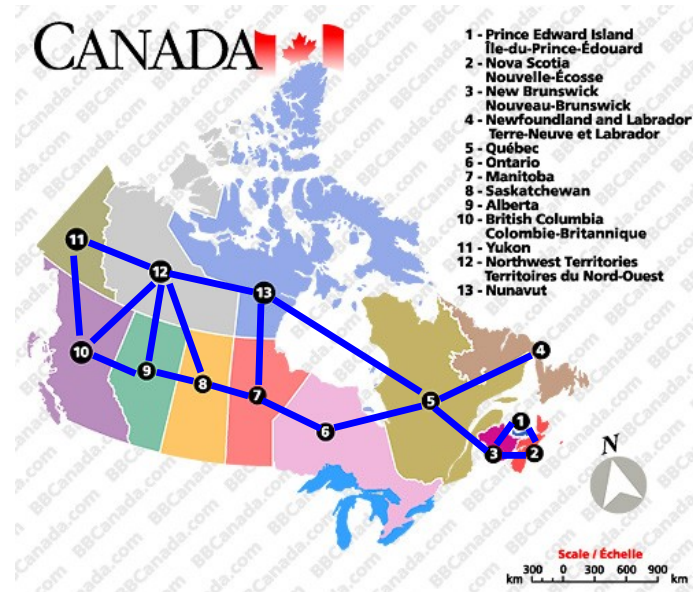### Stage B = Detailed Design of Solution

== Inputting Border Data ==

We can use the standard data-structure called a "map diagram" for entering border data.
The user records only **connections** between regions, rather than borders and large regions, like these blue lines:

Source: BBCanada http://www.bbcanada.com/bb_canada_map.cfm accessed 18 Feb 2011
'Maps courtesy of BBCanada.com, www.BBCanada.com '

Each line segment connects two neighboring regions. Notice that diagonal neighbors, e.g. #8 and #13, may have the same color.

For each region we only need a list of the connected neighbors. The data can be written like this:
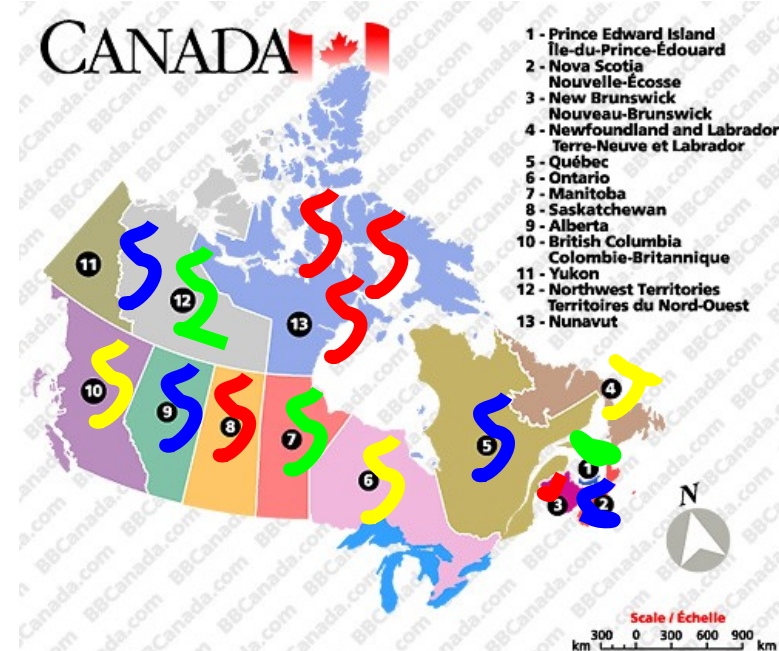
```
YU --> BC,NW
BC --> YU,NW,AL
NW --> YU,NU,BC,AL,SA
AL --> BC,NW,SA
SA --> AL,NW,MA
NU --> NW,MA,QU
MA --> SA,NU,ON
ON --> MA,QU
QU --> ON,NU,NF,NB
NF --> QU
NB --> QU,NS,PE
NS --> NB,PE
PE --> NB,NS
```

The user will actually type the data with commas rather than arrows (see example below).

== Displaying Results ==

Since the goal of the program is to enable students to color in maps, they don't really need an actual graphical print-out. All they need to know is which color to put in each region, like this:

Printed list of Colors	Students will color the map like this:
YU = blue BC = yellow NW = green AL = blue SA = red NU = red MA = green ON = yellow QU = blue NF = yellow	 1 - Prince Edward Island Île-du-Prince-Édouard 2 - Nova Scotia Nouvelle-Écosse 3 - New Brunswick Nouveau-Brunswick 4 - Newfoundland and Labrador Terre-Neuve et Labrador 5 - Québec 6 - Ontario 7 - Manitoba 8 - Saskatchewan 9 - Alberta 10 - British Columbia Colombie-Britannique 11 - Yukon 12 - Northwest Territories Territoires du Nord-Ouest 13 - Nunavut

Source: BBCanada

http://www.bbcanada.com/bb_canada_map.cfm accessed 18 Feb 2011

'Maps courtesy of BBCanada.com, www.BBCanada.com '

== Data Storage ==

In summary, the program needs two lists of data:

Borders (the first region in each line borders the others)	Colors
YU,BC,NW BC,YU,NW,AL NW,YU,NU,BC,AL,SA AL,BC,NW,SA SA,AL,NW,MA NU,NW,MA,QU MA,SA,NU,ON ON,MA,QU QU,ON,NU,NF,NB NF,QU NB,QU,NS,PE NS,NB,PE PE,NB,NS	YU,blue BC,yellow NW,green AL,blue SA,red NU,red MA,green ON,yellow QU,blue NF,yellow NB,red NS,blue PE,green

The Borders data should be stored in a data file, as it might be re-used or modified. The Colors data will be produced automatically by the program, so it needn't be stored in a data file.

== Processing ==

The most important part of the program is the automation that chooses appropriate colors, in accordance with the Borders data and following the rules outlined in the Success Criteria - e.g.:

- neighboring regions must have non-matching colors
- neighbors that meet only at a vertex may have the same colors
- only 4 colors should be used
- fewer than 4 colors should be used if possible

There are two possible strategies:

1. follow an algorithm that automatically produces a successful coloring plan

   - OR -

2. assign colors randomly and check whether the set of colors is acceptable (following the rules above) - if the colors don't work, then repeat with a different random set, until a successful set is found (or quit after 1000 tries)

The first strategy is what people try to do when they are searching for a coloring plan. I am unaware of a straightforward algorithm that works for every map. So I will use the random guess and check strategy. This flow-chart outlines the overall logic of the program:

== Development Plan ==

The programming can be divided into 4 versions, adding new features in each version.

#1 - Detecting Incorrect Colors (1 week)

- Create a list of Borders data and a single list of colors
- Check whether the colors work by
- Comparing colors of all neighbors by scanning through the Borders data

```
pseudocode for Checking
    SUCCESS = True
    for each REGION in the Borders list
        for each NEIGHBOR of the REGION
            look up REGION.COLOR in the Colors list
            look up NEIGHBOR.COLOR in the Colors list
            if REGION.COLOR == NEIGHBOR.COLOR
                SUCCESS = False
    return SUCCESS
```

#2 - Generating Sets of Colors (1 week)

- Generate a random set of colors
- Repeat assigning colors randomly until a correct coloring plan is found

```
pseudocode for RandomColors
    Colors = empty list
    for each REGION in the Borders list
        select a random COLOR 1..4 (or 1..3 if max-colors is 3)
        record the name of the REGION and the COLOR in the Colors list
```

#3 - Inputting Borders Data from a File (1 week)

Read Borders data from a data file, allowing the user to type the data into a text file, without needing to type directly into the programming code.

```
pseudocode for LoadingBordersData
    COUNT = 0
    Borders = empty list
    open data file
    repeat until end of file
        info = readLine   (e.g.  BC,YU,NW,AL)
        split info into array of Strings --> data[]
        append data[] to the Borders array
```
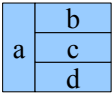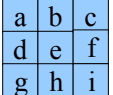
#4 - Final Program (2 weeks)

A totally complete final program should implement higher-level logic as described in the flowchart above. This must include ample testing.
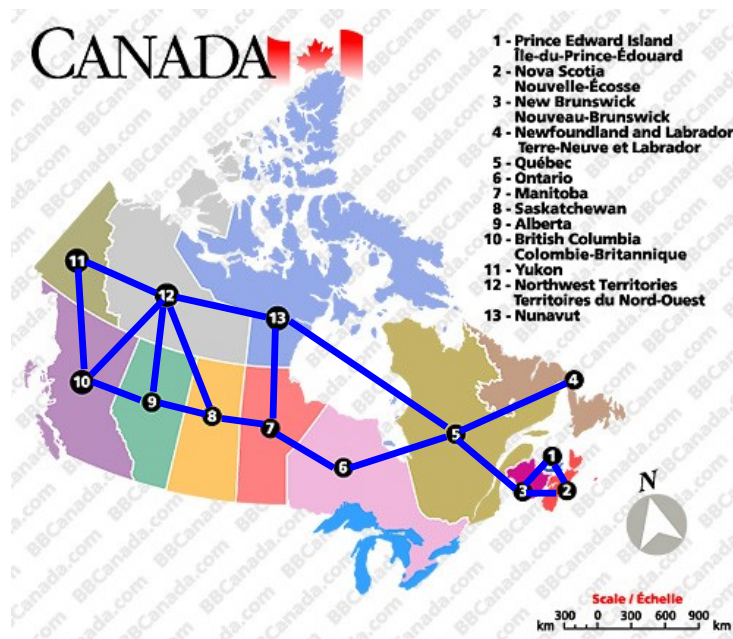
#5 - Final Testing (1 week)
This stage includes the end-user testing the final program.

== Testing Plan ==

Here are 3 BORDER lists for testing the program, together with a successful coloring plan and a defective coloring plan. Other defective colorings can easily be produced for testing.

Flag	**3x3 square grid**	**Canada** (see map below)

Flag

	Colors 1,2,3	
Borders	good	bad
a : b , c , d	1	1
b : a , c	2	2
c : a , b, d	3	3
d : a , c	2	1

3x3 square grid

	Colors 1,2,3,4	
Borders	good	bad
a : b , d	1	1
b : a , c , e	2	2
c : b , f	3	3
d : a , e , g	4	3
e : b , d , f , h	1	2
f : c , e , i	2	1
g : d , h	3	1
h : e , g , i	4	2
i : f , h	1	3

Canada (see map below)

	Colors=1,2,3,4	
Borders	good	bad
YU:BC,NW	3	3
BC:YU,NW,AL	4	4
NW:YU,NU,BC,AL,SA	2	3
AL:BC,NW,SA	3	2
SA:AL,NW,MA	1	1
NU:NW,MA,QU	1	1
MA:SA,NU,ON	2	2
ON:MA,QU	4	4
QU:ON,NU,NF,NB	3	3
NF:QU	4	4
NB:QU,NS,PE	1	1
NS:NB,PE	3	3
PE:NB,NS	2	2

1 - Prince Edward Island
    Île-du-Prince-Édouard
2 - Nova Scotia
    Nouvelle-Écosse
3 - New Brunswick
    Nouveau-Brunswick
4 - Newfoundland and Labrador
    Terre-Neuve et Labrador
5 - Québec
6 - Ontario
7 - Manitoba
8 - Saskatchewan
9 - Alberta
10 - British Columbia
     Colombie-Britannique
11 - Yukon
12 - Northwest Territories
     Territoires du Nord-Ouest
13 - Nunavut

 Source: BBCanada http://www.bbcanada.com/bb_canada_map.cfm accessed 18 Feb 2011
'Maps courtesy of BBCanada.com, www.BBCanada.com '

After stage 1, the sample coloring data is no longer needed, as the program will produce the colors automatically.

Test plan

Test Type	Nature of test	Example
Program opens to prompt for filename that exists	Double click on colors file icon	CANADA
Program inputs data from data file (#1)	Enter file name and see if program runs	
Test with simple data (#2)	On successful running of program check visually	Use example where only 3 colors are necessary
Test with inappropriate data (#4)	Check program does not run	Too much data may cause program not to complete
neighboring regions must have non-matching colors (#3)	On successful running of program check visually	
neighbors that meet only at a vertex may have the same colors (#3)	On successful running of program check visually	
only 4 colors should be used (#3)	On successful running of program check visually	
fewer than 4 colors should be used if possible (#3)	On successful running of program check visually	