Postage problem

- **Write a program that accepts the destination country (Canada or US) and the weight of a standard envelope and returns the needed postage.**

Weight	Canada	US
Up to 30g	$0.85	$1.20
Over 30g and up to 50g	$1.20	$1.80
Up to 100g	$1.80	$2.95
Over 100g and up to 200g	$2.95	$5.15
Over 200g and up to 300g	$4.10	$10.30
Over 300g and up to 400g	$4.70	$10.30
Over 400g and up to 500g	$5.05	$10.30

Postage data from Canada Post at https://www.canadapost.ca/cpo/mc/personal/ratesprices/postalprices.js on Sept. 2015

Effective starts to problem solving (from you)

- Handing typo on input (like Canada misspelled) and weight, individual entry or both input
- Range of the input
- Conditions to manage the table of postages
  - Max rate allowed
- What to do if conditions not in the table … error handling
- Noticing boundaries and we need to check on those
- Input format, eg. weight of 30.1 vs 30
- Which algorithm to use to fit the situation
  - ??? What is the situation
- How and where to store the table?

Problem Solving – Starting the process

- What comes into the program?
  - Do different data or modes need to be handled differently?
- What transformations do I need to make to the data?
  - Are there sub-problems or patterns that I can use?
- What part of the data is processed right away?
- What part of the data do I need to keep longer?
  - What tasks do I need to do to with that longer-term data?
    - How do I organize or store the data to make those tasks easy?
- What goes out of the program?
  - Do different data or modes need to be handled differently?

Problem Solving – Starting the process

- What assumptions can I make?
  - Are any given?
  - Can I reasonably make any of my own?
- What constraints exist?
- Are there strange cases to handle?
- What is important for the solution to do?

- Who are the users and how will they use it?
- What is the target environment?
- How stable are the requirements?

Postage Problem

- What are all the starting parameters for the postage problem?

Postage problem

- What comes in to the program?
  - Country and weight from the keyboard
  - The table of postage rates (never changes, so can be part of program itself).
- What transformations do I need to make to the data?
  - None
- What part of the data is processed right away?
  - Answer as soon as we get input
- What part of the data do I need to keep longer?
  - Nothing stored long-term…except the table of postage rates
- What goes out of the program?
  - The postage rate

Problem Solving – Starting the process

- What assumptions can I make?
  - The country and weight are given as integers
  - Just one query and we're done
  - ~~Weights are greater than zero~~
    - Part of what you expect for a solution, but not a reasonable assumption on input data
- What constraints exist?
  - Standard envelopes only
  - Only 2 countries to handle
- Are there strange cases to handle?
  - Country or weight outside the table
  - Boundary cases of weights
- What is important for the solution to do?
  - Nothing beyond the given output constraint

Problem Solving – Starting the process

- Who are the users and how will they use it?
  - Generally CS people, as an example program
- What is the target environment?
  - Basic command-line terminal. Should be portable across platforms as students may have Windows, Mac OCX, or Linux installed
- How stable are the requirements?
  - Just for a class example, so not expecting to change
    - Don’t need to plan to add more countries or more weight ranges in the near future

Evolution of solving problems

- Often follow a sequence of solutions
  - Can a computer solve the problem at all?
    - There are some problems that computers cannot solve
  - What is _a_ solution?
  - What is an efficient solution?
  - What is a practical solution?
  - What is a simple and practical solution?
  - What is an optimal solution?
  - What is a simple and optimal solution?
- Experience lets you start at different points in the sequence

Postage Problem

- How many different solution styles can you create?

Postage problem

- **Write a program that accepts the destination country (Canada or US) and the weight of a standard envelope and returns the needed postage.**

Weight	Canada	US
Up to 30g	$0.85	$1.20
Over 30g and up to 50g	$1.20	$1.80
Up to 100g	$1.80	$2.95
Over 100g and up to 200g	$2.95	$5.15
Over 200g and up to 300g	$4.10	$10.30
Over 300g and up to 400g	$4.70	$10.30
Over 400g and up to 500g	$5.05	$10.30

Postage data from Canada Post at https://www.canadapost.ca/cpo/mc/personal/ratesprices/postalprices.js on Sept. 2015

Postage Problem (class solutions)

Postage Problem

- The code must know that cases exist
  - Decide whether the cases appear in the code itself or in data structures that the code navigates.
    - Cases in the code: often easier to follow and ensure
    - Cases in data structures: easier to change or expand; more likely to treat the testing of all cases the same way

Encode Cases in the Code

- One independent “if” statement for each case
- Set of “if” statements and exploit previous failed tests using “else” clauses
  - “if” statements could be nested or not

Part data structure, part code

- Encode the boundaries in an array, search for the position in the array for the weight, and encode the answer for that solution into code

Data structures

- Use a data structure (two-dimensional array is enough) to store all of the rates.

Independent “if”

Get the country and weight

If (country is Canada and weight <= 30) report $0.85;

If (country is Canada and 30 < weight <= 50) report $1.20;

If (country is Canada and 50 < weight <= 100) report $1.80;

If (country is Canada and 100 < weight <= 200) report $2.95;

If (country is Canada and 200 < weight <= 300) report $4.10;

If (country is Canada and 300 < weight <= 400) report $4.70;

If (country is Canada and 400 < weight <= 500) report $5.05;

If (country is US and weight <= 30) report $1.20;

If (country is US and 30 < weight <= 50) report $1.80;

…

“if - else”

```
Get the country and weight
If (country is Canada and weight <= 30) report $0.85;
Else if (country is Canada and weight <= 50) report $1.20;
Else if (country is Canada and weight <= 100) report $1.80;
Else if (country is Canada and weight <= 200) report $2.95;
Else if (country is Canada and weight <= 300) report $4.10;
Else if (country is Canada and weight <= 400) report $4.70;
Else if (country is Canada and weight <= 500) report $5.05;
Else if (country is US and weight <= 30) report $1.20;
Else if (country is US weight <= 50) report $1.80;
…
```

“if - else” nesting

```
Get the country and weight
If (country is Canada) {
        If (weight <= 30) report $0.85;
        Else if (weight <= 50) report $1.20;
        Else if (weight <= 100) report $1.80;
        Else if (weight <= 200) report $2.95;
        Else if (weight <= 300) report $4.10;
        Else if (weight <= 400) report $4.70;
        Else if (weight <= 500) report $5.05;
} else if (country is US) {
        Else if (weight <= 30) report $1.20;
        Else if (weight <= 50) report $1.80;
        …
}
```

“if - else” deeper nesting

```
Get the country and weight
If (country is Canada) {
        If (weight <= 200) {                                    /* Canada and weight <= 200 */
                if (weight <= 50) {                             /* Canada and weight <= 50 */
                        if (weight <= 30) report $0.85          /* Canada and weight <= 30 */
                        else report $1.20                       /* Canada and 30 < weight <= 50 */
                } else {                                        /* Canada and 50 < weight <= 200 */
                        if (weight <= 100) report $1.80         /* Canada and 50 < weight <= 100 */
                        else report $2.95                       /* Canada and 100 < weight <= 200 */
                }
        } else {                                                /* Canada and weight > 200 */
                if (weight <= 400) {                            /* Canada and 200 < weight <= 400 */
                        if (weight <= 300) report $4.10         /* Canada and 200 < weight <= 300 */
                        else report $4.70                       /* Canada and 300 < weight <= 400 */
                } else report $5.05                             /* Canada and 400 < weight */
        }
} else if (country is US) {
                ...
}
```

2d-array for rate classes

Get the country and weight

boundaries = array with values 30, 50, 100, 200, 300, 500

Find index i such that boundaries[i-1] < weight <= boundaries[i]

rates = 2d array:

0.85, 1.20, 1.80, 2.95, 4.10, 4.70, 5.05

1.20, 1.80, 2.95, 5.15, 10.30, 10.30, 10.30

Report rates[country][i]

Switch solution

Get the country and weight

boundaries = array with values 0, 30, 50, 100, 200, 300, 500

Find index i such that boundaries[i] < weight <= boundaries[i+1]

/* We know that there are at most 7 rates, so combine the country and weight into one integer: 10's digit is country, unit digit is weight category. */

Class = country * 10 + i

```
Switch (class) {
        10: report $0.85
        11: report $1.20
        12: report $1.80
        …
        20: report $1.20
        21: report $1.80
        22: report $2.95
        …
}
```

Big table solution

```
// Create an array with 0 rows and 501 columns.  Each row corresponds to a country
// and each row gives the postage rate for each weight, in grams, of the envelope

rates = 2d array {
  { 0, 0.85, 0.85, …, 0.85, 1.20, 1.20, …, 1.20, 1.80, 1.80, …, 1.80, 2.95, … } ,
  { 0, 1.20, 1,20, …, 1.20, 1.80, 1.80, …, 1.80, 2.95, 2.95, …, 2.95, 5.15, …}
}

get country (0 for Canada, 1 for US)
get weight
return rates[country][weight];
```