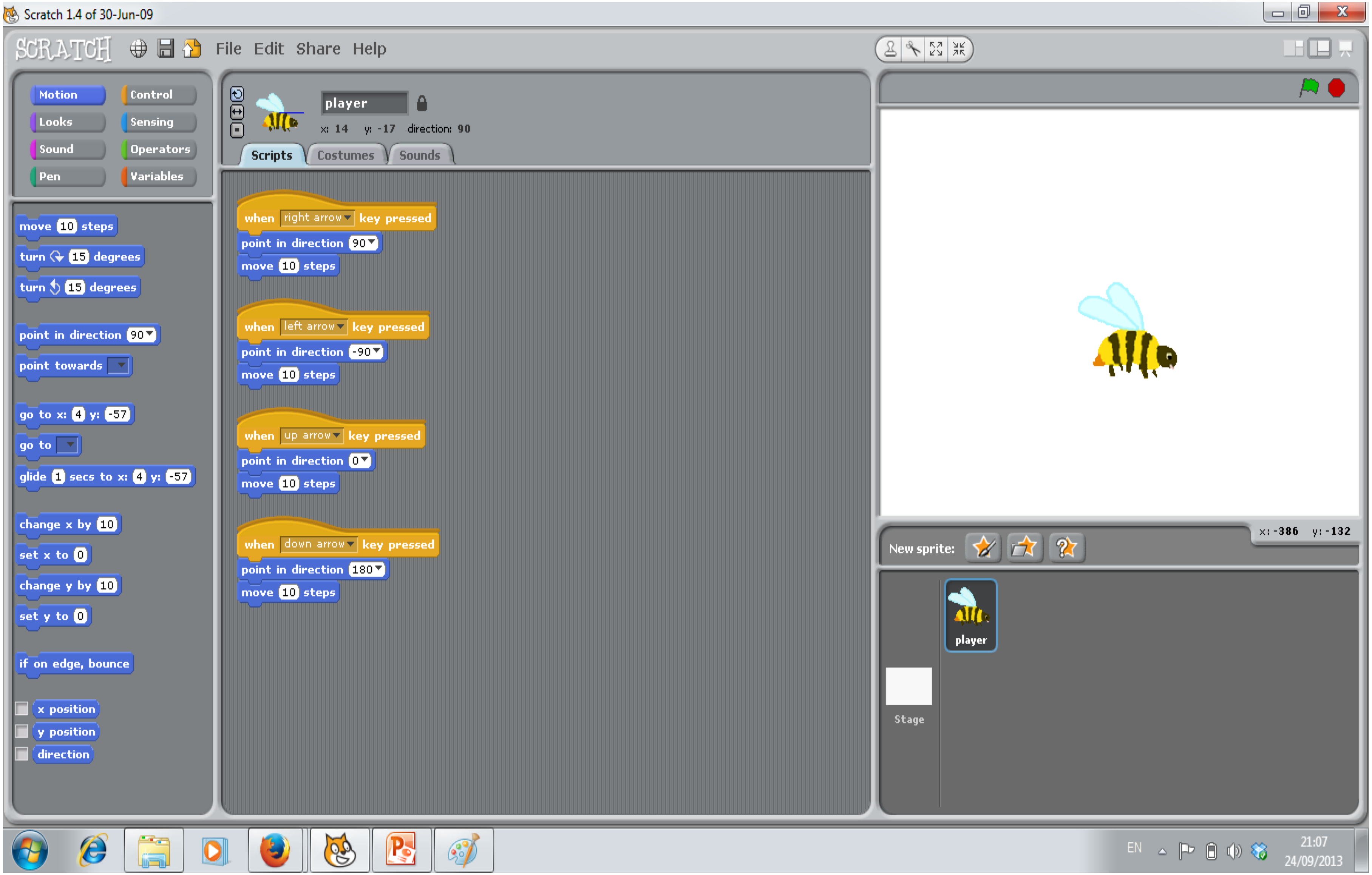
6. Flowcharts 1

What did we do last time?

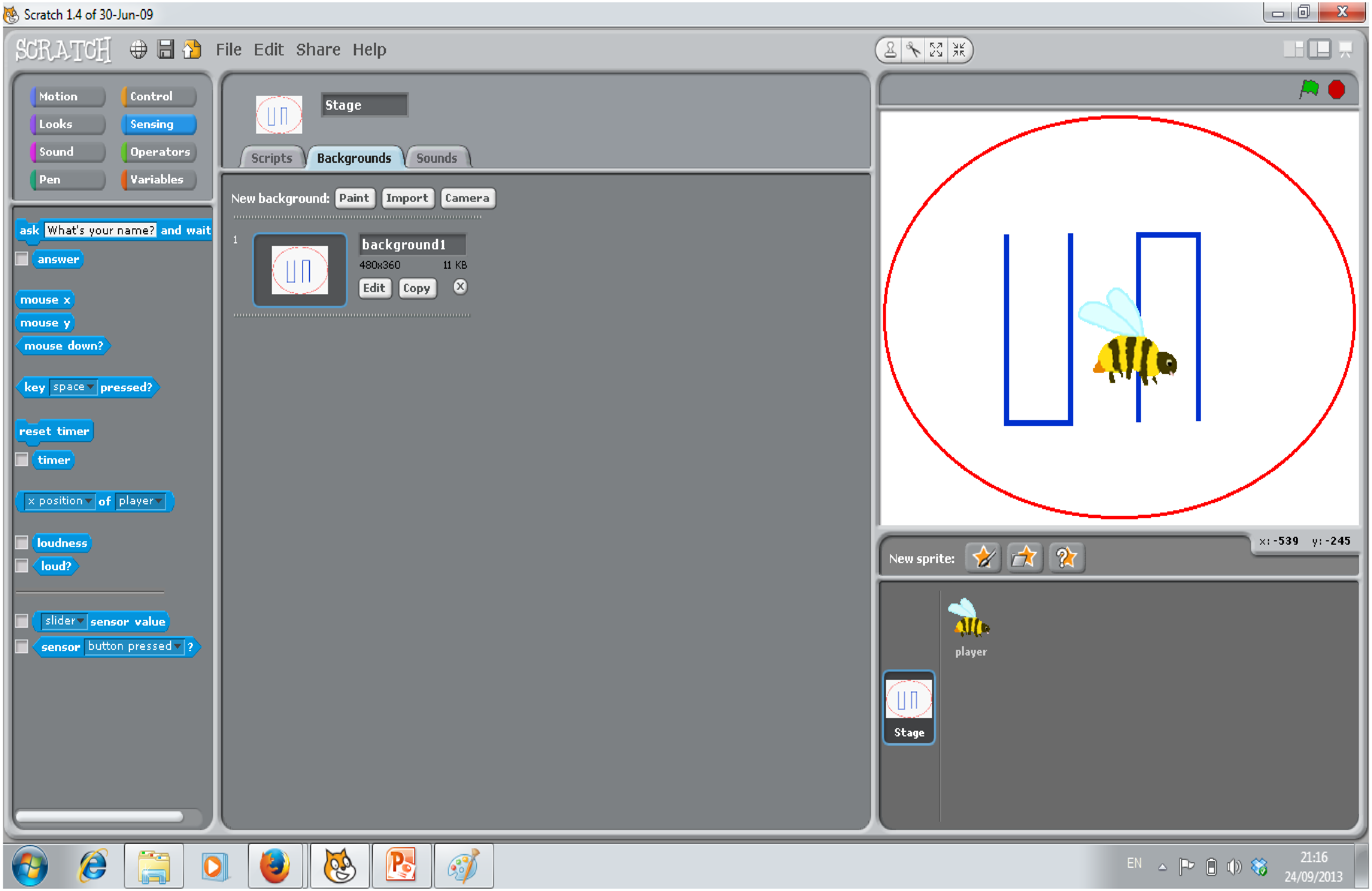
Using SCRATCH to build a maze game

Step 1: Build our player

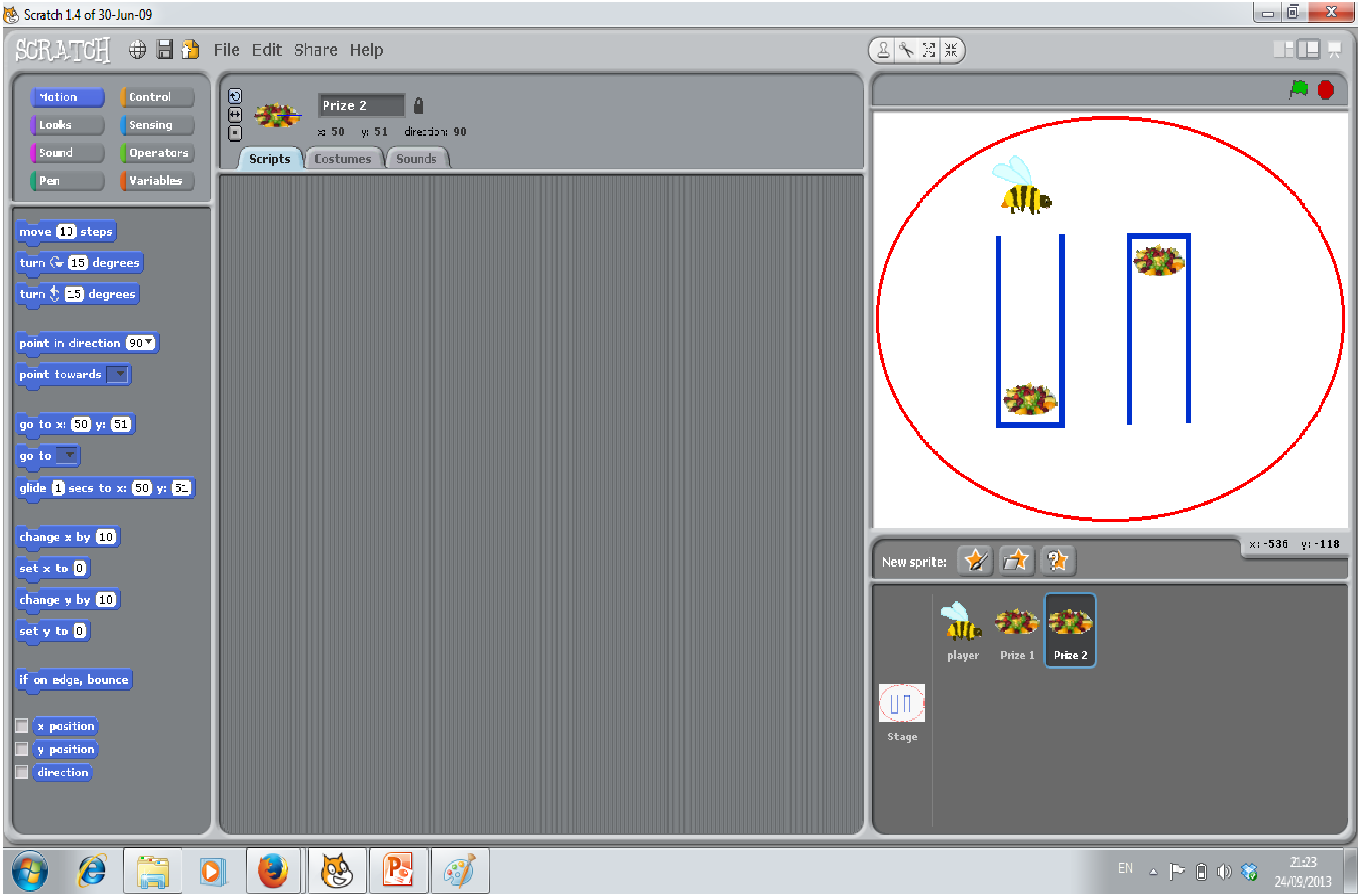


```
when right arrow key pressed
point in direction 90
move 10 steps

when left arrow key pressed
point in direction -90
move 10 steps

when up arrow key pressed
point in direction 0
move 10 steps

when down arrow key pressed
point in direction 180
move 10 steps
```

Step 2: Draw the stage



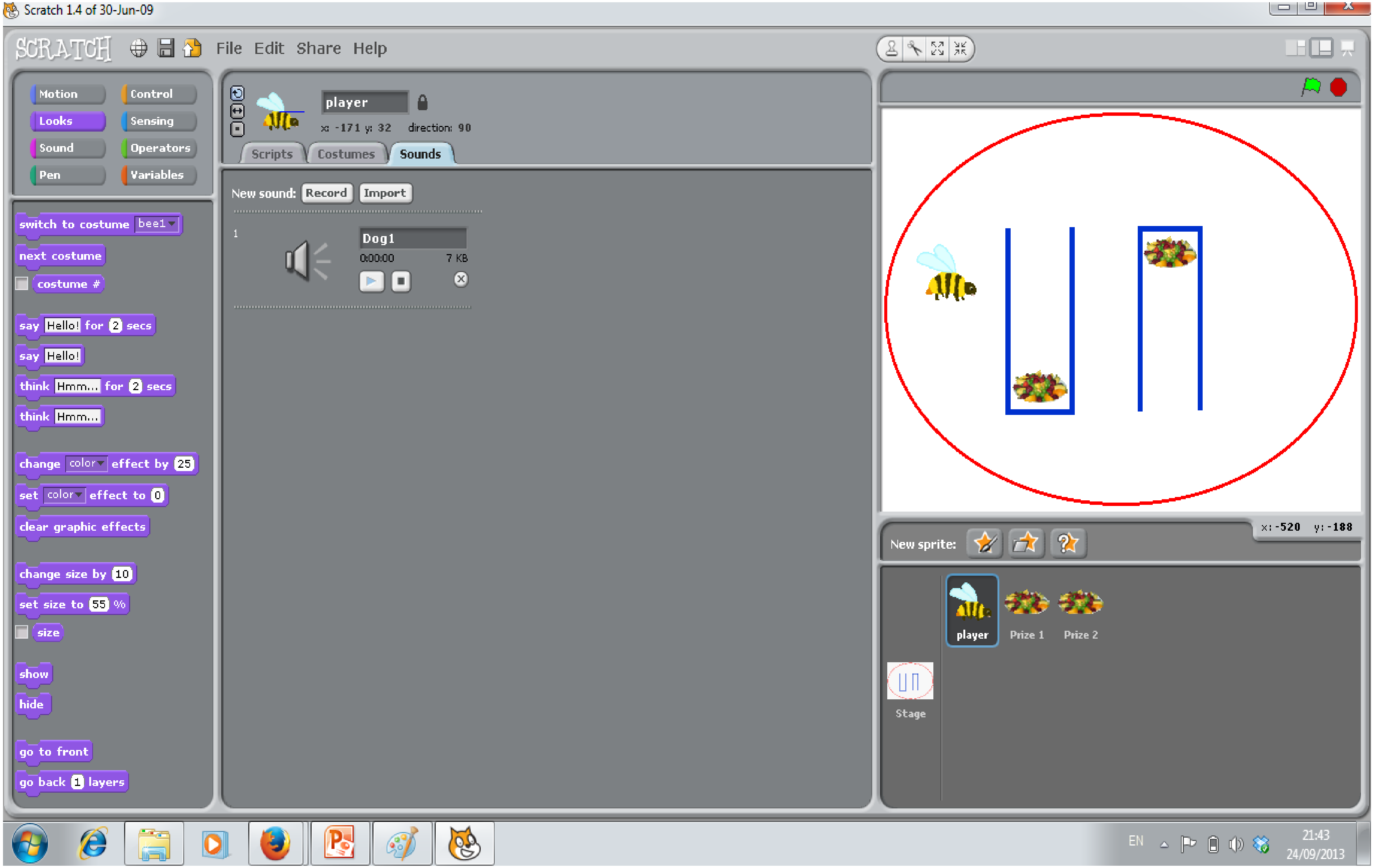
Step 3: Create some prizes



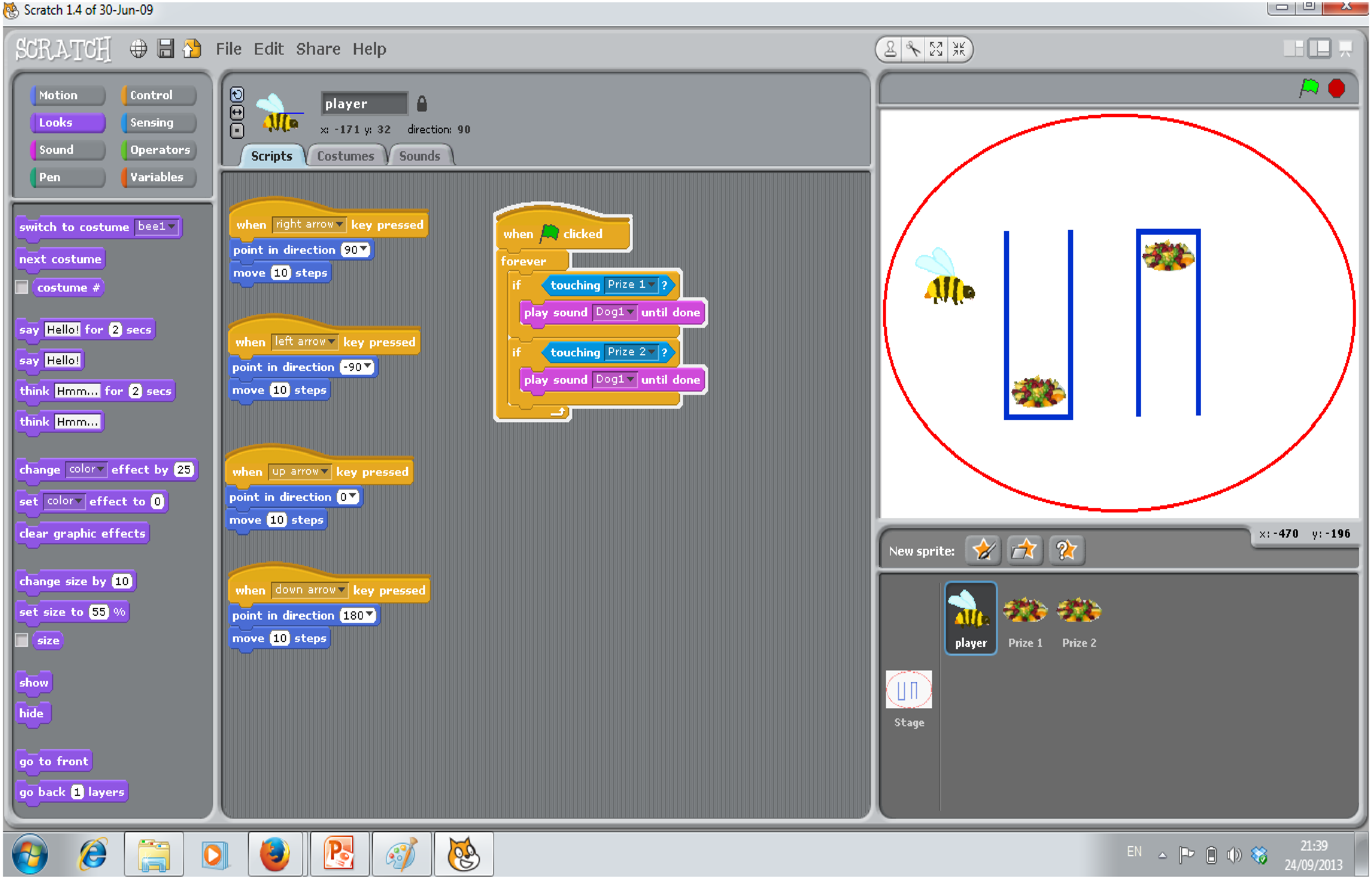
Prize 2

x: 50 y: 51 direction: 90

Step 4: Add sound to the player



Step 5: Program the player

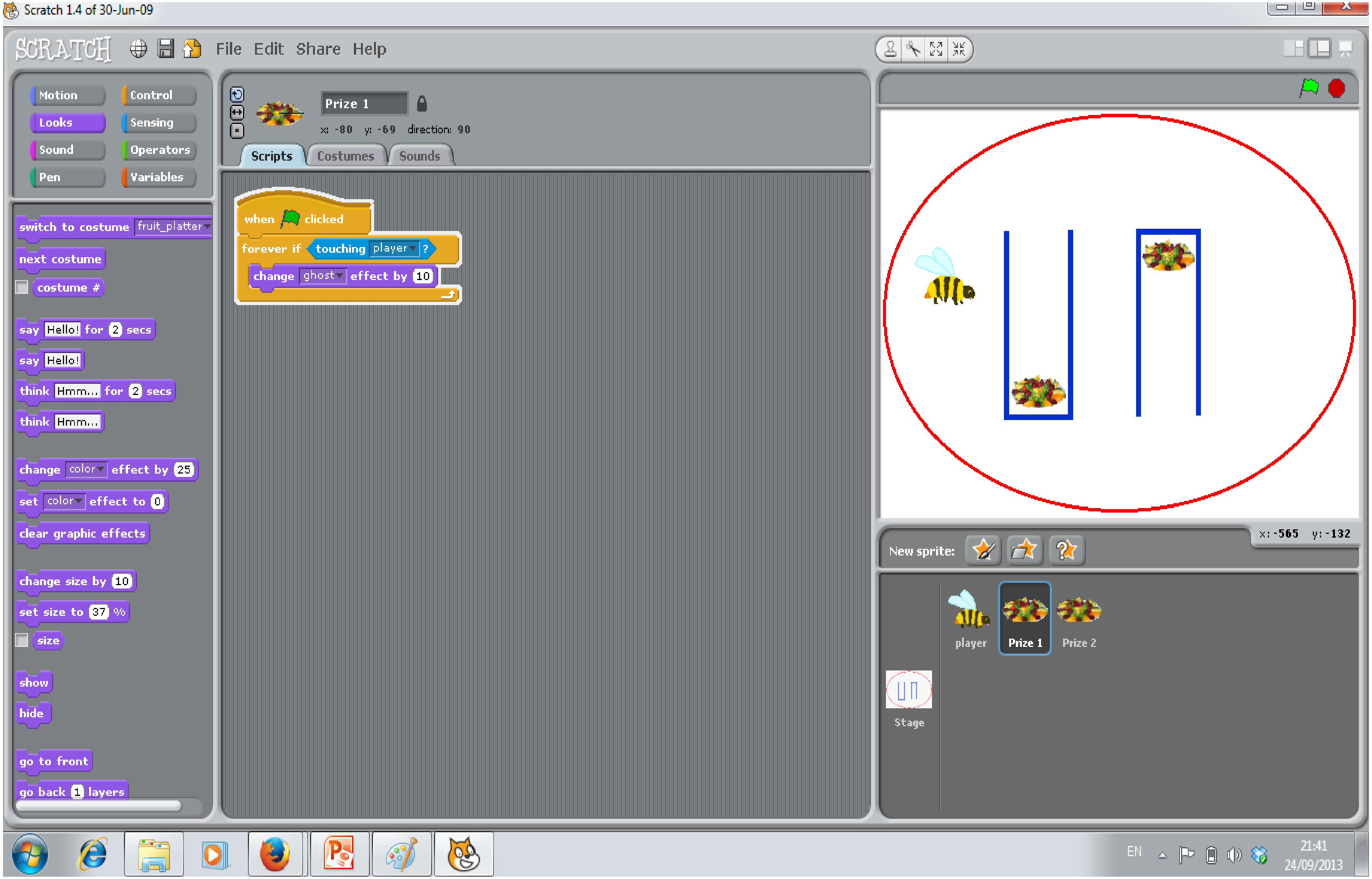


player

x: -171 y: 32 direction: 90

```
when right arrow key pressed
point in direction 90
move 10 steps

when left arrow key pressed
point in direction -90
move 10 steps

when up arrow key pressed
point in direction 0
move 10 steps

when down arrow key pressed
point in direction 180
move 10 steps

when ⚑ clicked
forever
    if touching Prize 1 ?
        play sound Dog1 until done
    if touching Prize 2 ?
        play sound Dog1 until done
```

Step 6: Program the prizes



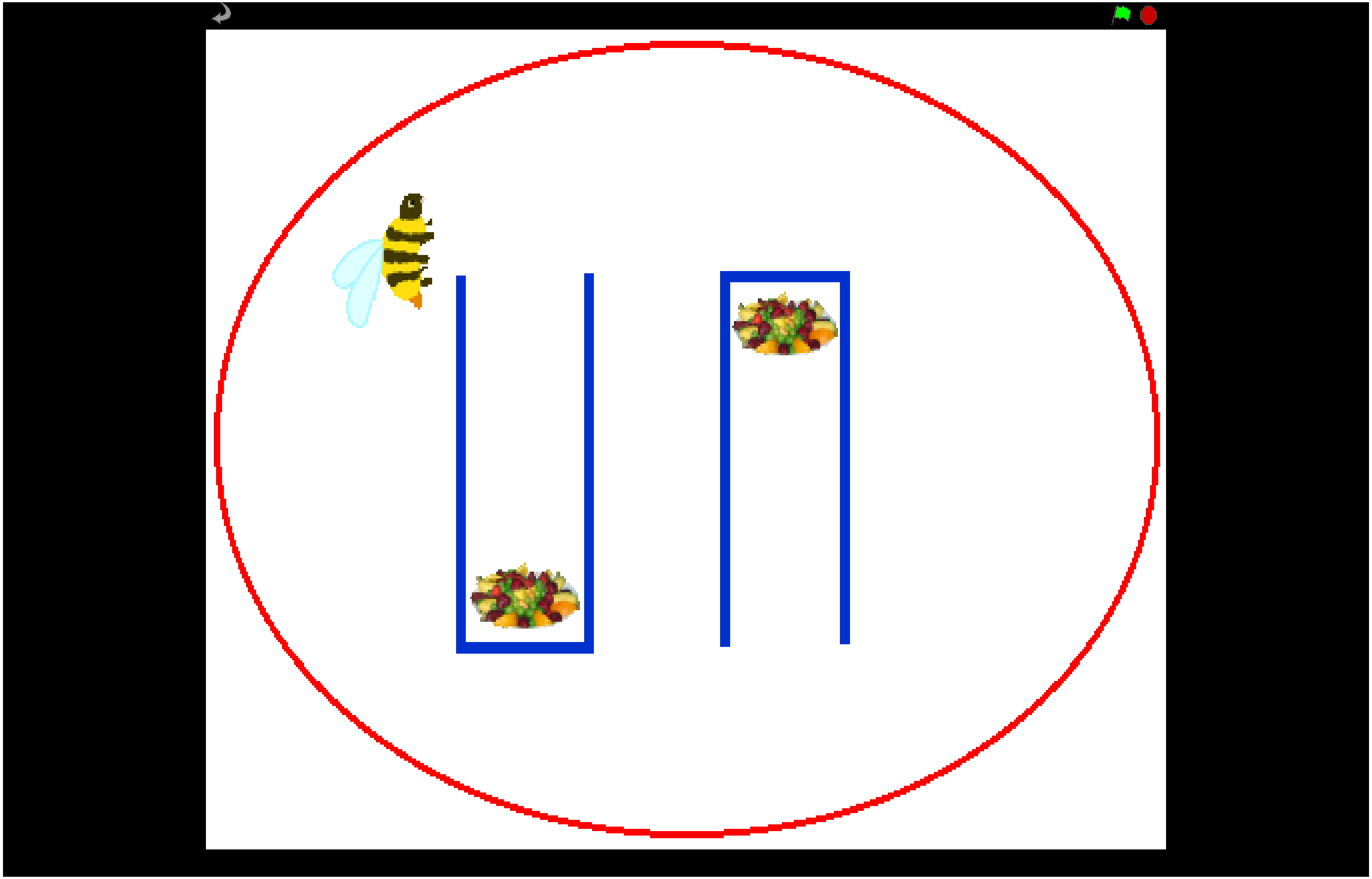
Scratch 1.4 of 30-Jun-09

Prize 1

x: -80 y: -69 direction: 90

```
when 🟩 clicked
forever if touching player ?
    change ghost effect by 10
```

Step 7: Game completed!



Flowcharts

Why flowcharting?

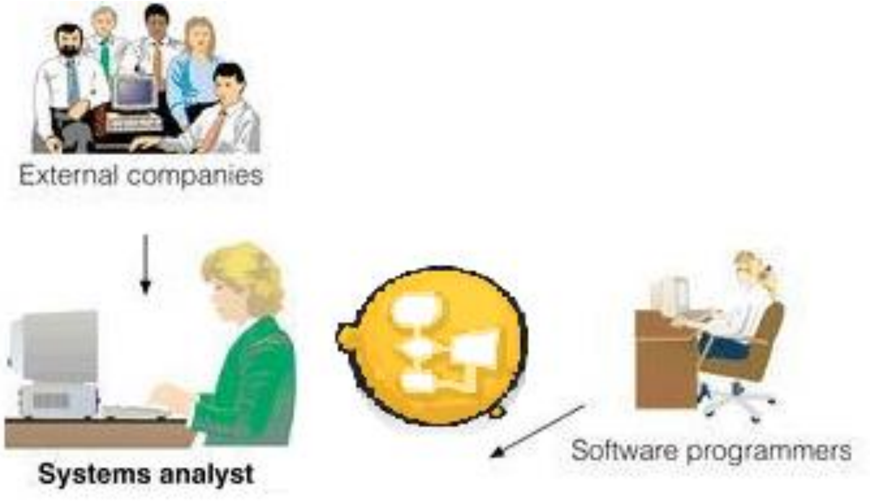
- Often the best way to understand a problem is to draw pictures.
- Pictures often provide us with a more complete idea of the situation than a series of short word or phrases can.
- However, pictures combined with text provide an extremely powerful tool for communication and problem solving.
- Algorithms can be developed more quickly when a flow chart is built to represent such an algorithm.

What is a flowchart?

- Logic diagram to describe each step that the program must perform to arrive at the solution.
- A popular logic tool used for showing an algorithm in graphics form.
- It may be used by both systems analysts and system designers in order to
  - – understand the system,
  - – and build a system.

Principle of good programming

- The program requirements must be specified in full and in writing. These specifications will be prepared by a systems analyst.
- A programmer has the task of converting these specifications into a written program.



Principle of good programming – cont.

- In developing a program, programmers should keep “working papers”. The “working papers” might include a decision table or flowchart (or both). They can refer back to these papers later to check what they have done in case:
  - there is an error in the program for correction;
  - the user of the program asks for a change in the program, e.g. for an extra bit of processing on input data, perhaps to produce an additional report.

Flowchart with programming

- A flowchart can be a really useful tool for a programmer. They can use the flowchart to guide them in terms of what they need to program (code).
- Flowcharts to a programmer is like the blue print to a architect!

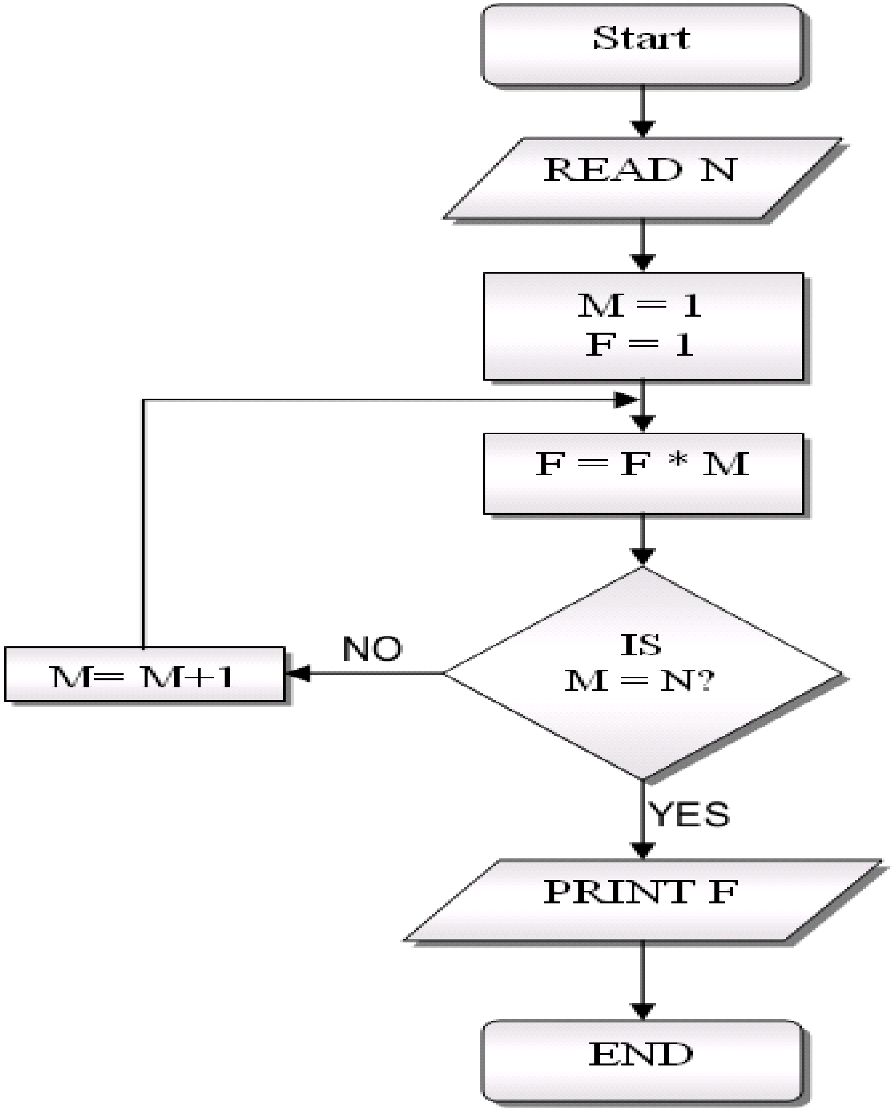
Uses (Advantages) of flowcharts

- To clarify the logic of a problem
- To analyse the actions resulting from a set of conditions.
- To sort out the procedural steps in the program.
- As aids to program construction and coding.
- As communicating documents, e.g. to explain the program to other programmers and the system analyst.

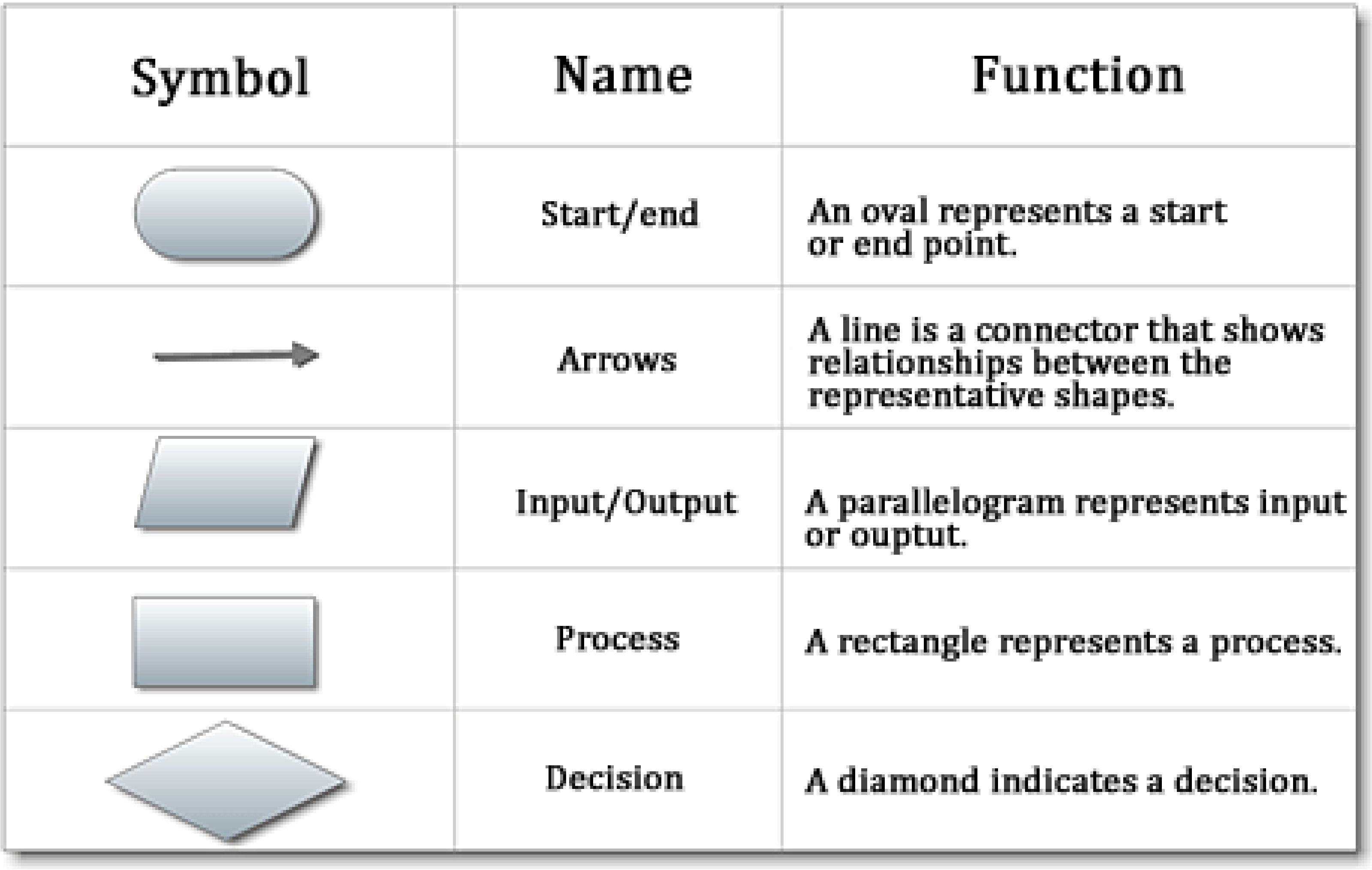
Disadvantages of flowcharts

- Complicated processing might require flowcharts that stretch on to several pages.
- They are not easy to amend. Alterations might involve a complete re-drawing of the flowchart, which could be a time consuming task.

An example of a flowchart

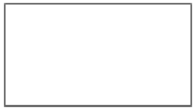


These are the flowcharting symbols



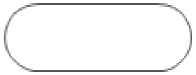
Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.

Other flowcharts symbols



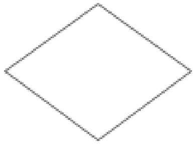
Process

Any processing function.



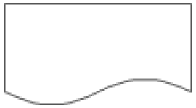
Terminator

Indicates the beginning or end of a program flow in your diagram.



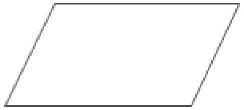
Decision

Decision point between two or more paths in your flowchart.



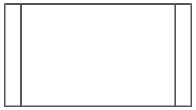
Document

Data that can be read by people, such as printed output.



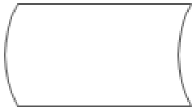
Data

Can represents any type of data in a flowchart.



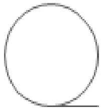
Predefined Process

A named process, such as a subroutine or a module.



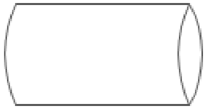
Stored Data

Any type of stored data.



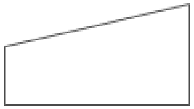
Sequential Data

Data that is accessible sequentially, such as data stored on magnetic tape.



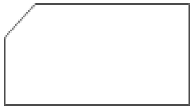
Direct Data

Data that is directly accessible, such as data stored on disk drives.



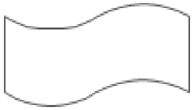
Manual Input

Data that is entered manually, such as with a keyboard or barcode reader.



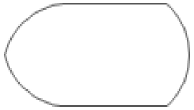
Card

Data that is input by means of cards, such as punch cards or mark-sense forms.



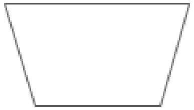
Paper Tape

Data that is stored on paper tape.



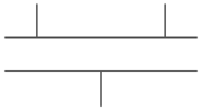
Display

Data that is displayed for people to read, such as data on a monitor or projector screen.



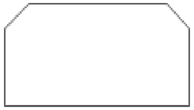
Manual Operation

Any operation that is performed manually (by a person).



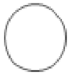
Parallel Mode

Indicates the synchronization of two or more parallel operations.



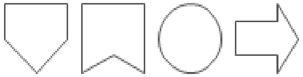
Loop limit

Indicates the start of a loop. Flip the shape vertically to indicate the end of a loop.



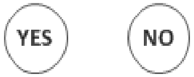
On-page Reference

Use this shape to create a cross-reference from one process to another on the same page of your flowchart.

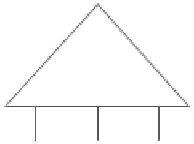


Off-page Reference shapes

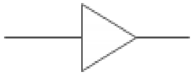
Use this shapes to create a cross-reference and hyperlink from a process on one page to a process on another page.



Yes/No decision indicators



Condition



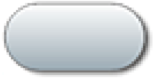
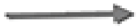
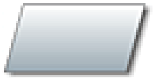
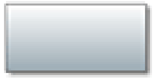
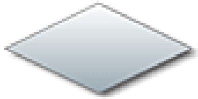
Control Transfer

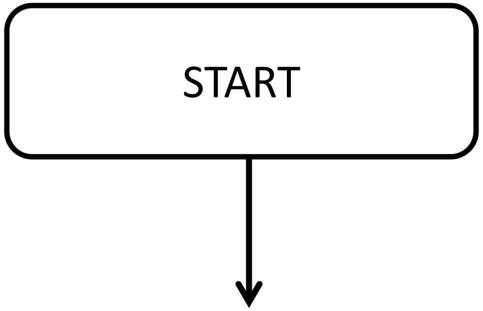
A location in your diagram where control is transferred. The triangle can be positioned anywhere on the line.

Flowcharts (Problem 1)

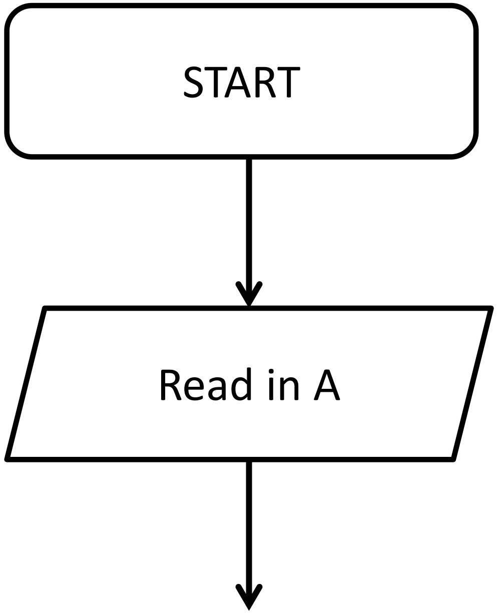
- So let’s say we want to express the following algorithm:

  Read in a number and print it out.

Symbol	Name	Function
	Start/end	An oval represents a start or end point.
	Arrows	A line is a connector that shows relationships between the representative shapes.
	Input/Output	A parallelogram represents input or ouptut.
	Process	A rectangle represents a process.
	Decision	A diamond indicates a decision.

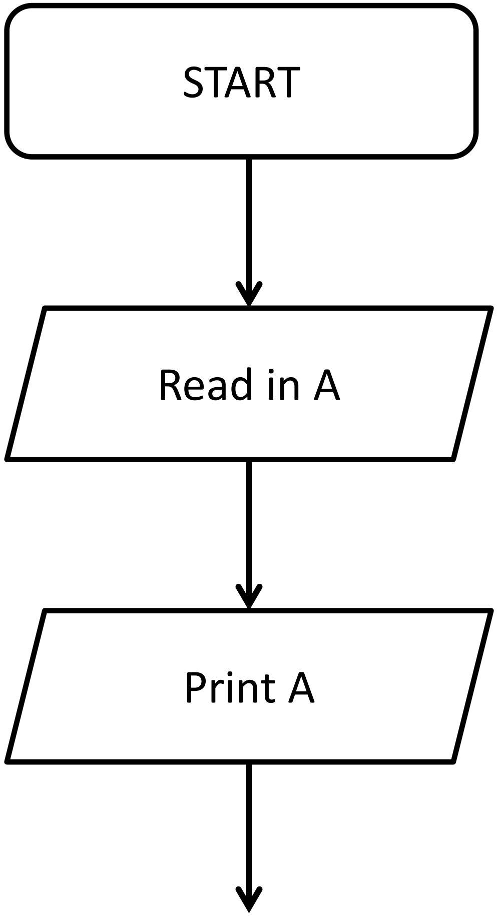


START



START

Read in A



START

Read in A

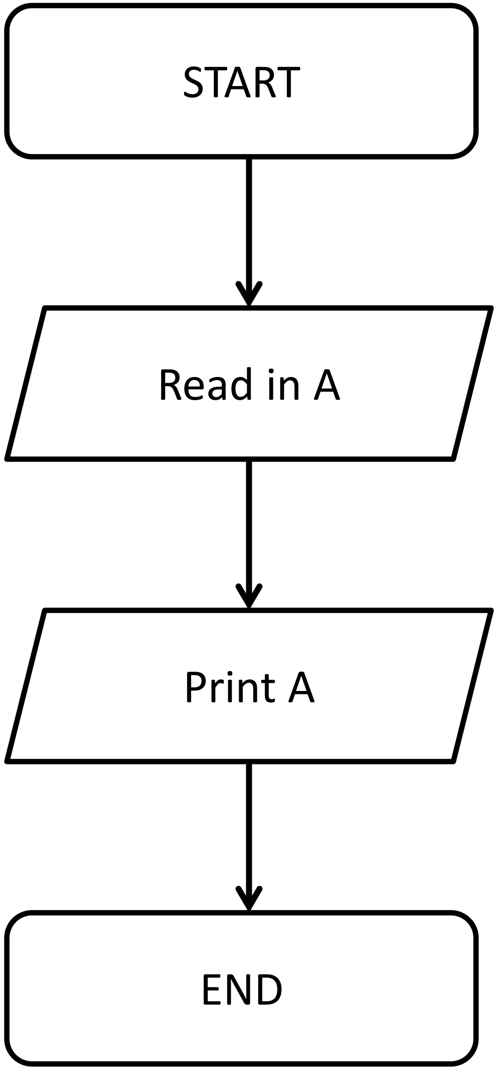
Print A

Start

Input

Output

End

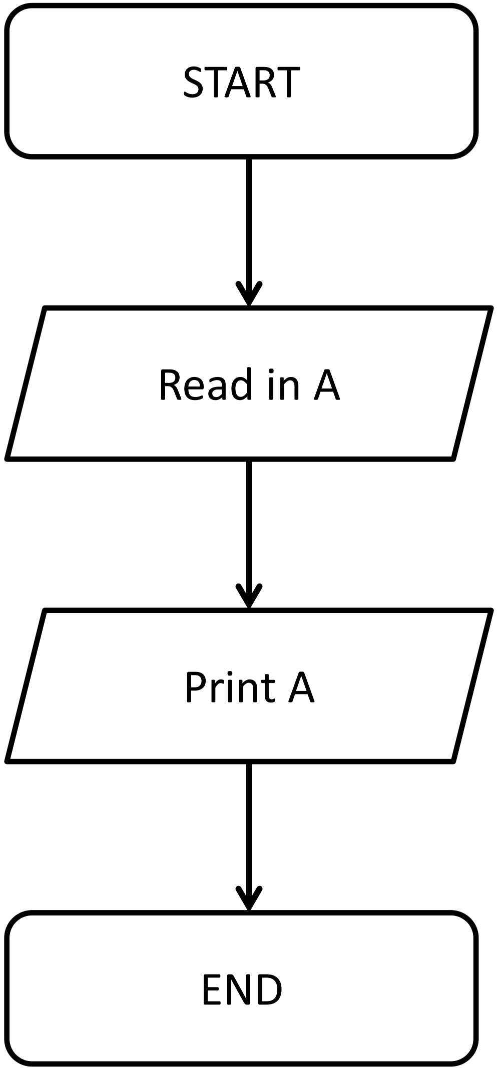


START

Read in A

Print A

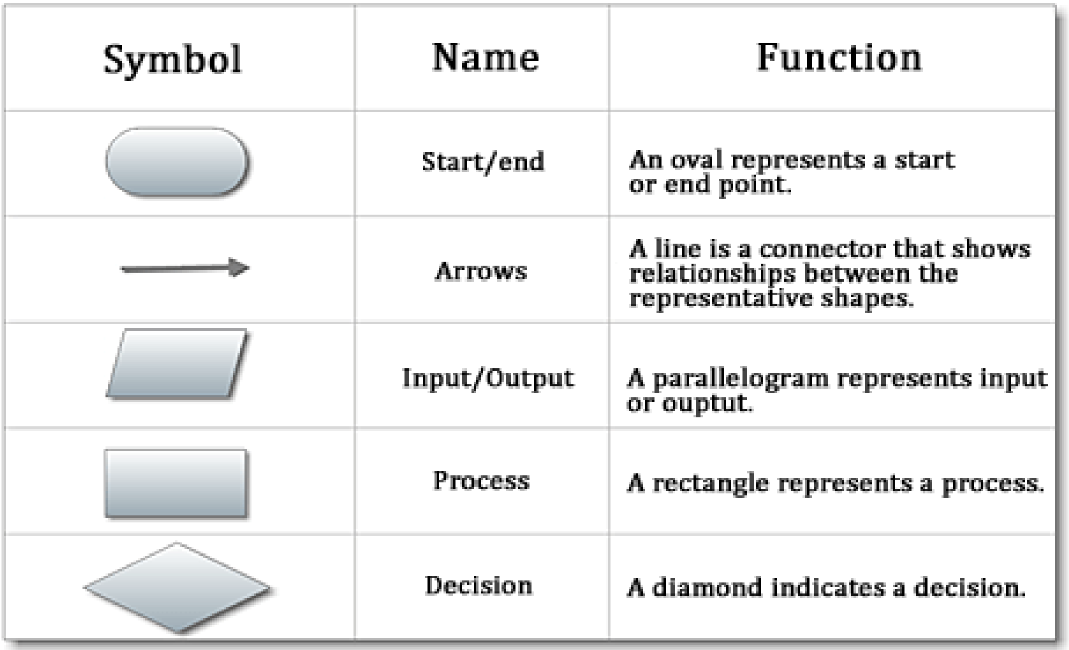
END


START
Read in A
Print A
END

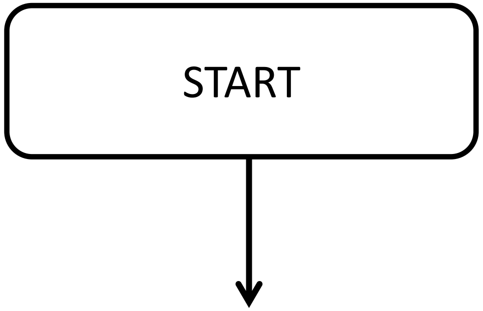
Flowcharts (Problem 2)

- So let’s say we want to express the following algorithm:

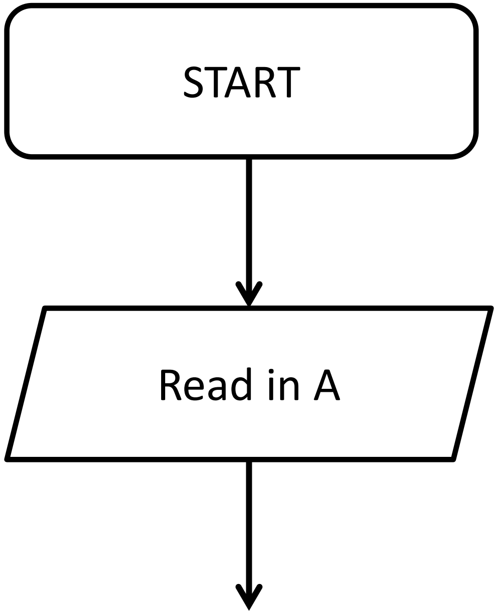
Read in a number and print out double the number.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.

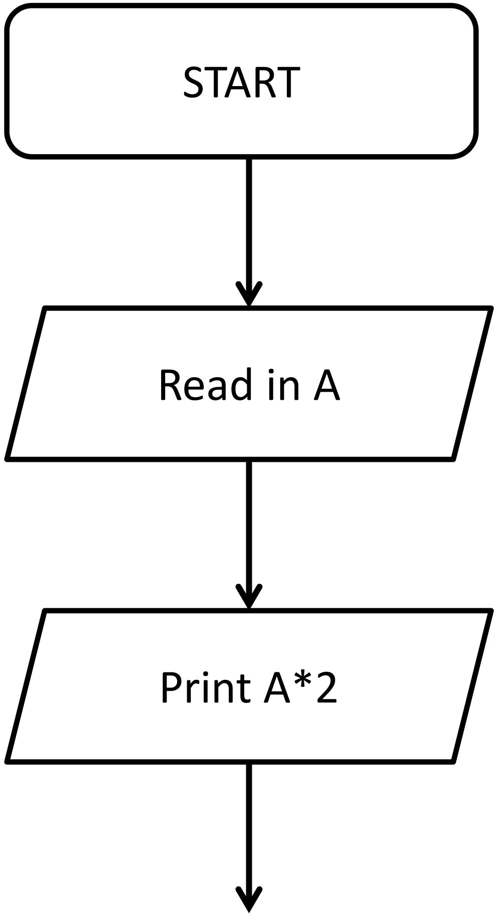


START



START

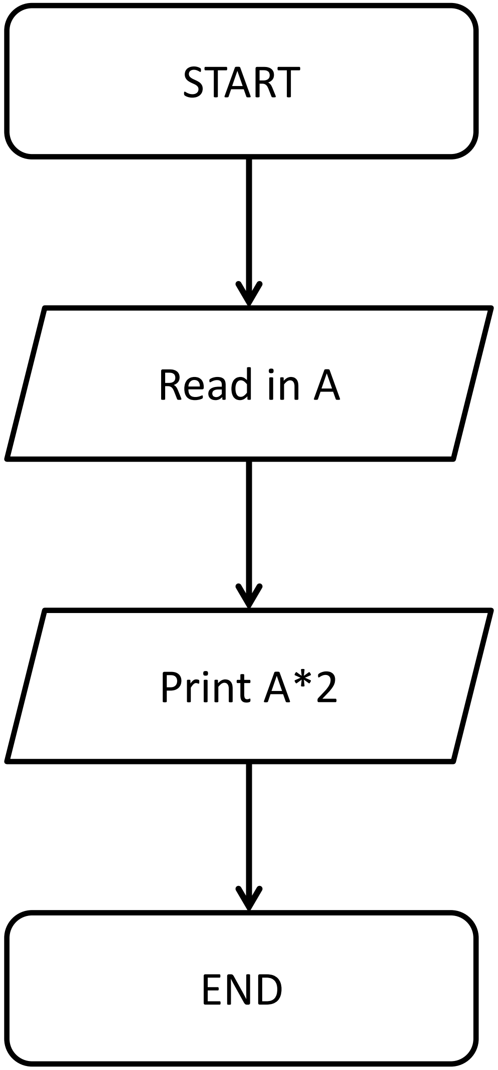
Read in A



START

Read in A

Print A*2



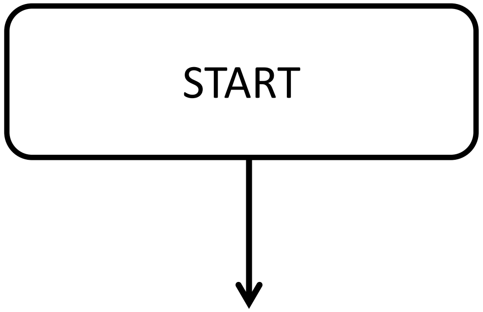
START

Read in A

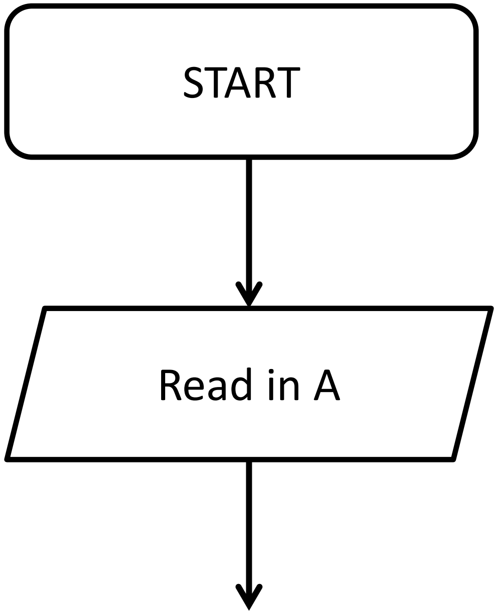
Print A*2

END

Or alternatively…

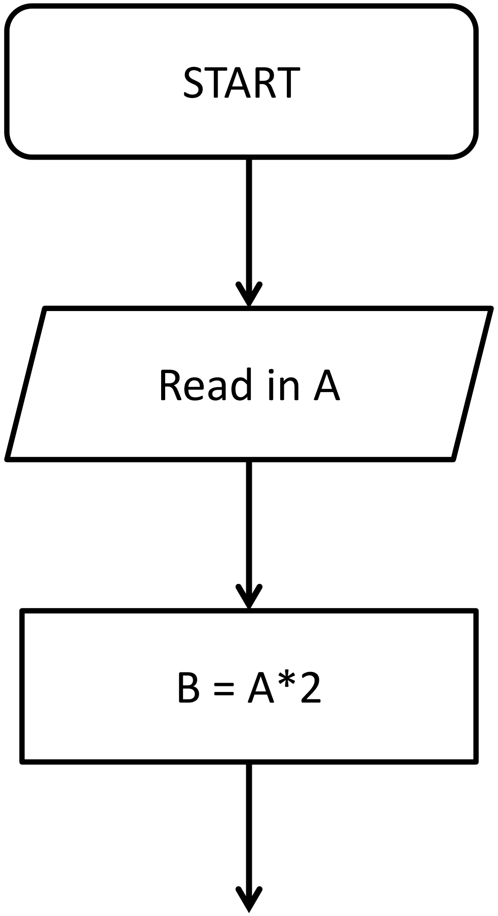


START

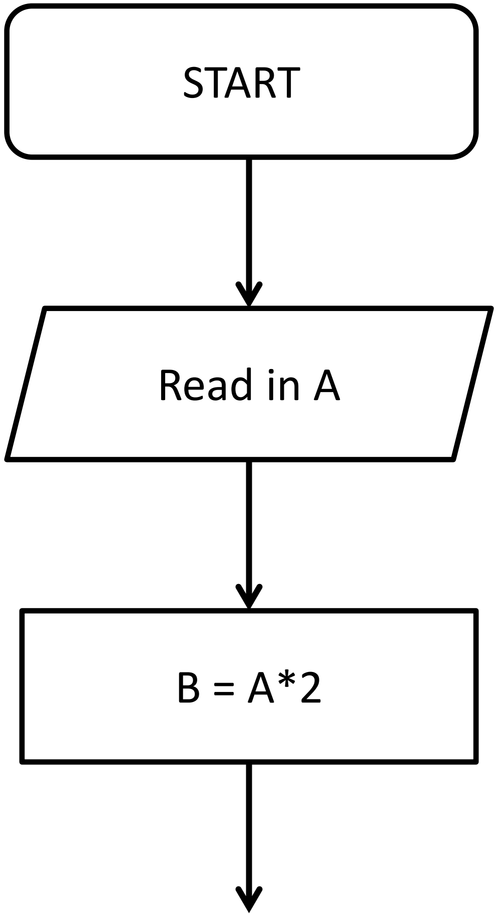


START

Read in A



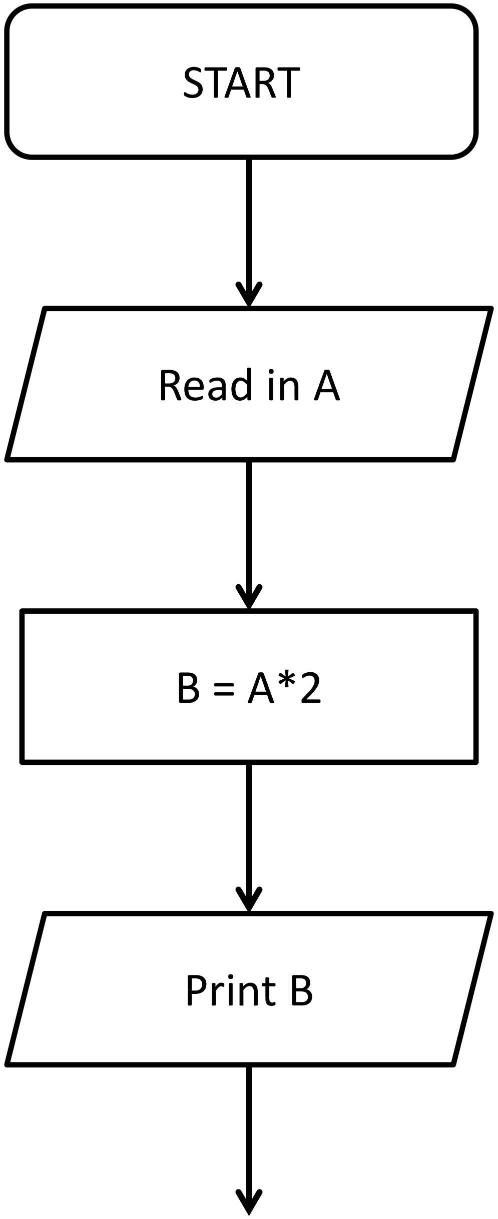
START
Read in A
B = A*2



B = A * 2

can be
read as

*“B gets the
value of A
multiplied by 2”*

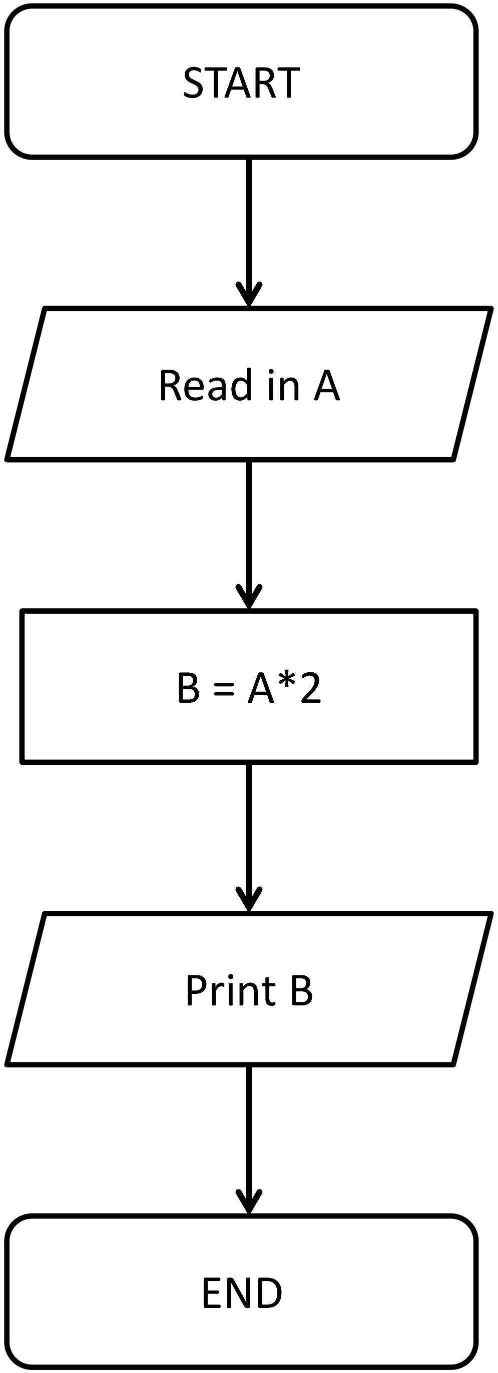


START

Read in A

B = A*2

Print B



START

Read in A

B = A*2

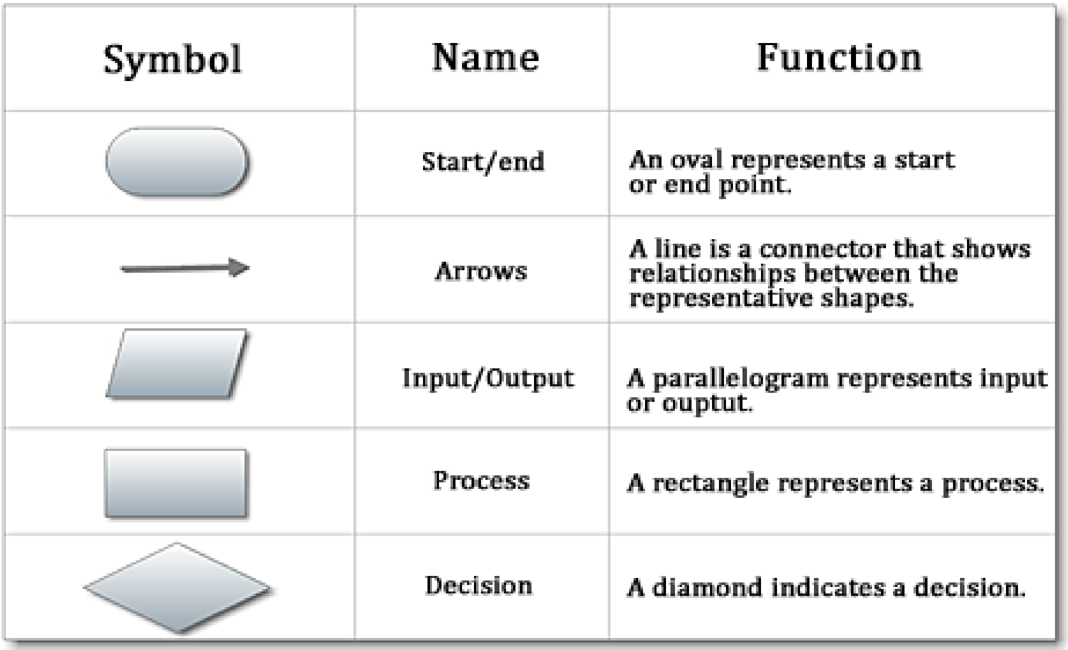
Print B

END

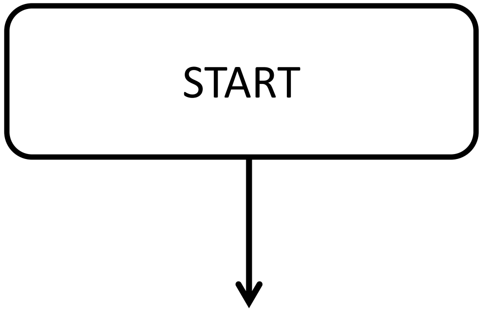
Flowcharts (Problem 3)

- So let's say we want to express the following algorithm:

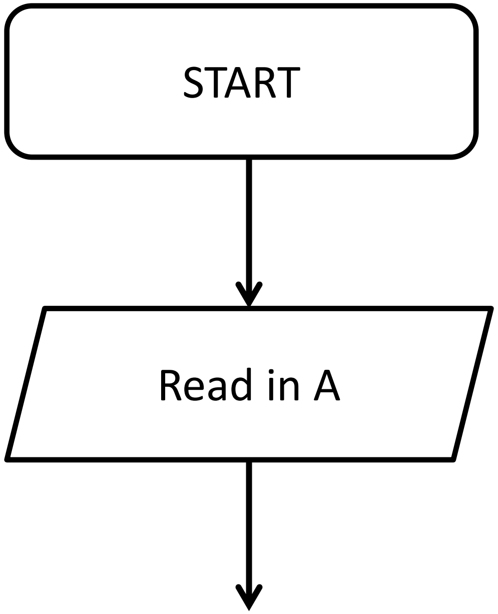
Read in a number, check if it is odd or even.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.

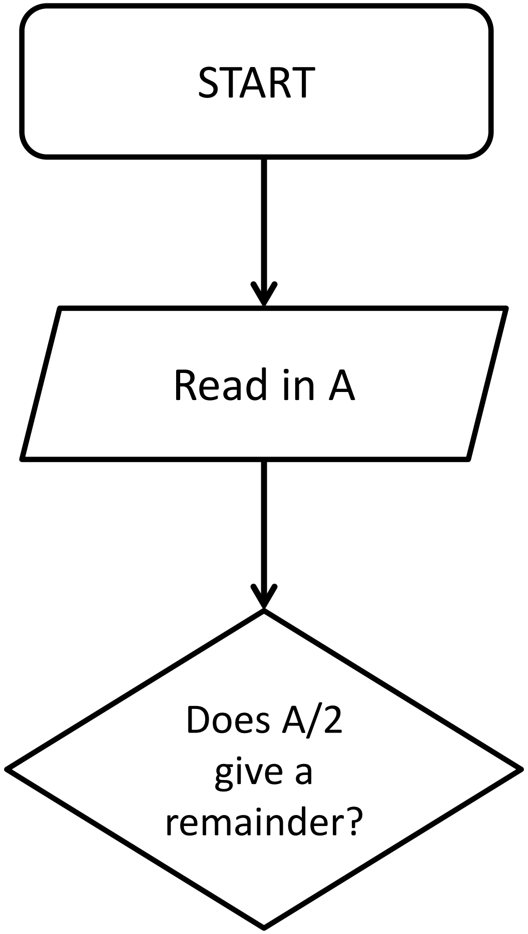


START



START

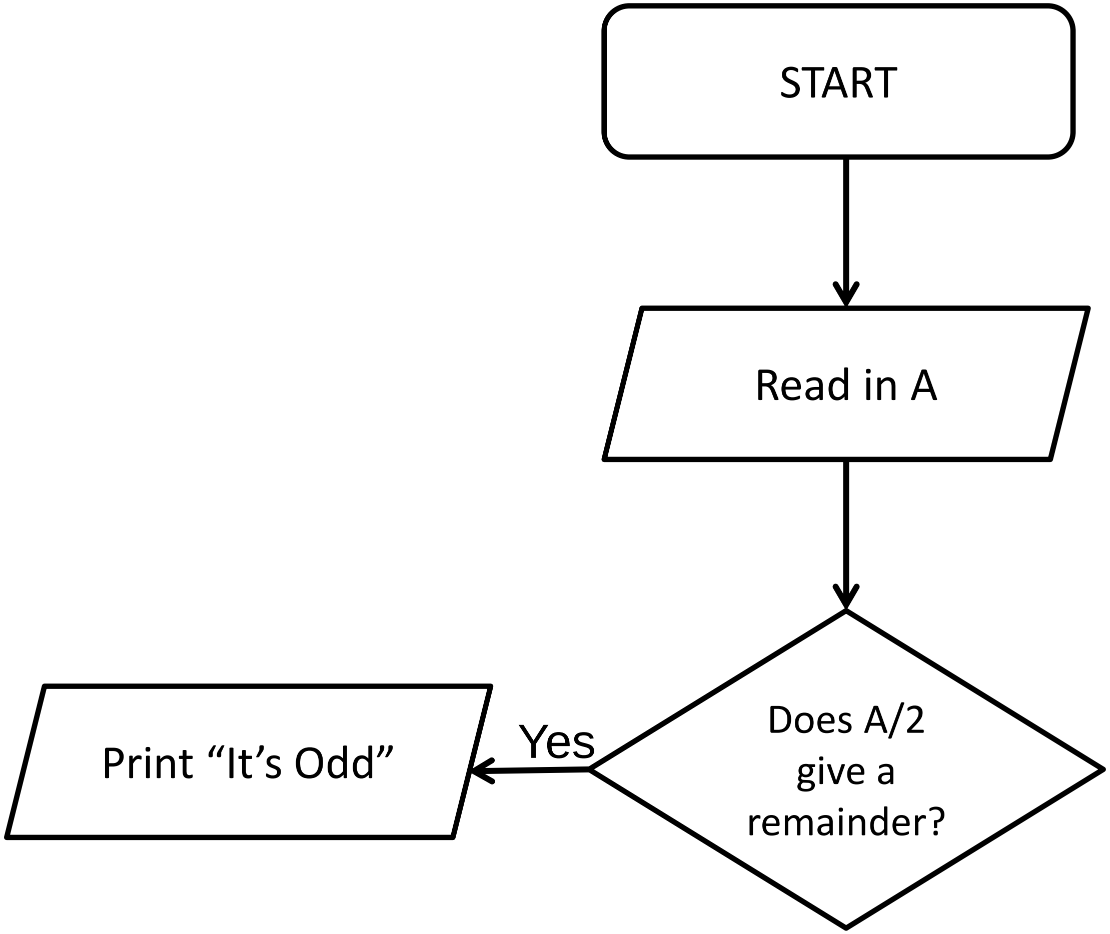
Read in A



START

Read in A

Does A/2 give a remainder?



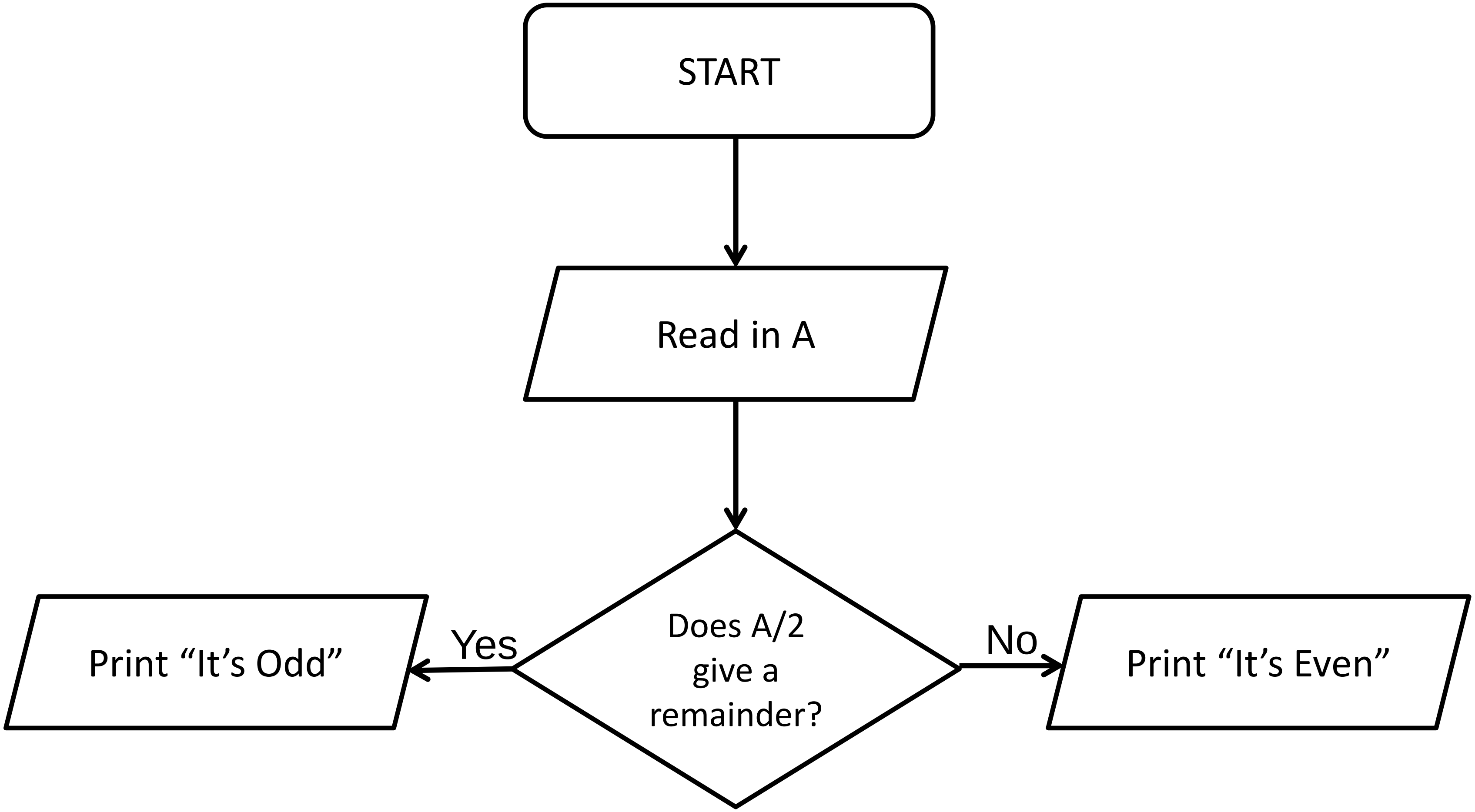
START

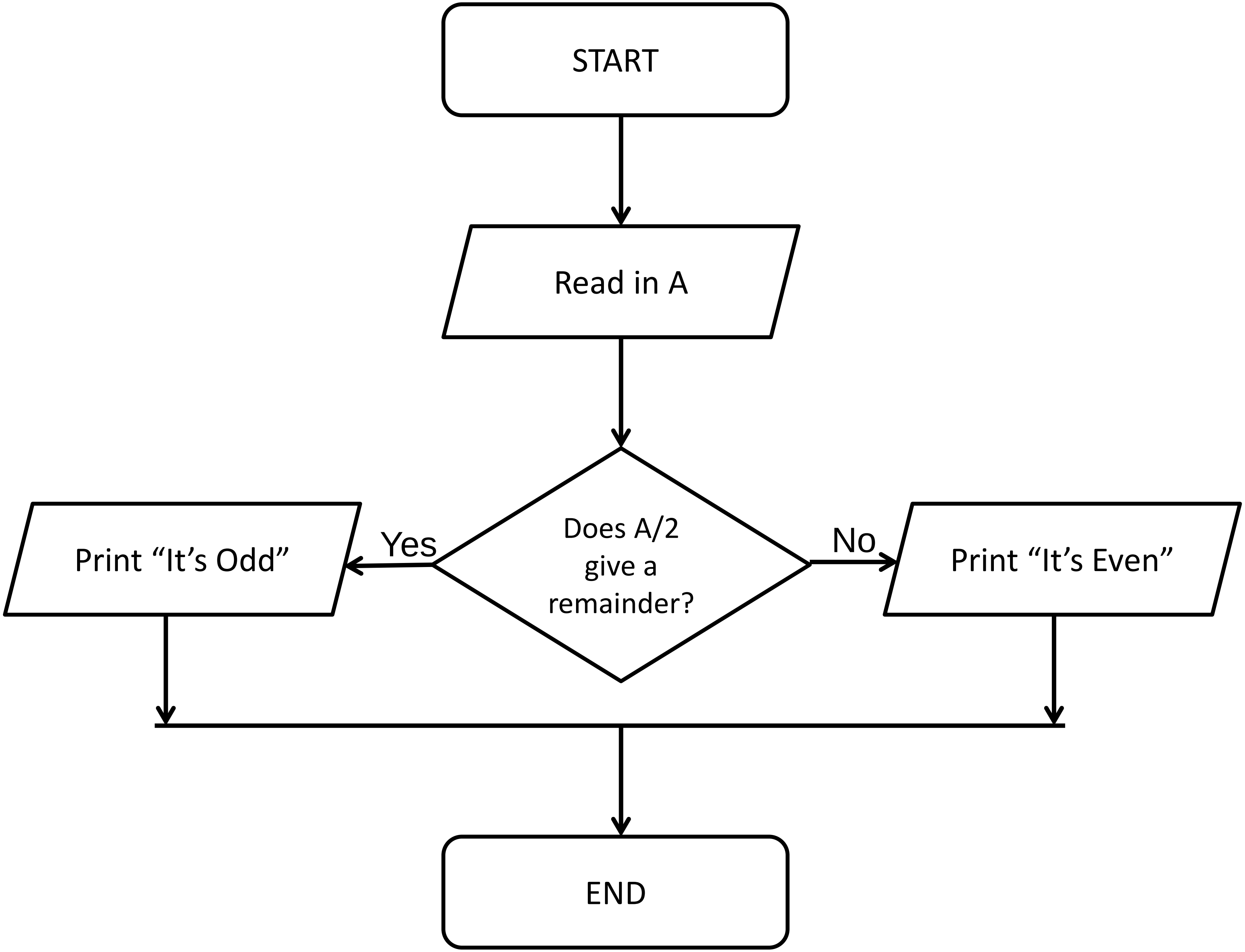
Read in A

Does A/2 give a remainder?

Yes

Print “It’s Odd”


START
Read in A
Does A/2
give a
remainder?
Yes
No
Print “It’s Odd”
Print “It’s Even”

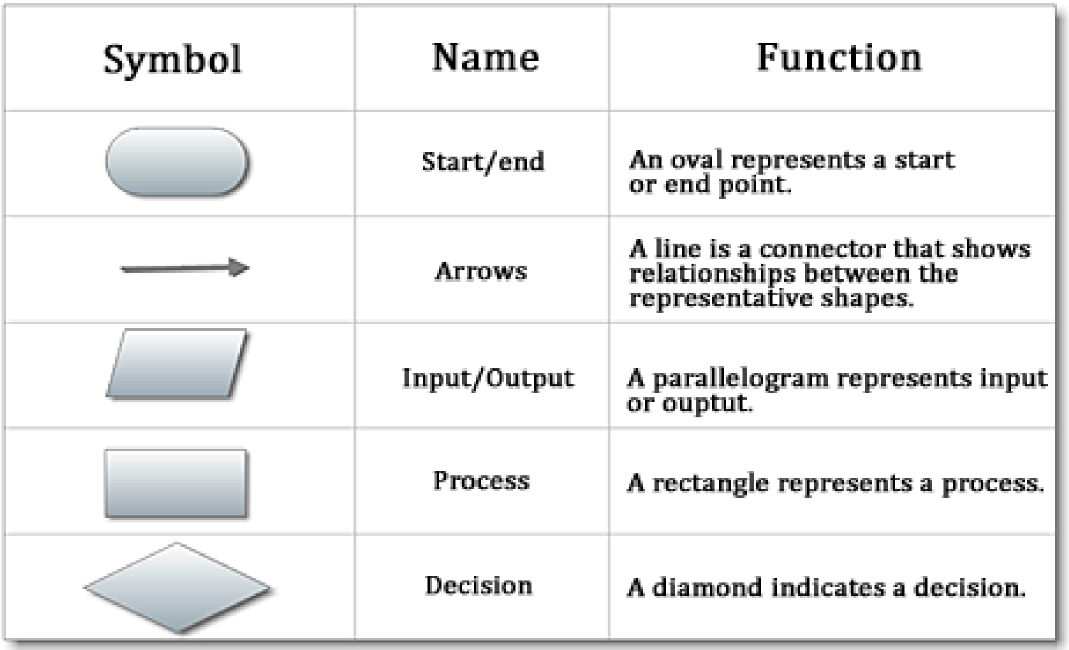


START
Read in A
Does A/2 give a remainder?
Yes
No
Print “It’s Odd”
Print “It’s Even”
END

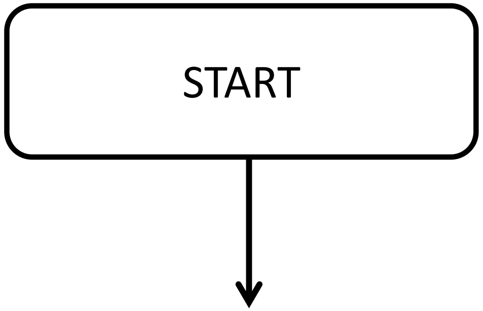
Flowcharts (Problem 4)

- So let’s say we want to express the following algorithm to print out the bigger of two numbers:

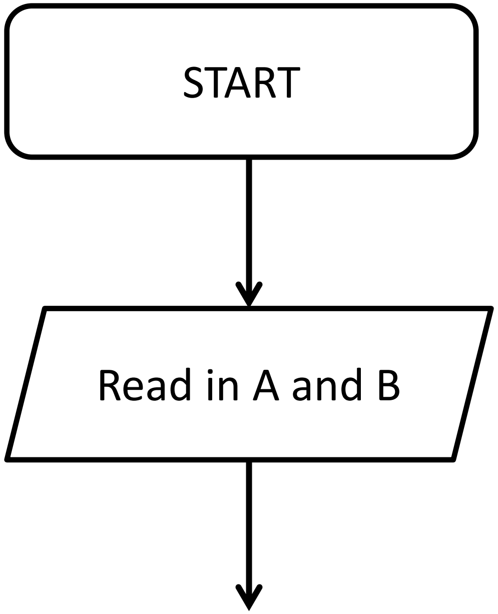
Read in two numbers, call them A and B. If A is bigger than B, print out A, otherwise print out B.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.

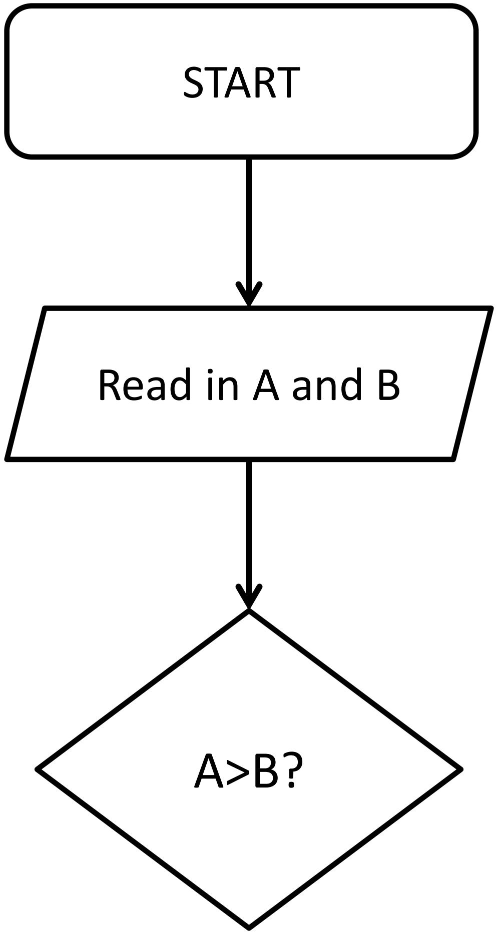


START



START

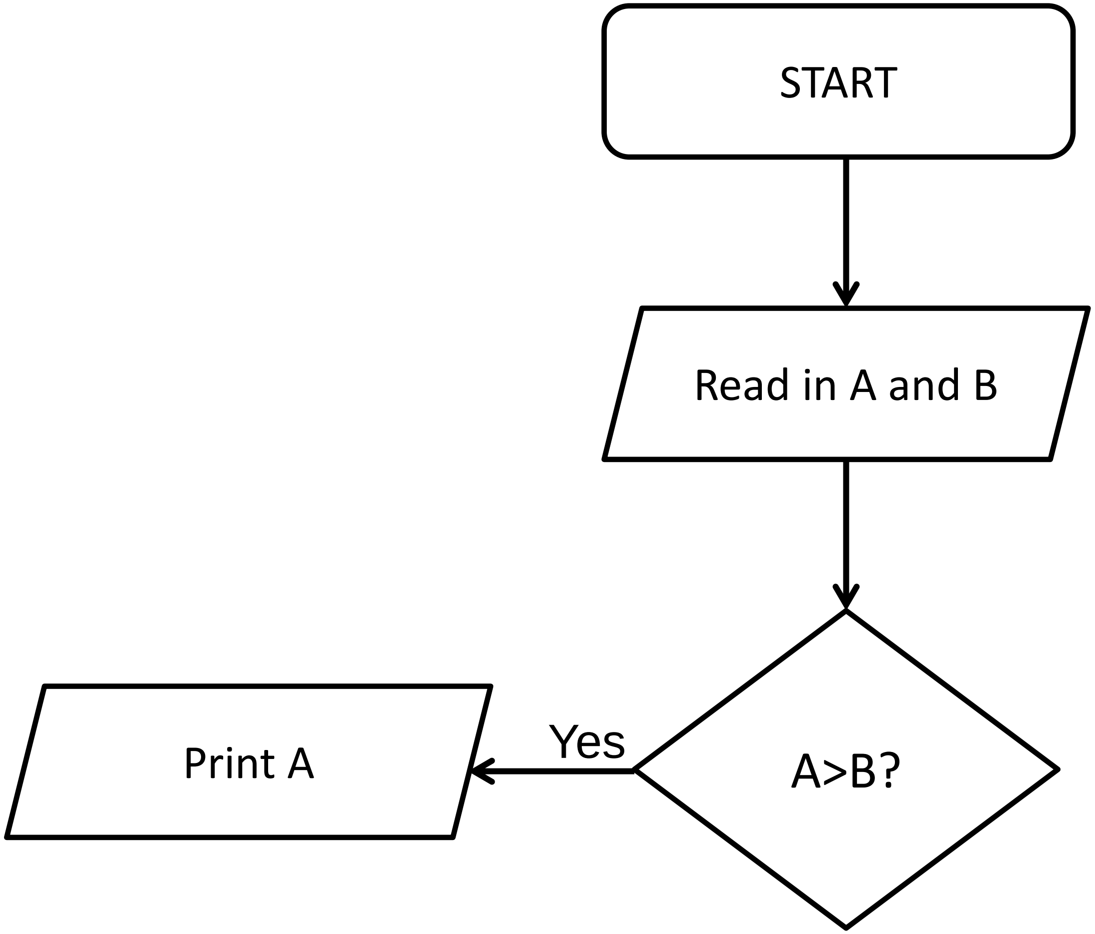
Read in A and B

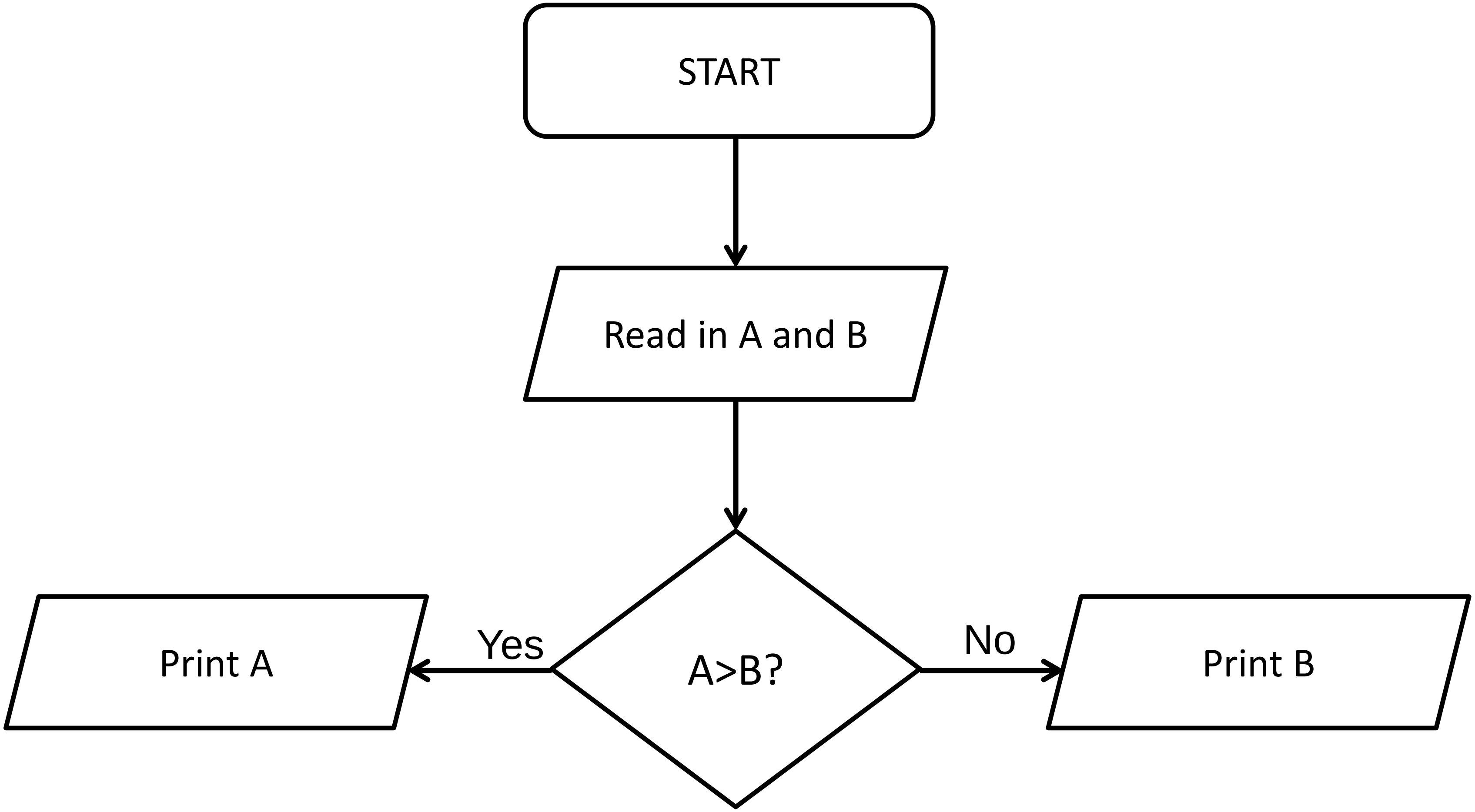


START

Read in A and B

A>B?


START
Read in A and B
A>B?
Yes
Print A



START

Read in A and B

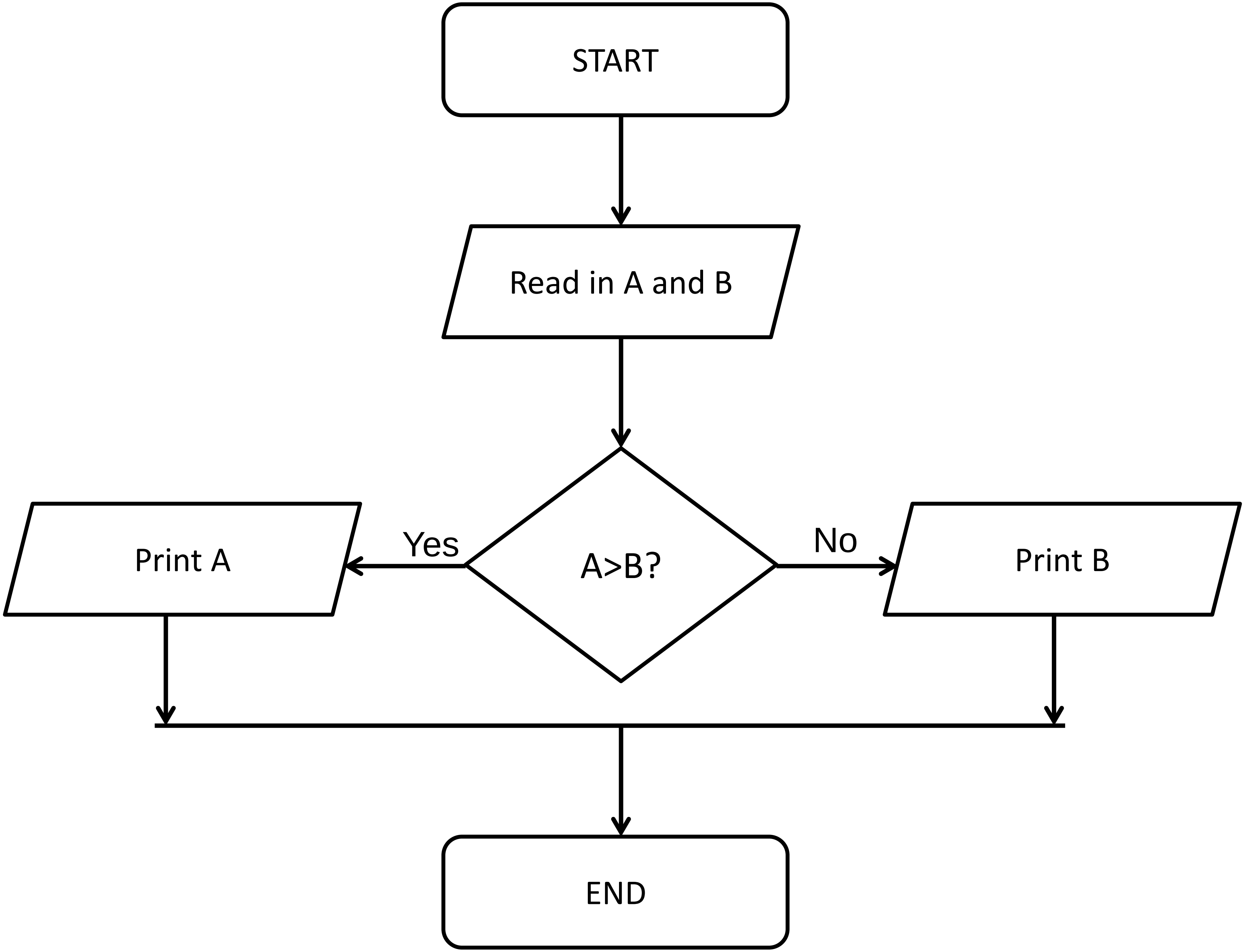
A>B?

Yes

Print A

No

Print B

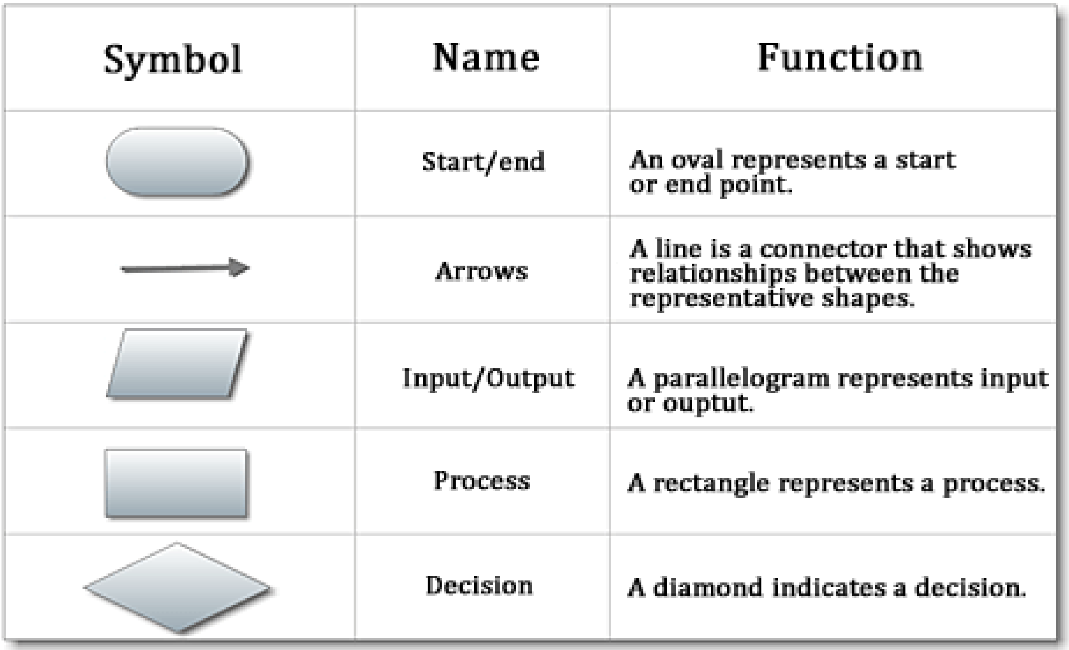


START
Read in A and B
A>B?
Yes
No
Print A
Print B
END

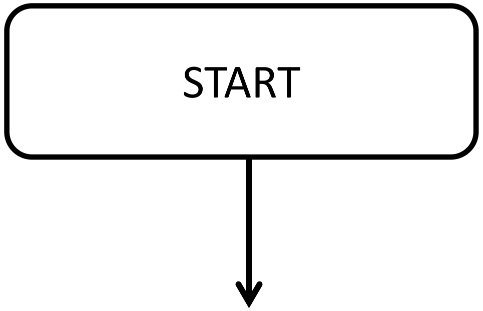
Flowcharts (Problem 5)

- So let’s say we want to express the following algorithm to print out the sum of two numbers:

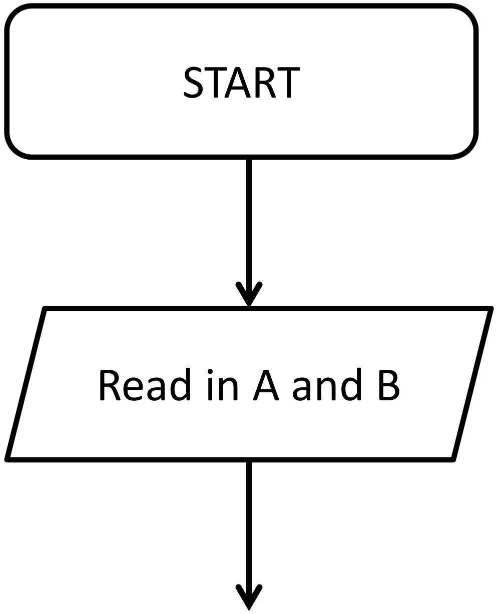
Read in two numbers, call them A and B.
Sum A and B, print out the result.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.

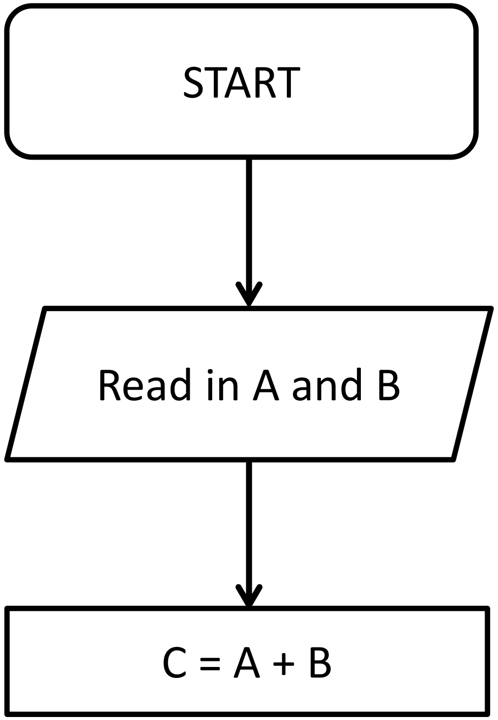


START



START

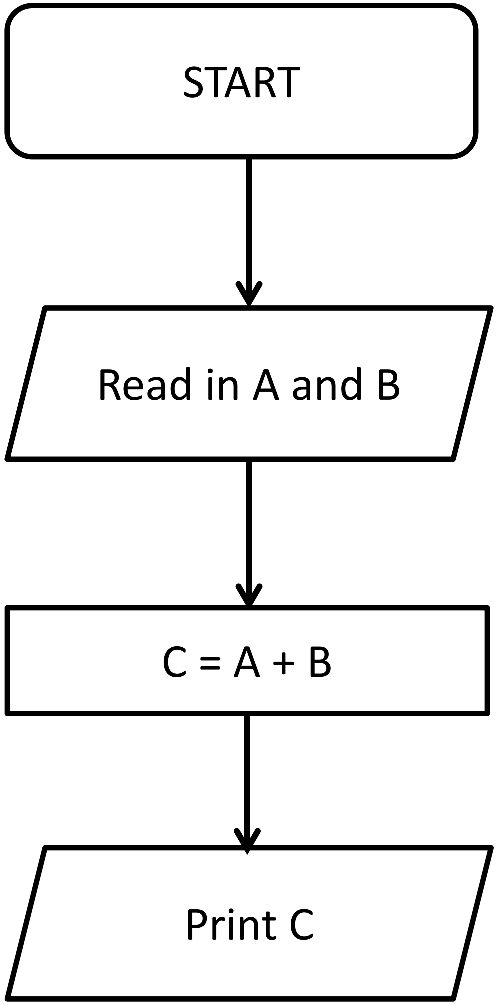
Read in A and B



START

Read in A and B

C = A + B

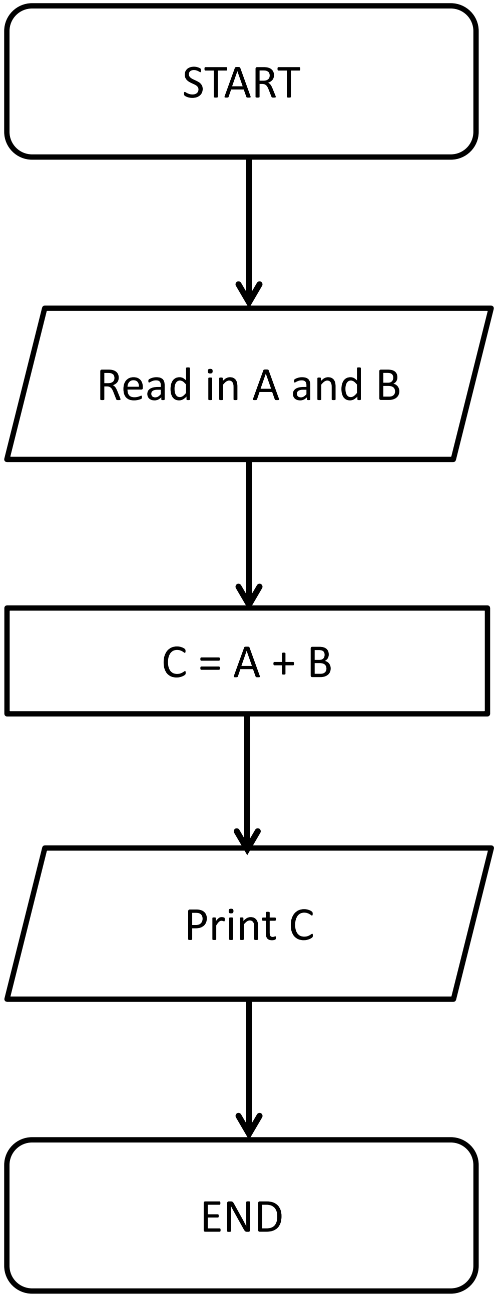


START

Read in A and B

C = A + B

Print C



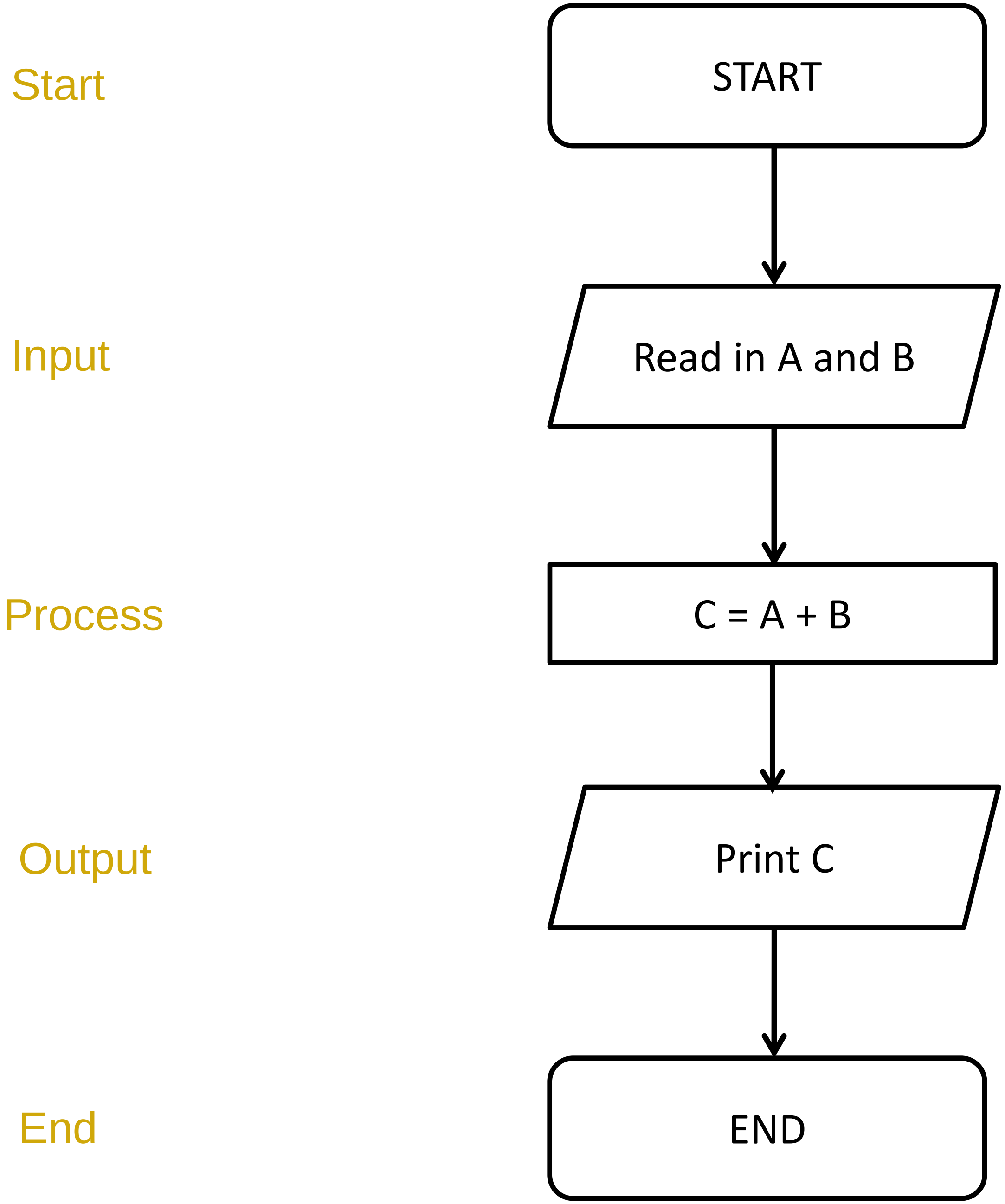
START

Read in A and B

C = A + B

Print C

END



Start

START

Input

Read in A and B

Process

C = A + B

Output

Print C

End

END

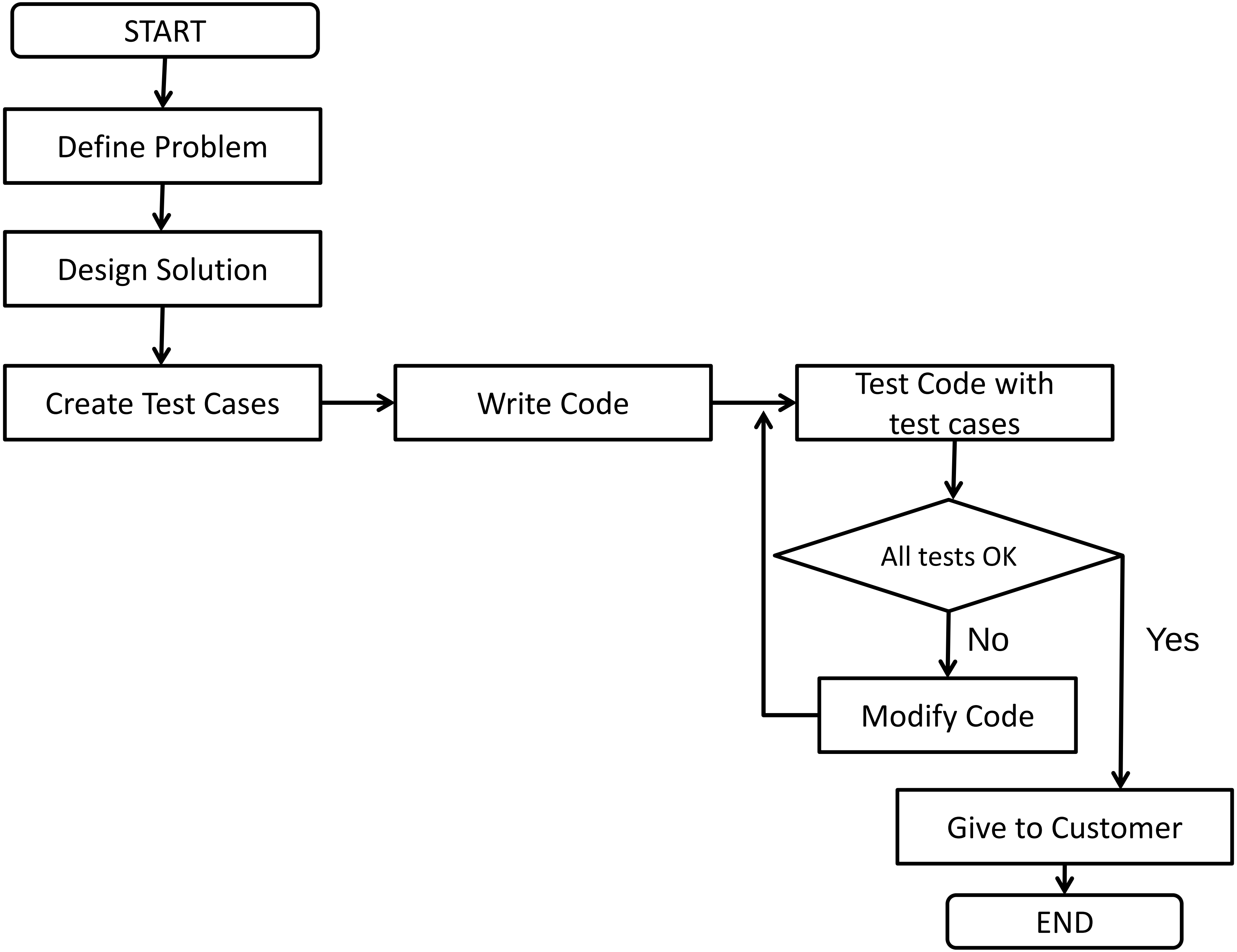
Flowcharts (Problem 6)

- Not just for algorithms!
- *You can express a work process in a flowchart!*

Define The Program Design Process Flowchart

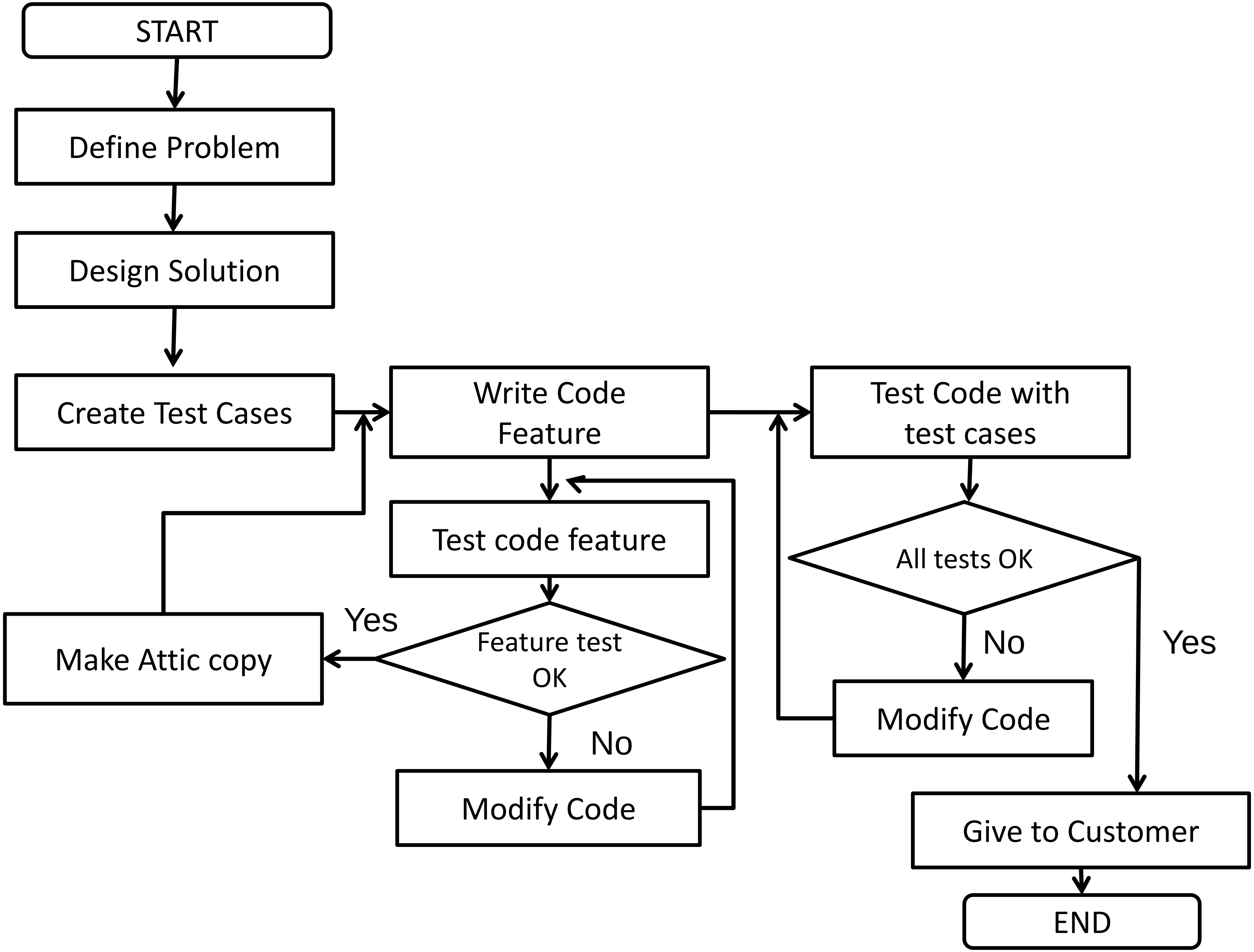
1. Problem Definition
   - What is the objective
   - What is the program to do
2. Design
   - Draw a picture or the execution steps
   - Write down in words the execution steps
3. Test Cases (how will you test it)
   - Write what you will use for testing that it runs and creates the right answer
     - Test Case 1 : 1 + 1 = 2 Simple Case
     - Test Case 2: 5 + 9 = 14 Normal Case
     - Test Case 3: 0 + 9 = 9 Edge condition
     - Test Case 4 : 5 + 0 = 5 Edge condition

       If this was division we could have division by zero issues and very small answers
     - Test Case 5 : 166666666666 + 788777777777777 = 788944444444443 test the very big
4. Write Code
   - Step by step, one piece of functionality at a time, get it working, save a copy.
5. Test Code with test cases
   - Debug the code, change ONLY ONE thing at a time, KEEP SAVING VERSIONS


START
Define Problem
Design Solution
Create Test Cases
Write Code
Test Code with test cases
All tests OK
No
Yes
Modify Code
Give to Customer
END

Or alternatively...

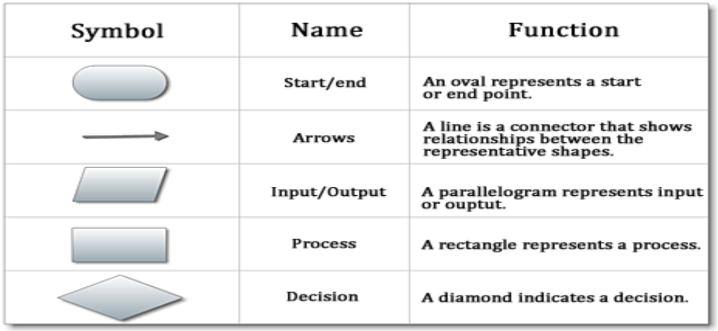
As you should be doing it
and thinking about it


START
Define Problem
Design Solution
Create Test Cases
Write Code Feature
Test code feature
Feature test OK
Yes
Make Attic copy
No
Modify Code
Test Code with test cases
All tests OK
No
Modify Code
Yes
Give to Customer
END

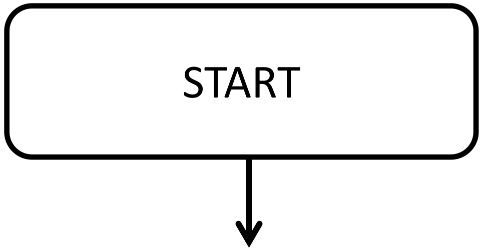
Flowcharts (Problem 7)

- So let’s say we want to express the following algorithm to print out the biggest of three numbers:

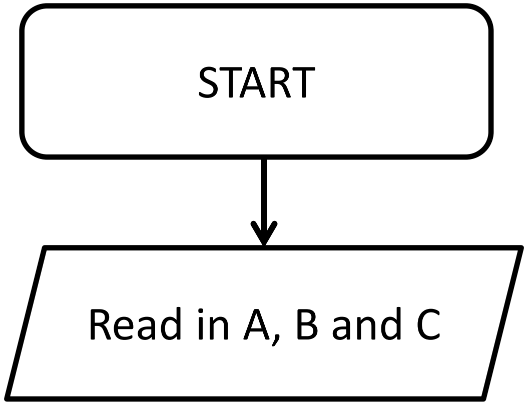
Read in three numbers, call them A, B and C. If A is bigger than B, then if A is bigger than C, print out A, otherwise print out C. If B is bigger than A, then if B is bigger than C, print out B, otherwise print out C.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point.
→	Arrows	A line is a connector that shows relationships between the representative shapes.
▱	Input/Output	A parallelogram represents input or ouptut.
▭	Process	A rectangle represents a process.
◇	Decision	A diamond indicates a decision.

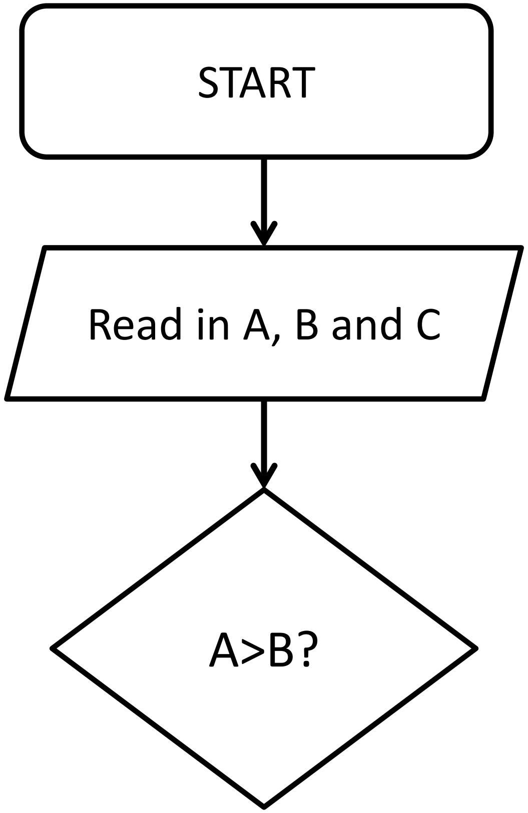


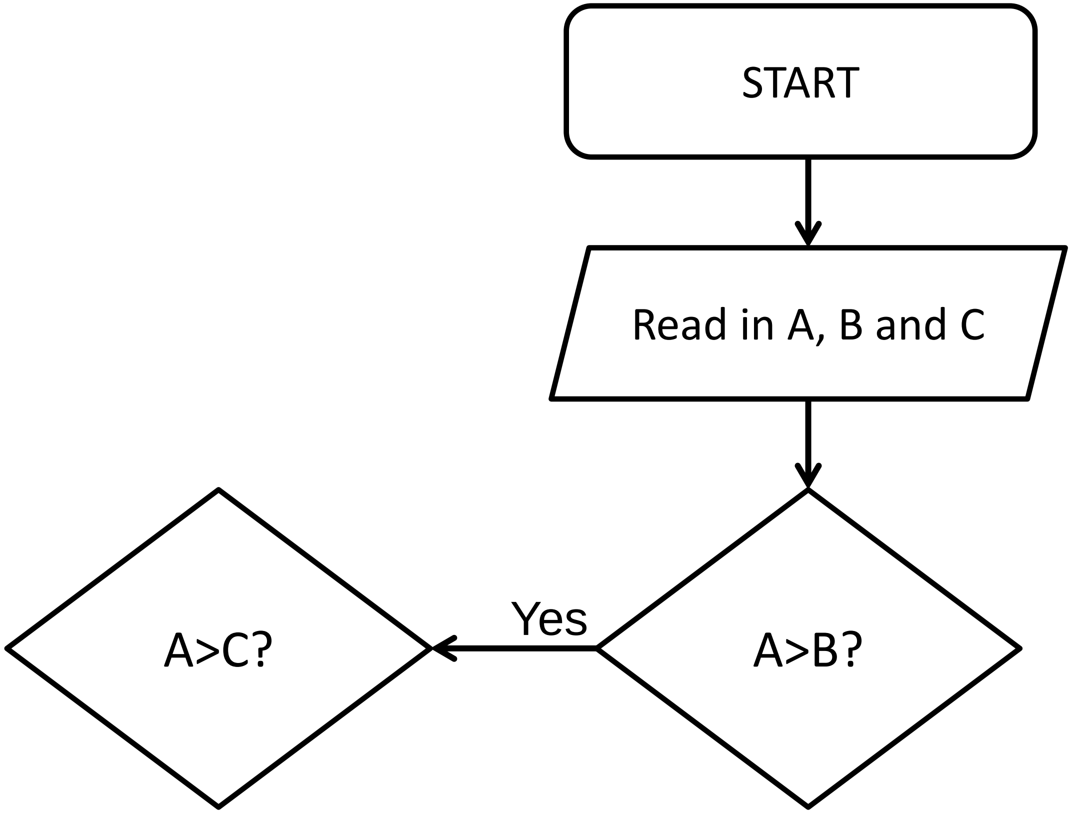
START

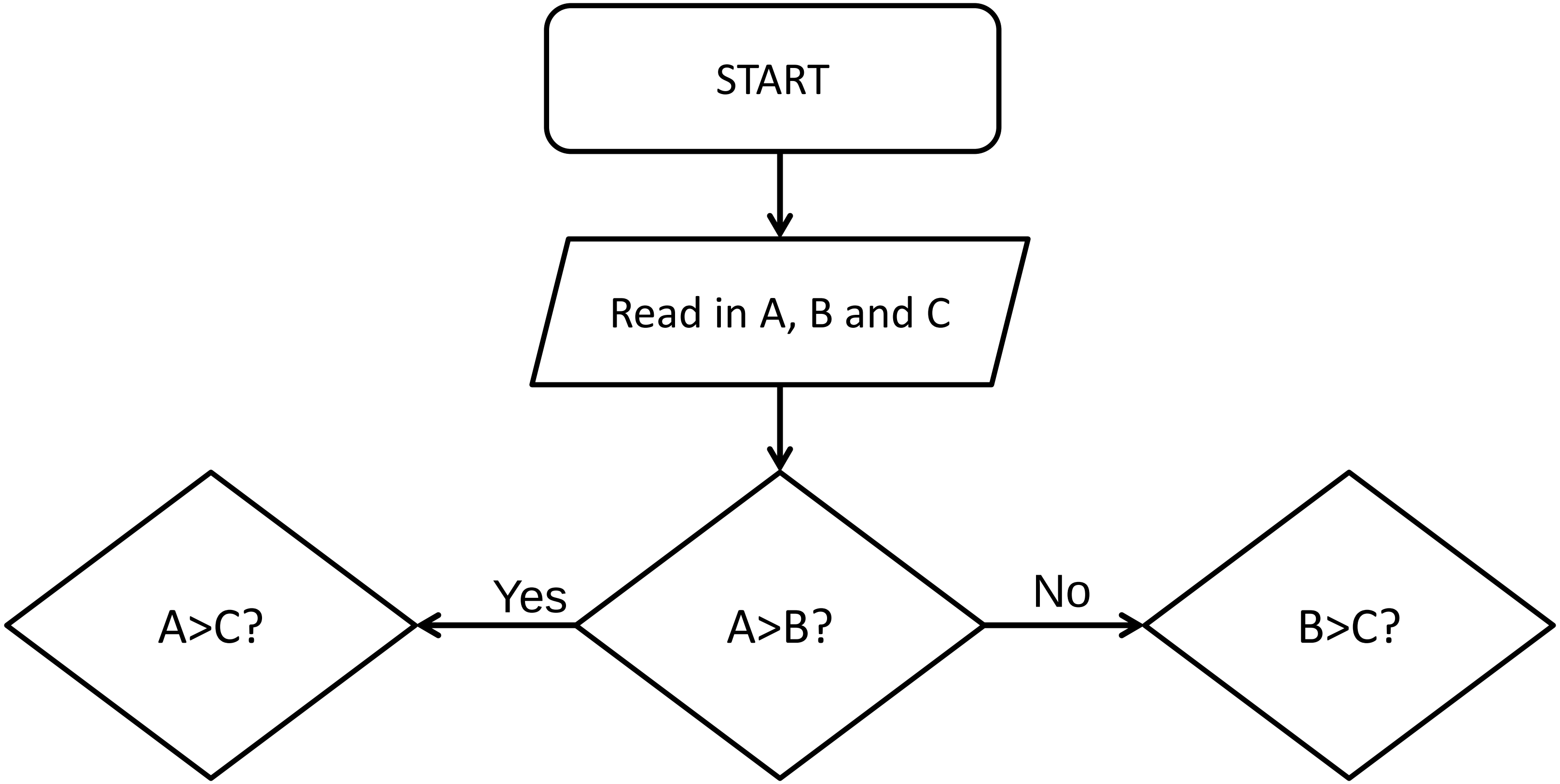


START

Read in A, B and C


START
Read in A, B and C
A>B?


START
Read in A, B and C
A>B?
Yes
A>C?


START

Read in A, B and C

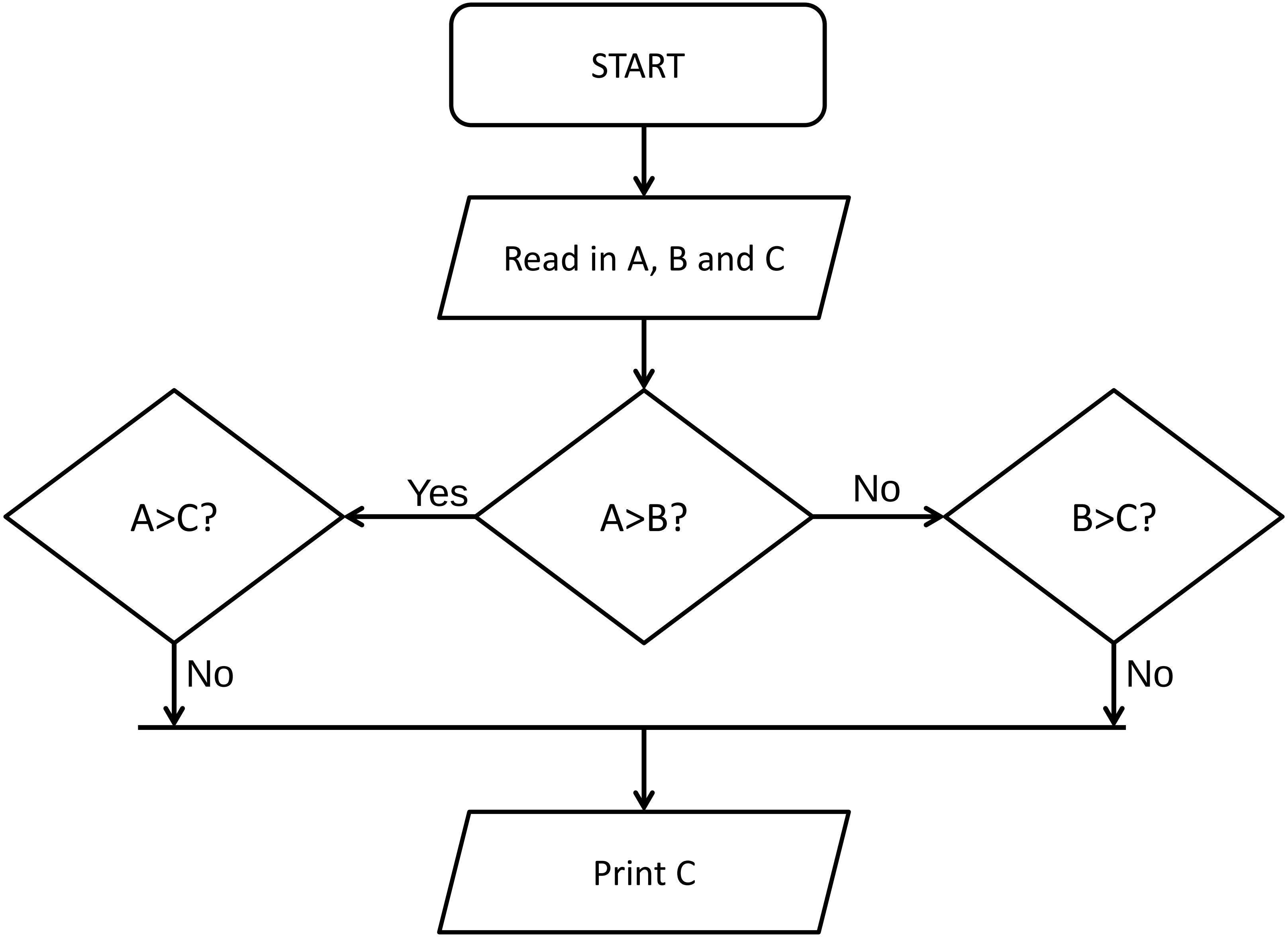
A>B?

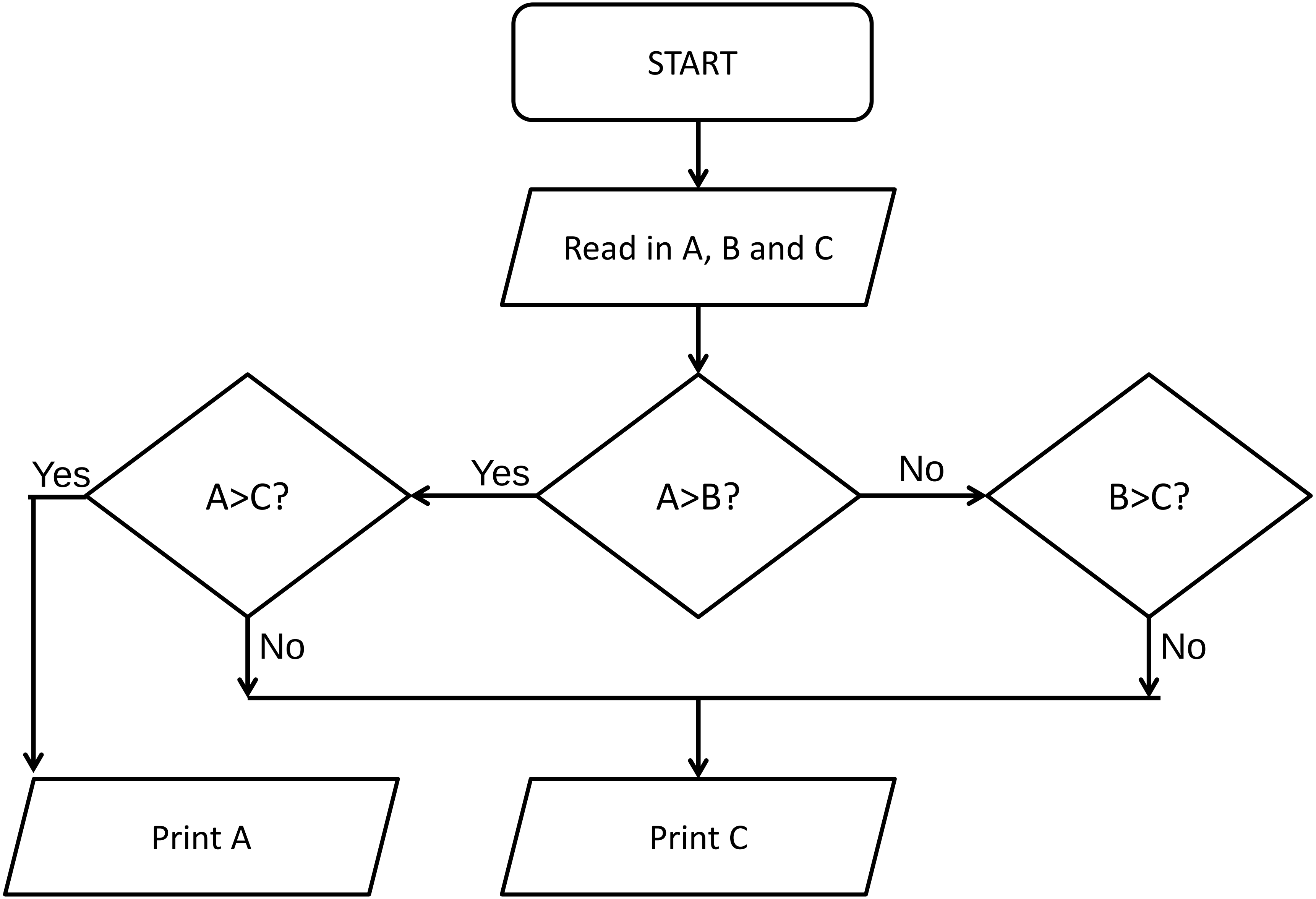
Yes

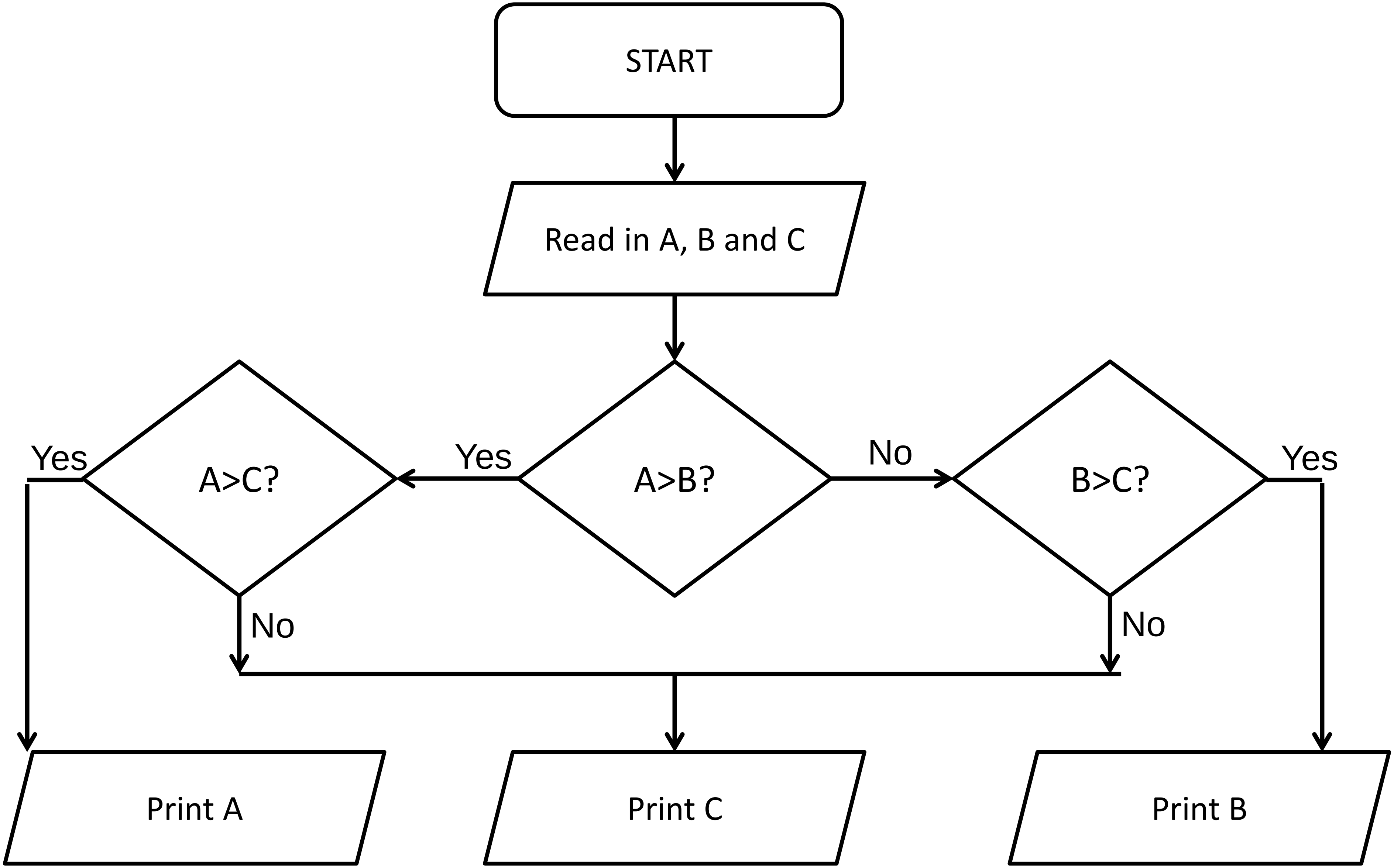
No

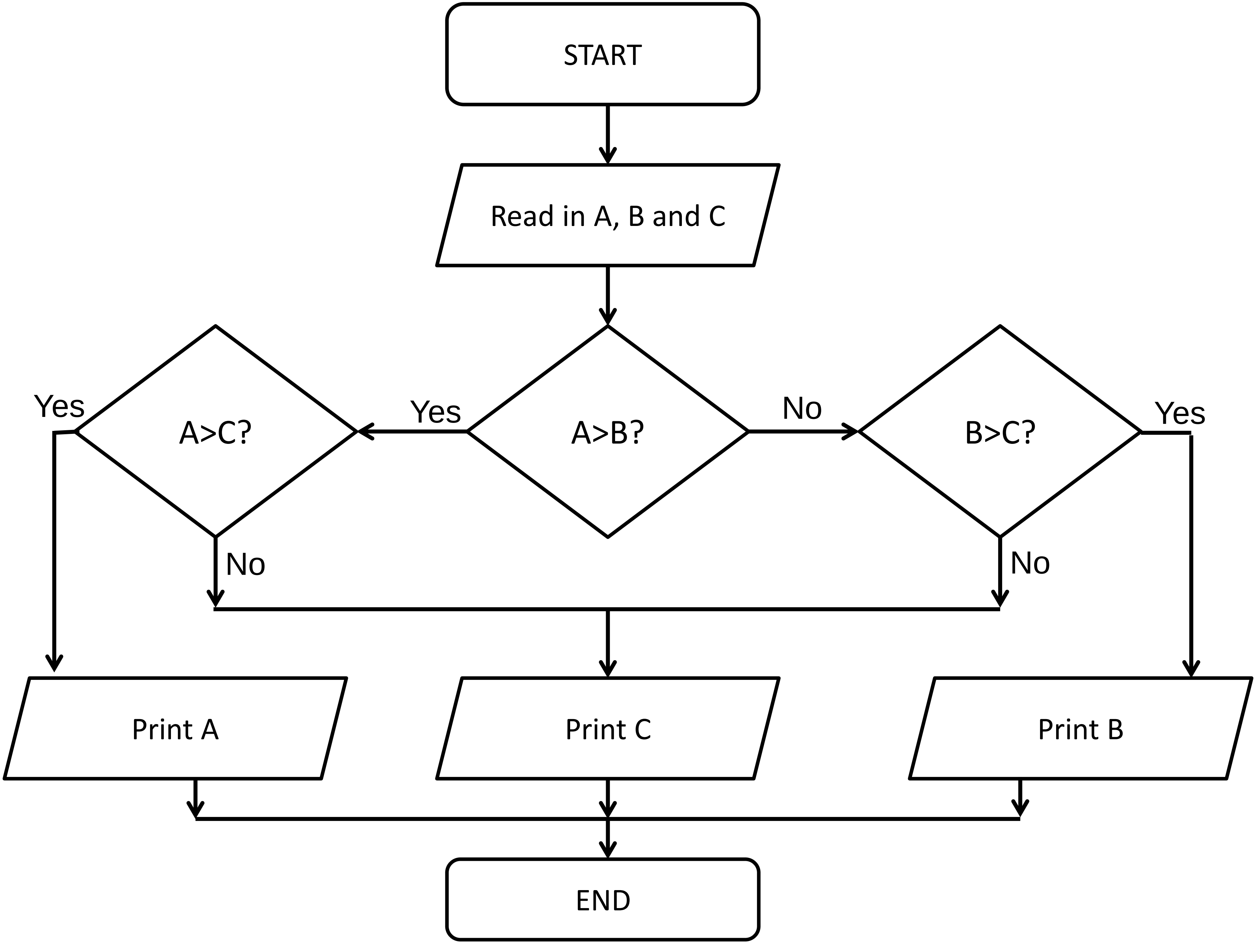
A>C?

B>C?


START
Read in A, B and C
A>C?
Yes
A>B?
No
B>C?
No
No
Print C

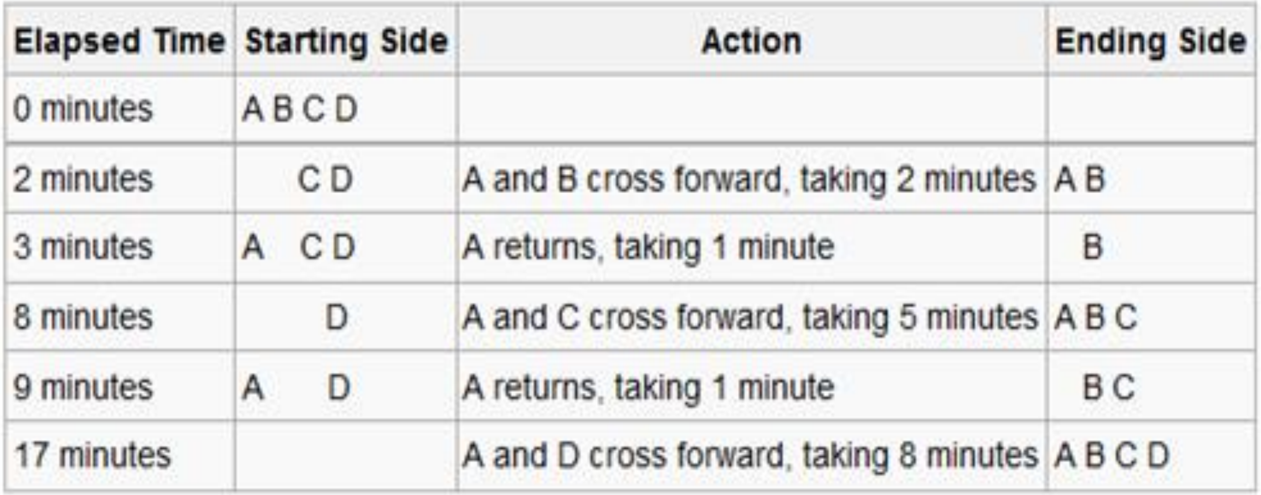

START
Read in A, B and C
A>B?
Yes
No
A>C?
B>C?
Yes
No
No
Print A
Print C


START
Read in A, B and C
A>C?
Yes
Yes
A>B?
No
B>C?
Yes
No
No
Print A
Print C
Print B


START
Read in A, B and C
A>B?
Yes
No
A>C?
Yes
No
B>C?
Yes
No
Print A
Print C
Print B
END

What if A=B=C?

- What happens if you put in A=B=C=1?
- Use the “Crossing” problem solution format (a chart) to see what is happening as you go through the loop
- For each iteration state the Initial state, Instruction/processing, Final state.



Elapsed Time	Starting Side	Action	Ending Side
0 minutes	A B C D		
2 minutes	C D	A and B cross forward, taking 2 minutes	A B
3 minutes	A C D	A returns, taking 1 minute	B
8 minutes	D	A and C cross forward, taking 5 minutes	A B C
9 minutes	A D	A returns, taking 1 minute	B C
17 minutes		A and D cross forward, taking 8 minutes	A B C D

References

- 2009, Barry, Paul and Griffiths, David; Head First Programming, O’Reilly Media Inc.
- 2009, Pine, Chris ; Learn to Program, 2nd Edition, The Pragmatic Programmers