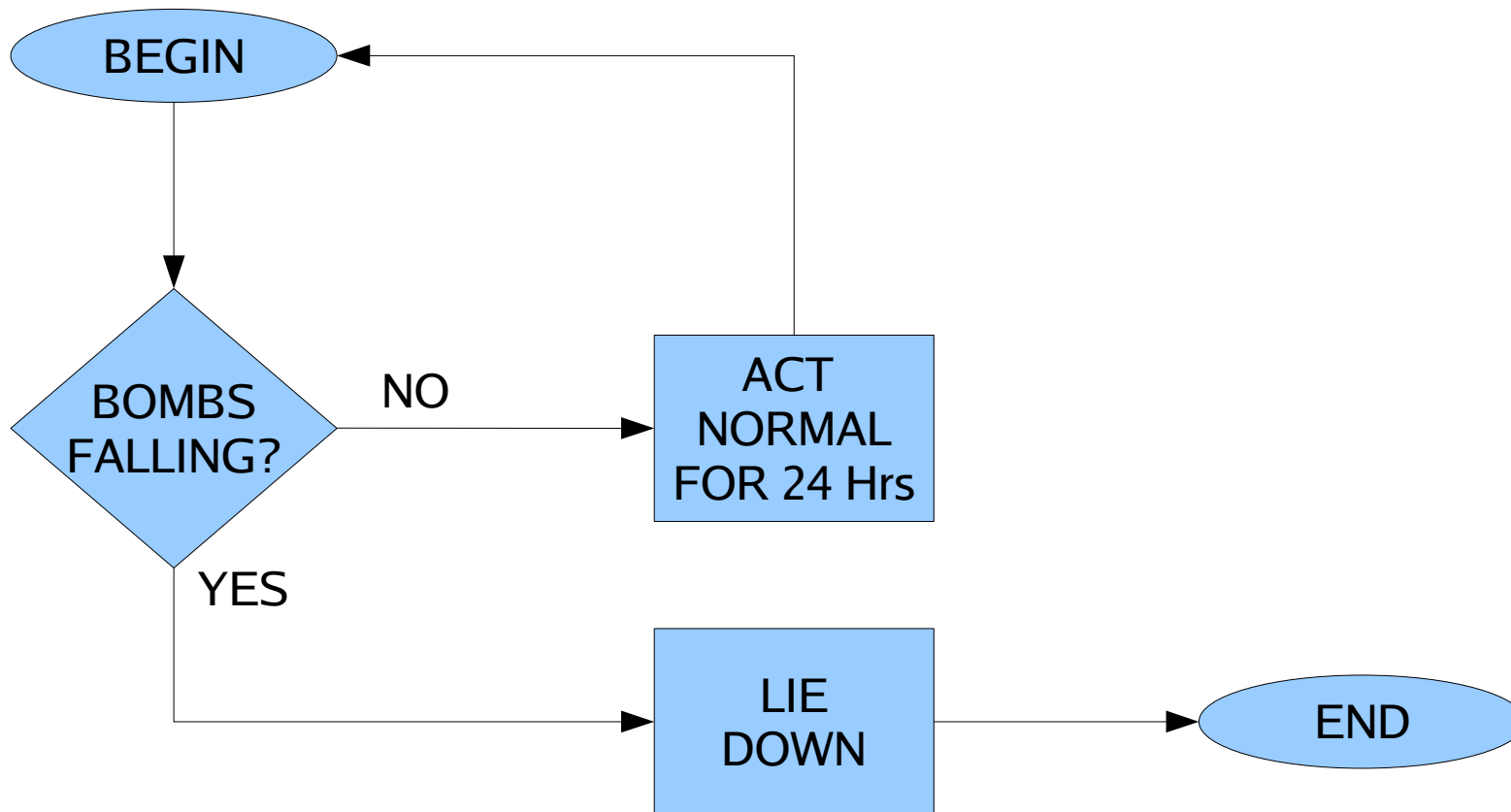


Flowchart

- Let's have an example:
- Let's rewrite the falling bombs algorithm:
 - 1.If bombs are falling then go to step 2 else go to step 4
 - 2.Lie down and enjoy
 - 3.Go to step 6
 - 4.Lead a normal life for 24 hours
 - 5.Go to step 1
 - 6.End



Flowchart

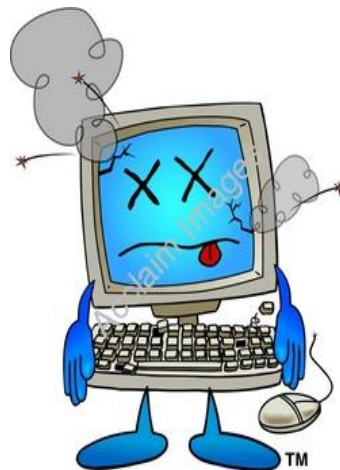


- You may find the flow chart easier to grasp than the written program.



Flowchart

- Flow charts are useful in helping to design algorithms
- Simple ones anyway
- And designing algorithms is what computer programming is all about!



FEED ME
ALGORITHMS



Flowchart

- The first step in writing any program is to ANALYZE the job to be done
- ... and see how to do it algorithmitically!
- Failure to think algorithmically has caused many software nightmares!



Flowchart



- Most software designers have horror stories about customers who didn't know EXACTLY what they wanted!

Yes.. that info should go into my files... or maybe not... the VPs are just as good... or maybe the treasurer's



Argh...



Flowchart

- Let's try a couple more examples...
- A little more like what a computer might actually asked to do

1. THE HOUSEMATE RECEIPTS PROBLEM

2. MULTIPLE PLUG-INS

Housemate Receipts



- Two housemates, Joey and Chandler, share their meals
- They both shop for food and save their receipts
- At the end of the month, they want to know who owes whom by how much.

Joey

Chandler

Ross' foot





Housemate Receipts

- Let's make the flow chart.
- We start by reasoning like so:
 - Let C = Chandler's expenses
 - Let J = Joey's expenses
 - Then the total expense is $C + J$
 - Each housemate's share is $(C + J) / 2$
- However, that is assuming that both consume the same amount of food.



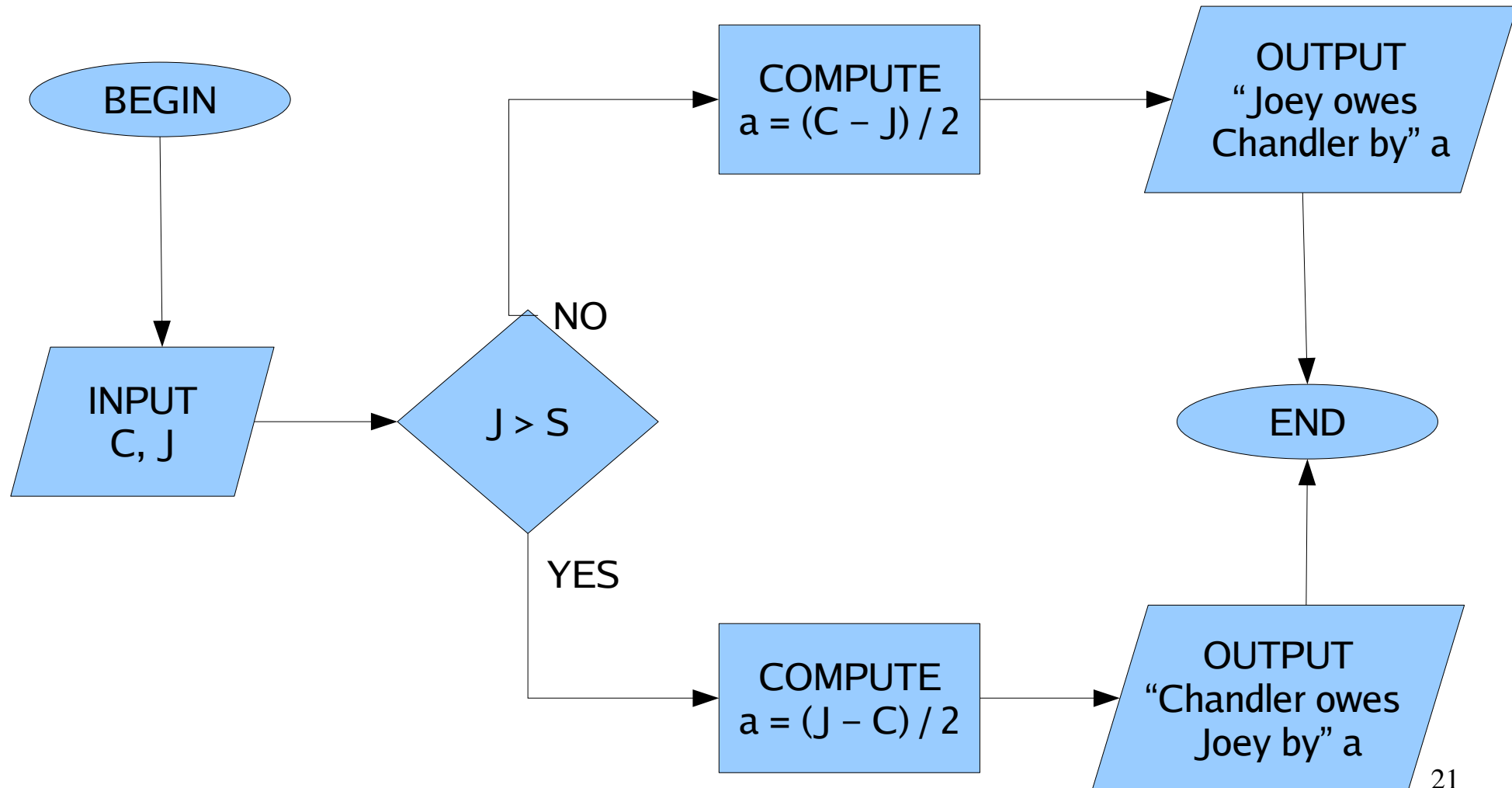
Housemate Receipts

- If Chandler outspent Joey, then $C > J$
 - And Joey owes Chandler $(C + J) / 2 - J$
 - Or $(J - C) / 2$
- Otherwise (Joey outspent Chandler)
- Or Joey owes Chandler $(C - J) / 2$
- The algorithm's output is to tell us who owes whom and how much.



Housemate Receipts

- Here it is:





Multiple Plug-ins

- We want to evaluate a single expression
- $x^2 + 2x + 10$
- We want to evaluate it repeatedly at different values of x
- Namely at $x = 0.0, 0.1, 0.2, \dots, 1.9, 2.0$



O, wow, this is heavy stuff!



Multiple Plug-ins

- The core of the algorithm will be this loop:

1. Plug the current value of x into $x^2 + 2x + 10$

2. Print the result

3. Next x

4. Return to step 1

- We also have to specify

- What x to start with
- When to stop
- And how to compute “next x ”

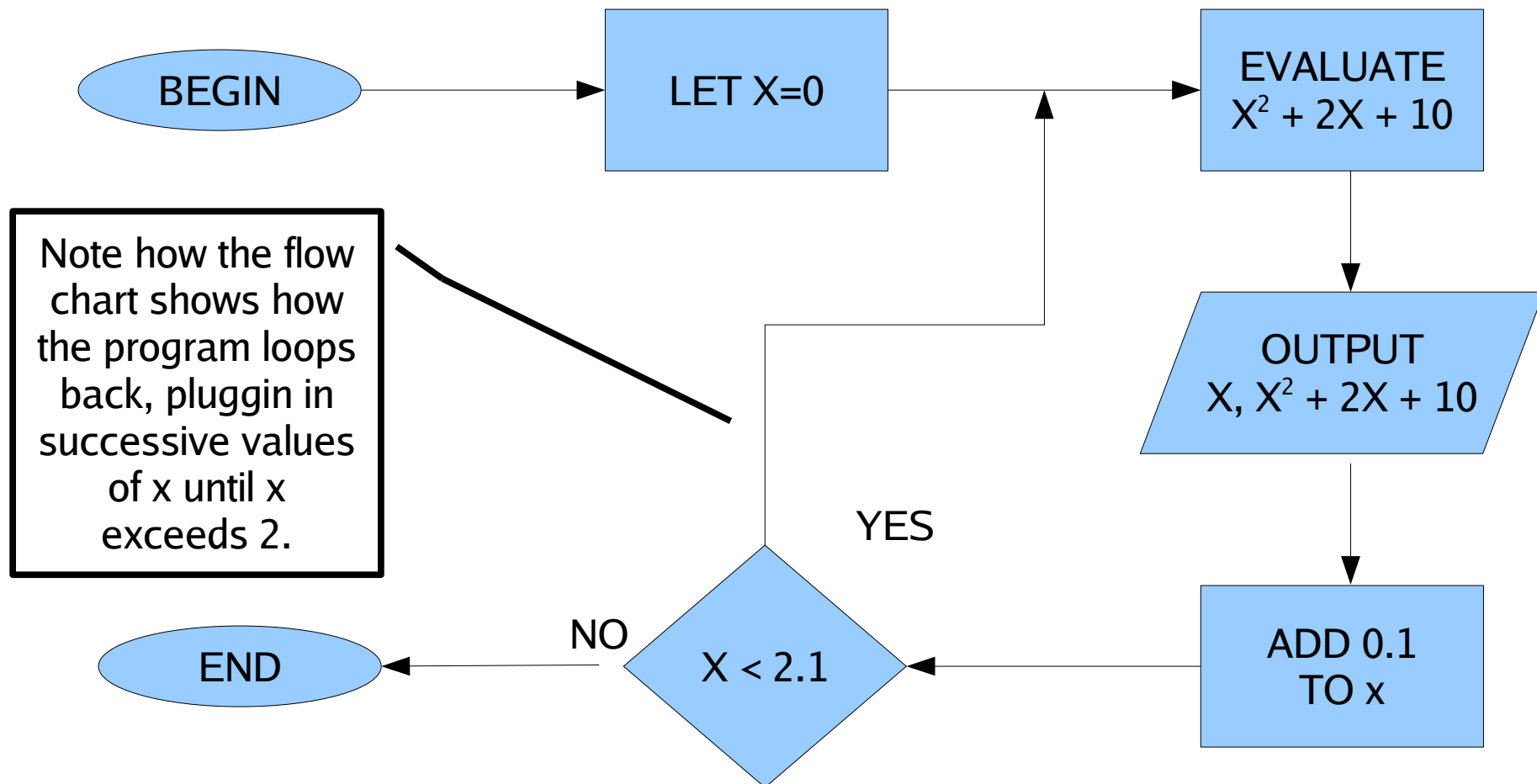
Now, this is really, really heavy stuff!





Multiple Plug-ins

- Here is the flow chart:



NOW?



- ... the \$1Million question (\$236K after tax):

How do you write an algorithm that is intelligible to a computer?



- IOW, how do you program a computer?

NOW?



- Unfortunately, you have to speak the computer's language
- Because the computer is still too stupid to understand yours!

It may be fast, honey, baby,
but it's thick!



- What language does the computer understand?

How?



- In the very beginning, programmers wrote directly in machine language – the binary code
- This was obviously a headache!



You'll need a computer just to figure out our headache drug cost!

How?



- Soon, they switched to assembly language,
- aided by automatic assemblers
- Which translated assembly language mnemonics into machine code
- Still something more is needed...

You'll get to read a thick source code just to debug a tic-tac-toe program in assembly!



How?



- And finally, the HIGHER-LEVEL programming languages were invented
- These contain familiar English-like commands such as “print”, “read”, and “do”
- Which are translated into machine language by complex programs called compilers or interpreters

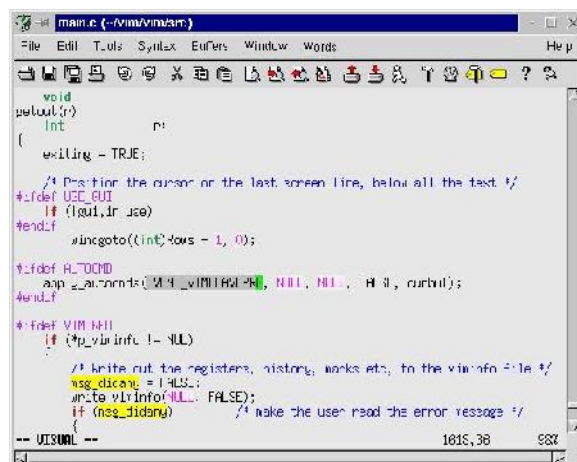


How?



- Higher-level programs are sometimes called “source code”
- And the machine-language translation is called “object code”

SOURCE CODE



OBJECT CODE

