Everyday Algorithms

- **Lamp doesn’t work**
  - Lamp plugged in? (No → Plug in lamp)
  - Lamp plugged in? (Yes → Bulb burned out?)
    - Bulb burned out? (No → Buy new lamp)
    - Bulb burned out? (Yes → Replace bulb)

- **Key Steps**
  1. Mix sugar and butter
  2. Stir in 2 eggs
  3. Stir in flour
  4. Put in pans
  5. Bake for 10 mins
  6. Fork test: Clear? (Yes → End)
    - Fork test: Clear? (No → Bake a cake)

Adapted from Fluency with Information Technology, Lawrence Snyder, 4th & 5th ed (Chap s 10, 18, 19)  
Millersville University: UNIV 103  
Dr. Nazli Hardy
Algorithm

- Problem solving by breaking down a complex problem into manageable, logical, decision-based steps
- A precise, systematic method used to produce a specified result

- Derived from the name of mathematician (geographer/astronomer), Al Khwarizmi as “an effective method for solving a problem expressed as a finite sequence of steps”.
  - As opposed to logarithm

- We use them in everyday life
  - Recipes
  - Alphabetizing files/ CDs/ books
  - Learned – like arithmetic
  - Driving instructions
  - Looking up information (e.g. dictionaries, googling)
5 Essential Properties of Algorithms

1. Input specified
   - Data to be transformed during the computation to produce the output
   - Must specify type, amount, and form of data

2. Output specified
   - Data resulting from the computation—intended result
   - It is possible to have no result (especially in computing problems)
Five Essential Properties

3. Definiteness
   - Specify the sequence of events
   - Anyone and everyone else should be able to reproduce
   - Details of each step, including how to handle errors

4. Effectiveness
   - The operations are doable

5. Finiteness
   - Must eventually stop
   - (important especially in computing problems)
Flow Charts

Waking up routine of average students (and maybe professors without little children)

Start

Process step 1

Decision?

choice 1

Process step 2

choice 2

Process step n

End

Alarm Rings

Delay

Set for 5 Min.

Hit Snooze Button

Average 3 Times

Ready to Get Up?

Yes

Climb Out of Bed

End

Adapted from Fluency with Information Technology, Lawrence Snyder, 4th & 5th ed (Chaps 10, 18, 19)  Millersville University: UNIV 103  Dr. Nazli Hardy
Language in Algorithms

- Natural language
  - For people, we use a natural language like English
  - Ambiguity is common in natural language
  - Confusion even when you speak the same language
    - Why?

- Programming Language
  - Formal languages designed to express algorithms
  - Cannot afford ambiguity
  - Precisely defined; no ambiguity
    - E.g. Java, C++
The Context of a Program

- Program can fulfill 5 properties of an algorithm, be unambiguous, and still not work right because it is executed in the wrong context
  - e.g., last name/surname in Western countries means family name; in Asian countries it may be written first
  - First name vs. given name in South Asian countries

- **Context matters**: Driving instructions
  - "From the Limmat River go to Bahnhof Strasse and turn right."
  - Assumes you are traveling in a specific direction. If you are not, the directions will fail.
Programming and programmers ...
Overview: Programming Concepts

- Programming: act of formulating an algorithm or program

- Basic concepts have been developed over last 50 years to simplify common programming tasks

- Concepts we use today will be expressed in JavaScript
  - Fully general programming language
  - Designed for making active web pages/apps
The best way to program is to understand the basics and then to DIVE in!

- Names, values, variables
- Declarations
- Data types, numbers, string literals and Booleans
- Assignment
- Expressions
- Conditionals
Identifiers and Their Rules

- An Identifier is the character sequence that is a variable’s name
  ```javascript
  var us_president = BarackObama;
  ```

- Must be formed following specific rules
  - Must begin with a letter or underscore ( _ ) followed by any sequence of letters, digits, or underscore characters
  - Cannot contain spaces or dashes or punctuations
  - Case sensitive (Capitalization matters!)
  - Convention is to use lower case (as used in most high level language)
### Identifiers and Their Rules

<table>
<thead>
<tr>
<th>Valid</th>
<th>Invalid</th>
</tr>
</thead>
<tbody>
<tr>
<td>firstOne</td>
<td>1stOne</td>
</tr>
<tr>
<td>first1</td>
<td>first-1</td>
</tr>
<tr>
<td>first_1</td>
<td>first$1</td>
</tr>
<tr>
<td>first_One</td>
<td>first One</td>
</tr>
<tr>
<td>fIRSToNE</td>
<td>First1!</td>
</tr>
<tr>
<td>_special</td>
<td>5</td>
</tr>
<tr>
<td>very_long_name_ok</td>
<td>happy:)</td>
</tr>
</tbody>
</table>

For each invalid identifier, what rule is broken?
The Statement Terminator

- A program is a list of statements
  - Several statements may be run together on a line

- Each statement is terminated by the statement terminator symbol
  - In JavaScript, this is the semicolon ( ; )

- Forgetting the semi-colon is a very common error, so be warned (and wary)
Rules for Declaring Variables

- Undefined values
  - A variable may be declared, but may not yet have any value associated with it (its initial value)

\[
\begin{align*}
\text{var speed;} & \quad // \text{undefined initial value} \\
\text{speed} = 42; & \quad /* \text{now it has a value} */
\end{align*}
\]

Notice the use of commenting
Initializing a Declaration

- We can set an initial value as part of a declaration:
  - `var speed = 42;`

- Related variables may be grouped in one declaration-initialization; unrelated variables are usually placed in separate statements
  
  ```
  var numA = 27;
  var numB;
  var numC = 45;
  ```

- OR
  
  ```
  var numA = 27, numB, numC = 45;
  ```

- what programmers commonly do, but is not required, is called a *programming convention* e.g. variable names beginning with a lower case
3 Basic Date Types in JavaScript

- In JavaScript, values are grouped into related categories called **data types**, or just **types**

1. **numbers** (or numeric, for arithmetic)
2. **strings** (used for text)
3. **Booleans** (used for logic)
Strings

- Strings are sequences of keyboard characters

- Strings are always surrounded by single ( ' ' ) or double quotes ( " " )

- Strings can initialize a declaration
  - var hairColor = “black”, sign=‘Leo’, eyeColor = “brown”;
  - var greeting = “Hello, how are you today?”;
  - var x = 5, y = 6, z = 9;
  - var a = “123”;
  - var b = 123;

- Note that the string value “123” is not the same as the number value 123
**Escape Sequences**

Table 18.1 Escape sequences for characters prohibited from string literals

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Character</th>
<th>Sequence</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>\b</td>
<td>Backspace</td>
<td>\f</td>
<td>Form feed</td>
</tr>
<tr>
<td>\n</td>
<td>New line</td>
<td>\r</td>
<td>Carriage return</td>
</tr>
<tr>
<td>\t</td>
<td>Tab</td>
<td>'</td>
<td>Apostrophe or single quote</td>
</tr>
<tr>
<td>&quot;</td>
<td>Double quote</td>
<td>\</td>
<td>Backslash</td>
</tr>
</tbody>
</table>

```javascript
var fourTabs = " \t\t\t\t " (this variable represents 4 tab spaces)

var bothQuotesInOne = " \"\" (this variable represents "")
```
Boolean Values

Two logical values: true and false

```javascript
var foreignLangReq = false;
var mathReq = true;
var totGradCred = false;

var foreignLangReq = false, mathReq = true, totGradCred = false;
```
Assignments

- Programming is \textit{not} Algebra

- Consider this statement
  \[ x = x + 1; \]

  - In Algebra, this is impossible… no number is the same as one greater than itself

  - In programming, what we are actually saying is “\( x \) becomes \( x+1 \)” or “\( x \) is assigned the value \( x+1 \)” meaning take the value in \( x \), add 1, then store the new value back in \( x \)
Relational Operators

- Make comparisons between numeric values

- Outcome is a Boolean value, true or false
  - < less than
  - <= less than or equal to
  - == equal to (note difference between = and ==)
  - != not equal to
  - >= greater than or equal to
  - > greater than

- 5 < 10 evaluates to true

- 8.54 == 14.7 evaluates to false

- speed > 42 depends! (on______________________)
Logical Operators

- Used with Boolean values

- Logical *And*
  - Operator is `&&`
  - `age > 12 && age < 20`
  - Outcome of `a && b` is true if both `a` and `b` are true; otherwise it is false

- Logical *Or*
  - Operator is `||`
  - `age == 12 || age == 20`
  - Outcome of `a || b` is true if either `a` is true or `b` is true
  - Outcome of `a || b` is false if either `a` is true or `b` is false

- Logical *Not*
  - Operator is `!`
  - `!(age > 100)`
  - Unary operator. Outcome is opposite of value of operand
Recap …

```html
<html>
<head>, <title>, <body>
<form>

Declaration of variables
  type, name, value

Basic Algebra

Psedudocode
```
Example 1: Currency Conversion

<html>
<head>
<title>currency conversion</title>
<style type="text/css">
body {background-color: pink; font-family: helvetica; text-align: center}
</style>
</head>
<body>
<h1 style="color: white">the currency conversion calculator</h1>
<form>
<b>amount in usd:
  <input type=text name="amountUSD" size = 4 onChange = 'amountEuro.value = amountUSD.value * 0.75'>
<br>
amount in euro:
  <input type=text name="amountEuro" size = 4 onChange = 'amountUSD.value = amountEuro.value / 0.75'></form>
</body>
</html>
Example 2: Weight Conversion

<html>
<head>
<title>weight conversion page</title>
<style type = "text/css">
body {background-color: blue; font-family: helvetica; text-align: center}
</style>
</head>

<body>
<h1 style = "color : white">the weight conversion calculator</h1>
<hr>
<p>Nazli Hardy, UNIV 103, Example 2</p>
<form>
<b>
weight in kgs:
</b>
<br>

Fill in (write pseudocode)
</form>

</body>
</html>
Example 3: Temperature Conversion

```
<html>
<head>
<title> temperature conversion </title>
<style type = "text/css">
body {background-color: green; font-family: helvetica; text-align: center}
</style>
</head>

<body>
<h1 style = "color : white"> the temperature conversion calculator</h1>
<hr>
<p>Nazli Hardy, UNIV 103, Programming Example 3

<form> <b>
temperature in degrees Celsius:

Fill in (write pseudocode)

</form>

</body>
</html>
```
Sample Exam Question

Given that 1 USD = 0.75 Euros, which of the following correctly reflects the partial code for a form (as per class examples):

a. `<input type=text name = "amountUSD" size = 4 onChange = 'amountEuro.value = amountUSD.value / 0.75'>`

b. `<input type=text name = "amountUSD" size = 4 onChange = 'amountEuro = amountUSD / 0.75'>`

c. `<input type=text name = "amountUSD" size = 4 onChange = 'amountEuro.value = amountUSD.value * 0.75'>`

d. `<input type=text name = "amountUSD" size = 4 onChange = 'amountEuro = amountUSD * 0.75'>`

e. none of the above