CMPT 120
Intro to CS & Programming I

WEEK 1 (Jan. 6-10)

Lecture 2: From algorithms to pseudocode

http://www.sfu.ca/~jlumbros/Courses/CMPT120/

Contact

Best way (for course questions):

cmpt-120-help@sfu.ca

Or else:

- Instructor: Jérémie Lumbroso <jeremie.lumbroso@sfu.ca>
- TAs:
  - Bhavesh Gupta <bgupta@sfu.ca>,
  - Dan Lin <lindanl@sfu.ca>,
  - Zhensong Qian <zqian@sfu.ca>.

Remarks:

- Always include CMPT 120 at the beginning of your subject line.
- Office hours at TASC 1 9025 (Monday and Friday 12:30pm-1:30pm).

Quick Check

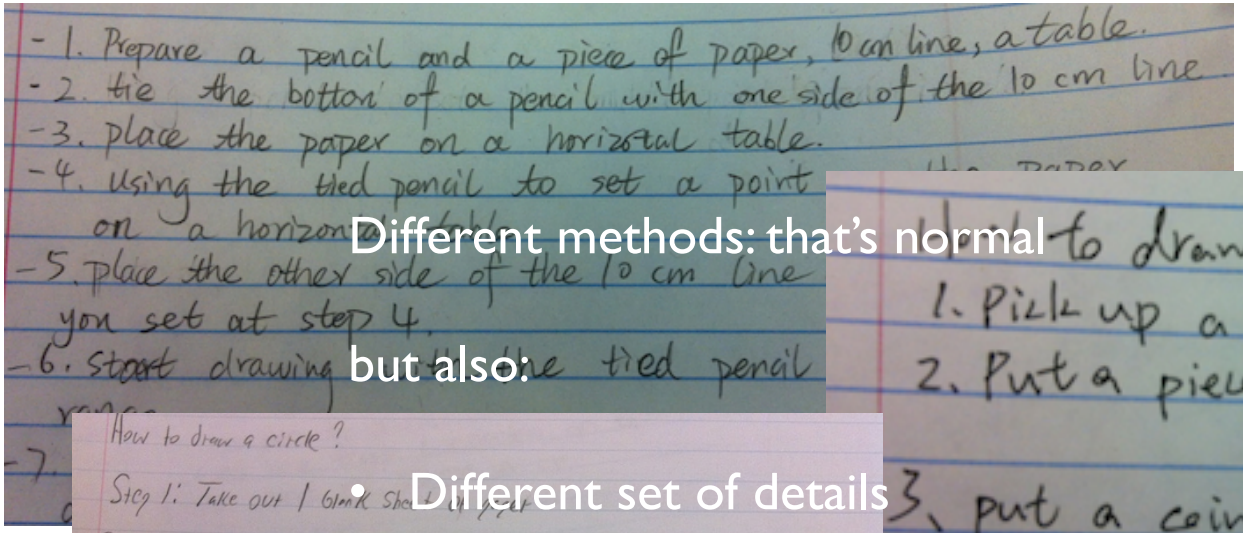
How many of you have an iClicker?

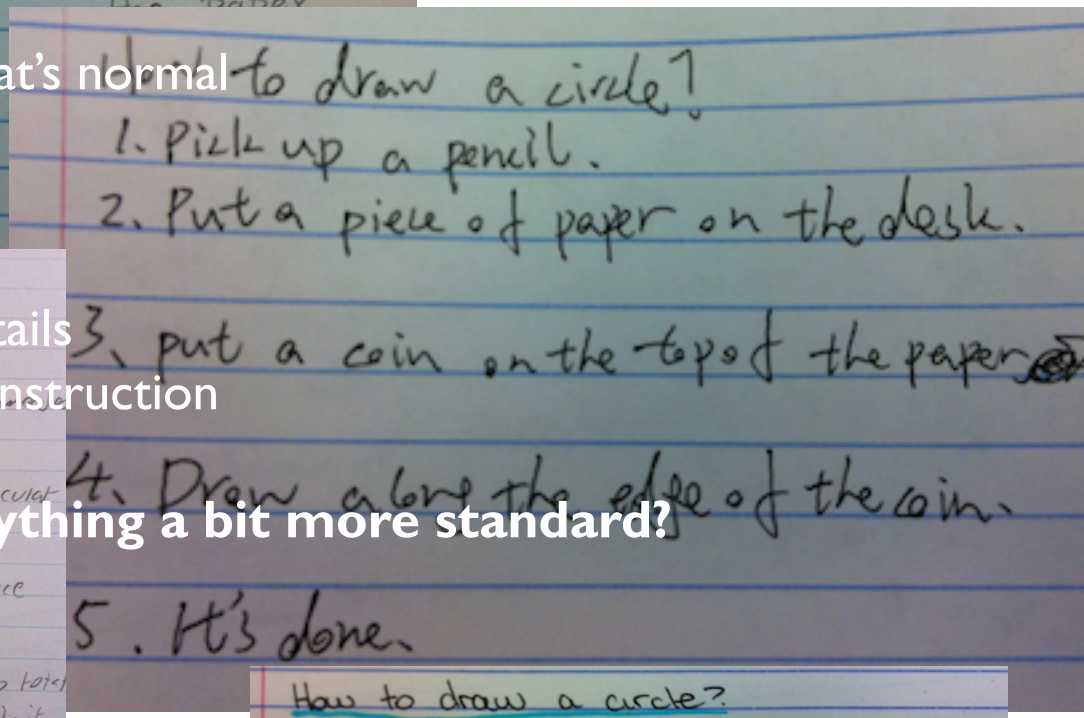
- **A** iClicker 1
- **B** iClicker 2
- **C** iClicker GO (does it finally work??)
- **D** Other
- **E** I don't have an iClicker

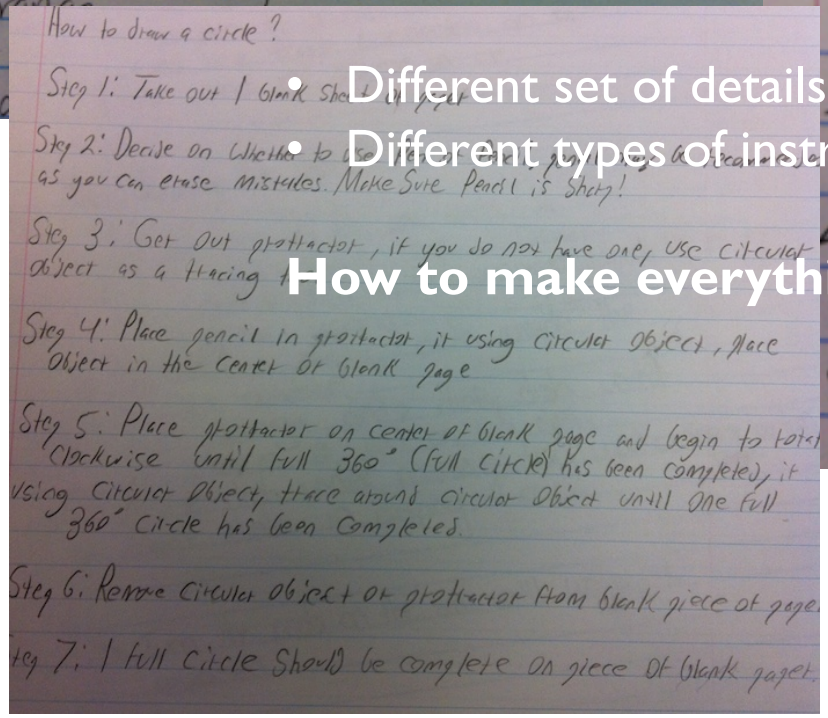
Describing algorithms in a rigorous way

WHAT IS PSEUDOCODE?

Many Ways to Draw Circles

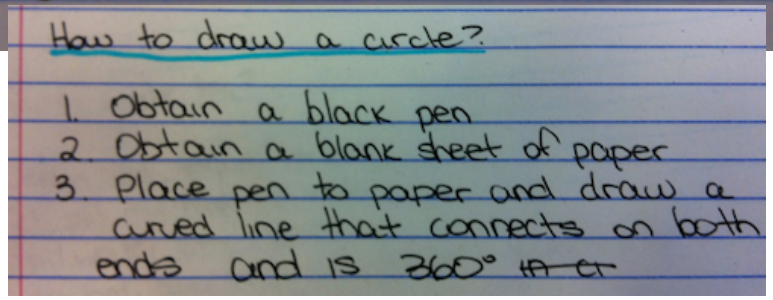






- Different methods: that's normal
- but also:
  - Different set of details
  - Different types of instruction

How to make everything a bit more standard?



Many Ways to Draw Circles

- How to make everything a bit more standard?
- **Better question:** why is there such a difference in length between some submissions and others?
- English language – in this case – is both:
  - ambiguous (you feel like you haven't said enough)
  - granular (you feel like you can always say more)
- Need to use another language!

What is "pseudocode"?

- Previous lecture: algorithms break down problems into chunks that are understandable by computer
- Pseudocode = language in which to describe algorithms when read by humans
- Pseudocode = language targeted at humans so can be free of complicated computer syntax
- People use pseudocode to describe algorithms

Algorithm: Simple Guessing 1

- A friend thinks of number between 1 and 10
- You need to guess it (no need to be efficient)

Idea

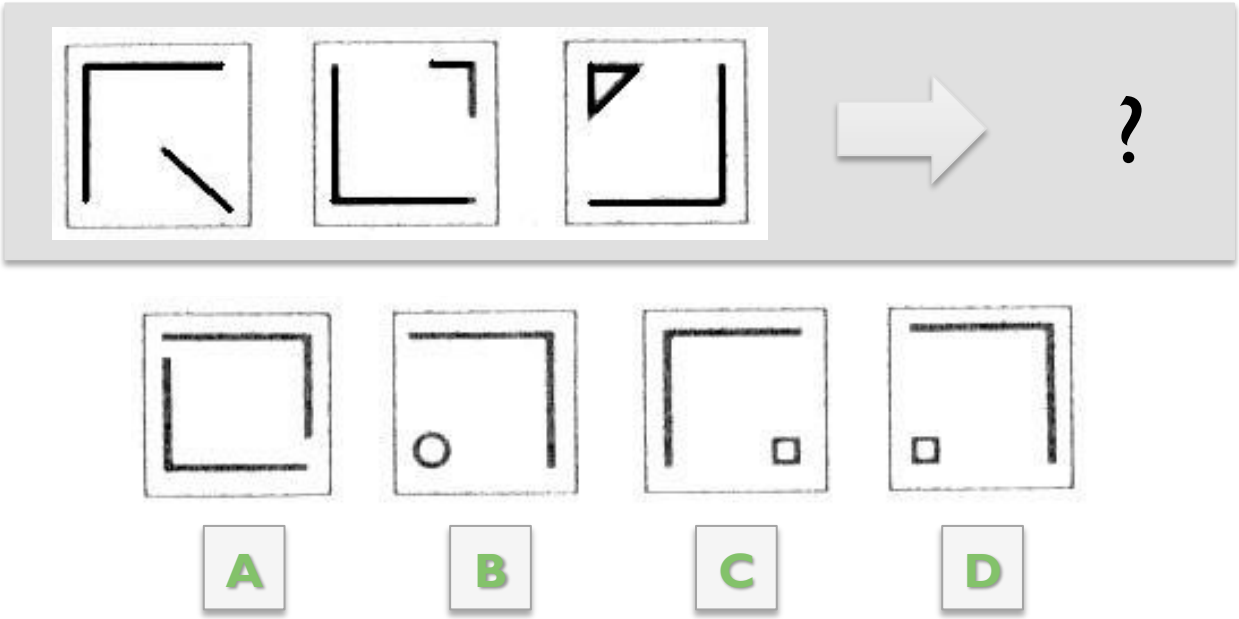
- Start with 1
- Guess that the number is 1
- If it is correct, hurray!
- If it is incorrect, try again with 2
- And so on, and so on

Question: Does this take a "finite amount of time", or can it go on forever? Why? If my friend thinks of 4 will it stop? 7? 11?

A YES, this will stop **B** NO, it is possible that this will go on forever

"And so on, and so on"

What is the next element in this sequence?



Guessing how a pattern continues can be difficult for us! And is impossible for computers…

Can you do better?

Previous example uses just a sequence of instructions. A lot is hidden in the "And so on, and so on."

Do it yourself 

- Take a piece of paper out
- Write out an algorithm using "blocks" and "parameters" and specifying the input and output

When you are done, use iClicker:

A I am done and I think I got it

B I am done, and I think I did not get it, or I gave up

Algorithm: Simple Guessing 2

Input: the range (smallest, largest) where number is
Output: the number of my friend

- Make one guess (*X*)
  1. Guess that the number is *X*
  2. If incorrect, then continue
  3. Else then stop algorithm and return *X* as answer

- Guess number (smallest, largest)
  1. For all numbers *K* between smallest and largest
     1. Make one guess (*K*)

A few important keywords to express algorithms

PSEUDOCODE KEYWORD

Some Statements

- Storing information (more variables later)
  - `set num1 to 10`
  - `set num2 to num1 + 10`
  - `set num3 to num1 x num2`
- Printing out text, reading a number
  - `print "Hello world!"`
  - `print "Error: number out of range!"`
- Query user (ask for input and store in variable)
  - `read a number into num4`
- Returning a result at the end of a function
  - `return myresult`
- Making comments that are not part of algorithm
  - `# anything after a "sharp" is ignored`

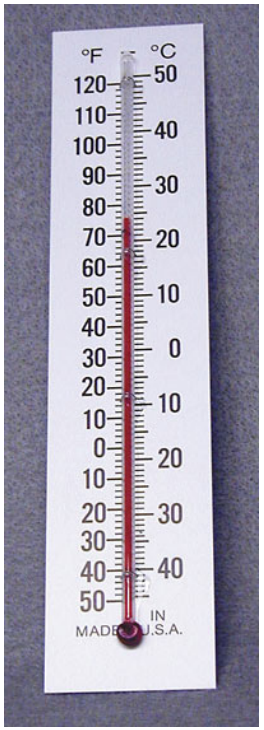
Example

To convert from Fahrenheit to Celsius: take the temperature in Fahrenheit, subtract 32, then divide by 9/5 ≈ 1.8

```
print "Enter a temperature in Fahrenheit:"
read number in tempInF

# Make the conversion and store it in "tempInC"
set tempInC to (tempInF - 32)/1.8

# Print it out
print "Your temperature in Celsius is:"
print tempInC
```



WARNING!!

- Pseudocode is informal
- What is important is: the base blocks that we use (assignment, printing, returning, etc.)
- What is not important: what words we use to describe them

Example

- for storing **we** use: `set myvar to 10`
- while others may use: `myvar := 10`
- or even: `myvar ← 10`

`if` statement: Making Choices

```
if <condition> then <what to do when true>
else <what to do when false>
```

- The `else` (called "alternative") can be omitted
- When there is a lot of things to do, we can use blocks, by indenting the code

```
if <condition> then
  <lots of stuff that gets done only when condition is true>
  <more stuff that still only gets done when condition is true>
else
  <stuff that gets done only when condition is false>
  <more of that stuff>
```

Example

```
print "Enter your grade (from A, B, C, D, E)"
read letter in studGrade

if studGrade == 'A' or studGrade == 'B' or studGrade == 'C' then
  print "You passed!!"
else
  print "You failed (but I'm sure you had a great instructor"
  print "who did his best to help you)."
```

Notice the condition: comparison using `==` for equality, and using `or` to test several possibilities at once.

Loops/iterations

- Sometimes need to **repeat** same instructions
  - because we need to make several copies (*for instance, to make several sandwiches*)
    - `do N times…`
  - because we need to repeat until a condition is reached (*drag the pen until the circle is complete*)
    - `while circle is not complete…`
  - because we need to examine all elements in a set or a sequence
    - `for each element x in the sequence S…`

- All types of loops are basically the same
  - the difference is convenience/practicality
  - don't worry if you don't understand what I mean
  - what this means for you is that there is no "right choice" for these iterations because sometimes can be said as conveniently with all types of loops

Example

Input: a set of integers S, which is set to { 10, 4, 3, 4, 15 }
Output: the average of S

```
set mynum to 0

# Add all elements of S into mynum
for each element x in S
  set mynum to (mynum + x)

# Print average
print mynum/5
```

Little exercise. Rewrite this algorithm so that it would work if S was of a different size.

A I am done and I think I got it

B I am done, and I think I did not get it, or I gave up

Example Modified

Input: a set of integers S
Output: the average of S

```
set mynum to 0
set total to 0

for each element x in S
  set mynum to (mynum + x)
  set total to (total + 1)

print mynum/total
```

Did you get it right?

A. Yes, I got it right!

B. No, I did not get it right, but I see how I could have done it

C. No, I did not get it right, and I don't think I could do it

Pacing and Understanding

How do you feel about pseudocode?

A. Too easy, this lecture is going too slow

B. The concept and examples are introduced at a good pace

C. Too fast, but I feel confident that I can go back over it myself

D. Too fast, and I need you to slow down

E. Pseudocode is desperately confusing, I feel so dumb ☹

Algorithm: ???

Input: *S* is a set of positive integer numbers

```
set current to -1

for each element x in S
do
  if x > current then
    set current to x

if current == -1 then
  return "SET IS EMPTY"
else
   return current
```

what does this algorithm do?

A Figured it out!!

B I am giving up, I don't understand at all

Practicing pseudocode and algorithms while fun

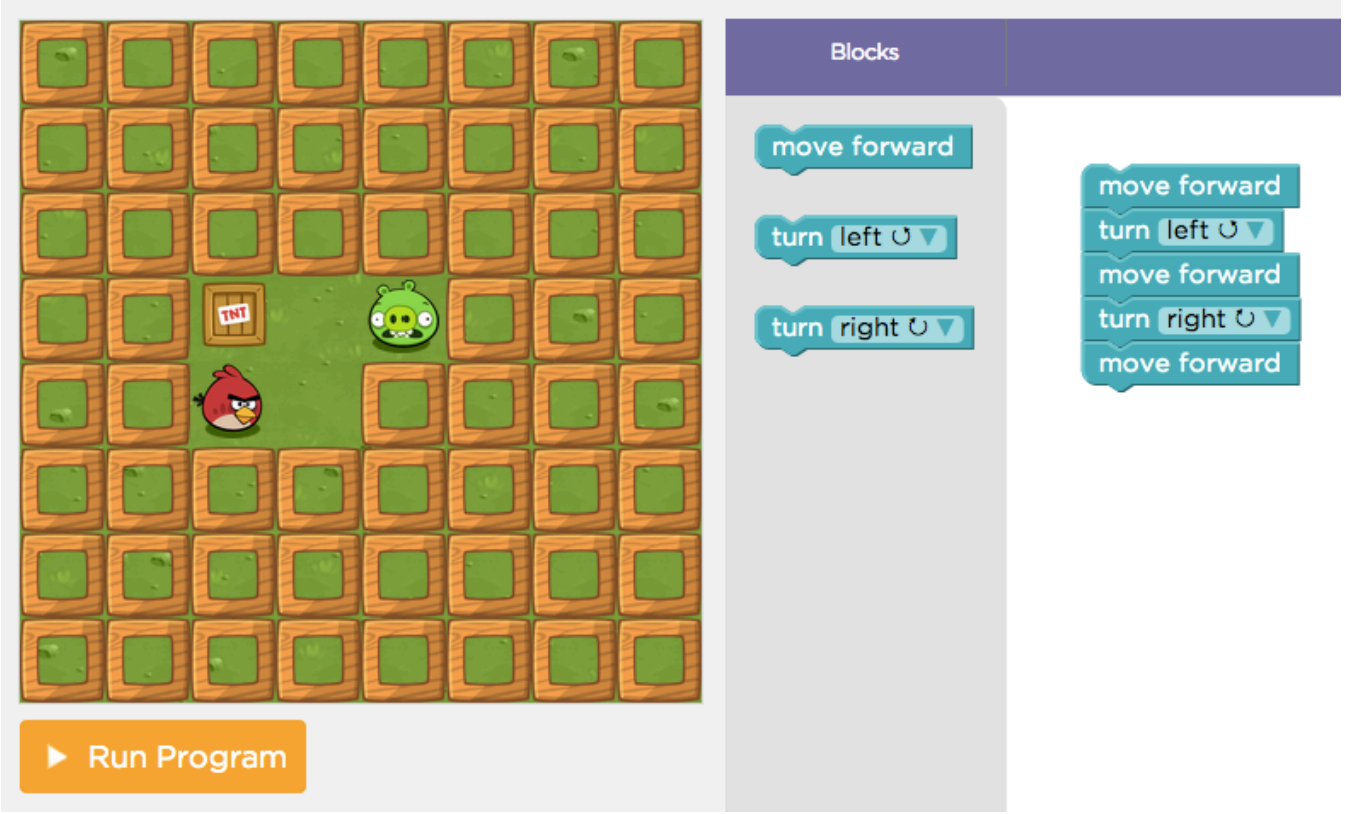
INTERACTIVE PRACTICE

"Hour of Code"

- Online exercises where you must build pseudocode using "prewritten blocks"
- Goal = guide character through maze
- Each set is supposed to take an hour
- This pseudocode can be translated into real actual code

"Hour of Code"

- Go: http://code.org/api/hour/begin/codeorg



Lightbot: Fun Puzzles

- Game to practice thinking about algorithms
- "*Make a robot light up on blue squares*"
- Not assignment, but I **strongly encourage** you to try this out in your own time (for instance over the week-end)
- http://light-bot.com/hoc.html