# CSc 111 Assignment 9:
# File I/O

### *Learning Outcomes:*

When you have completed this assignment, you should understand:

- How to design functions that read/write formatted input/output from/to files

### *How to hand it in:*

Submit your file named `assignment9.c` through the Assignment 9 link on the CSC111 conneX site.

### *Grading:*

- Late submissions will be given a **zero grade**.
- You must use the file `assignment9.c` provided to write your solution. Changing the filename or any of the code given in the file will result in a **zero grade**.
- Your function names must match exactly as specified in this document or you will be given a **zero grade**.
- We will do **spot-check grading** in this course. That is, all assignments are graded BUT only a subset of your code might be graded. You will not know which portions of the code will be graded, so all of your code must be complete and adhere to specifications to receive marks.
- Your code must run without errors on the ECS Lab machines or a **zero grade** will be given.
- It is the responsibility of the student to submit any and all correct files. Only submitted files will be marked. Submitting an incorrect file is not grounds for a regrade.
- If the assignment requires the submission of multiple files, then all files must be submitted.

**Marks will be given for…**

- your code producing the correct output
- your code following good coding conventions
    - Proper indentation
    - Documentation above each function
    - Names of variables should have meaning relevant to what they are storing
    - Appropriate use of named constants
    - Use of whitespace to improve readability
    - Proper use of variables to store intermediate computation results
    - Appropriate use of helper functions

### *Get started:*

- Download `assignment9.c` from conneX and save it to your computer
- Create a CLion project with `assignment9.c` as the source
- Below are 7 function specifications. For each specification
    - The `assignment9.c` file contains the function prototype.
    - Implement and test the incomplete functions
    - We suggest you write a helper method to print out the contents of your 2D array to help with the testing of your `to_array` function.
    - We have provided you with 4 sample input and corresponding output file to help with your understanding and testing of the problem. Download the input files to the directory to allow your program to read them and compare your results to the given output file.

## *Background Information:*

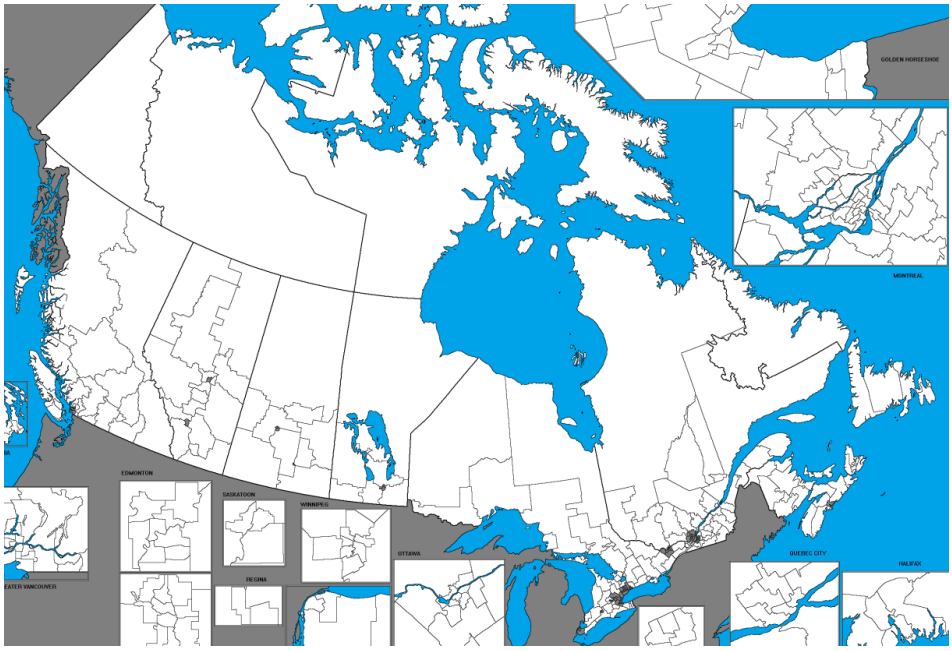
You are going to design a program to help calculate voting results in Canada. This page gives you some background on the voting process in Canada.

- Canada is split into 338 geographical ridings (regions), as shown on the map below.
- There are 4 major political parties: NDP, Conservative Party (CP), Liberal Party (LP) and Green Party (GP).
- In order for a party to win a federal election and to have their party leader become the Prime Minister, that party must ***win more ridings*** than any other party.
    - To win a riding, a party ***must get more votes in that riding*** than any other party.

***For example:***
Suppose one of the 338 ridings had the following party votes:
NDP: 9,880 votes
Conservative Party: 12,782 votes
Liberal Party: 11,542 votes
Green Party: 7,600 votes
**The Conservative Party would win this particular <u>riding</u>.**

Suppose the parties won the following number of ridings (out of 338 ridings) in the election:
NDP: 50 ridings
Conservative Party: 100 ridings
Liberal Party: 138 ridings
Green Party: 130 ridings
**The Liberal Party would win the <u>election</u>.**

Suppose across all 338 ridings, the parties had following votes:
NDP: 120,000 votes
Conservative Party: 225,000 votes
Liberal Party: 340,000 votes
Green Party: 360,600 votes
**The Green Party would have won the <u>popular vote</u> (this doesn't get them elected).**



### *Function Specifications:*

**`to_array`**

This function takes the name of an input file as a null terminated string and a 2D array of integers as the destination. The function should read and store the data from the input file into the corresponding rows/columns of the destination array. The function should return -1 if the given filename cannot be opened successfully, otherwise it should return the number of rows read from the file into the destination array (this number will be <= MAXROWS).

If the file exists, it is guaranteed to have some number of rows between 0 and MAXROWS, where each row represents a vote by one single voter. Each row will have 3 integers be in the following format:

- Column 1: unique voter id (one per person)
- Column 2: the number corresponding to the riding this voter is voting in
  (i.e., an integer between 0 and 337)
- Column 3: the number corresponding to the party this voter voted for
  (i.e., an integer between 0 and 3) where:
  - 0 means the NDP party
  - 1 means the Conservative party
  - 2 means the Liberal party
  - 3 means the Green party

Example:
If the input file had the following text in it:

```
34564  17   1
37897  308  3
41230  104  0
```

When this function is complete, the 2D destination array will contain the following values and it will return a value of **3** (there are 3 rows in the destination array):

```
{ {34564, 17, 1},
  {37897, 308, 3},
  {41230, 104, 0} }
```

**`compute_riding_results`**

This function takes three arguments: a 2D input array of integers holding voter data, a 2D integer array to hold results in and the number of row of voter data in the input array.

The input array is in the following format:

- it has a row for each vote
- each row has the following 3 columns:
  - a unique voter ID,
  - the riding number the voter voted in
  - the party number the voter voted for

After the function completes, the result array should be in the following format:

- it has a row for each of the 338 ridings (0 through 337)
- each row has a column for each party:
  - Column 0 is the number of votes for NDP
  - Column 1 is the number of votes for Conservative
  - Column 2 is the number of votes for Liberal
  - Column 3 is the number of votes for Green
- The cells should contain the total number of votes in the input array that were made for the corresponding party(column) in the corresponding riding (row)

Example:
The following is a small example to help you understand the problem. Your function will be tested with additional tests with larger amounts of data.

| `data` | `result` | **Explanation** |
|---|---|---|
| 40601 0 3<br>15664 1 0<br>21298 1 2<br>78710 2 0<br>27598 1 3<br>45218 1 0 | 0 0 0 1<br>2 0 1 1<br>1 0 0 0<br>// the remaining 334 rows will contain all zeros as there were not votes in those ridings yet | `data`:<br>Voter #40601 voted for Green party in riding #0<br>Voter #15664 voted for NDP party in riding #1<br>Voter #21298 voted for Liberal party in riding #1<br>Voter #78710 voted for NDP party in riding #2<br>Voter #27598 voted for Green party in riding #1<br>Voter #45218 voted for NDP party in riding #1<br><br>`result`:<br>Riding 0 had:<br>- 0 votes for NDP<br>- 0 votes for Conservative<br>- 0 votes for Liberal<br>- 1 votes for Green<br>Riding 1 had:<br>- 2 votes for NDP<br>- 0 votes for Conservative<br>- 1 votes for Liberal<br>- 1 votes for Green<br>Riding 2 had:<br>- 1 votes for NDP<br>- 0 votes for Conservative<br>- 0 votes for Liberal<br>- 0 votes for Green |

### `index_of_max`

This function takes an array of integers and the length of the array and returns the index of the maximum value in the array. If there is more than one index with the maximum value, the function should return -1. You can assume the length is greater than or equal to 1.

Examples:
If `index_of_max` is called with array: `{-3,-2,-1,-5}` and length: `4`
it will return: `2` since -1 is the largest value and it is at index 2 of the array

If `index_of_max` is called with array: `{3,4,1,6,1}` and length: `5`
it will return: `3` since 6 is the largest value and it is at index 3 of the array

If `index_of_max` is called with array: `{3,4,1,6,1,6}` and length: `6`
it will return: `-1` since 6 is the largest value and it is at index 3 and 5 of the array

### `calculate_winner`

This function takes 2 arguments: a 2D input array of integers and a 1D output array of integers. The function should determine and store the winner of each riding in the output array. The function should also count the number ridings each party won and the party that won in the most ridings is the overall winner. The function should return the corresponding number of the party that is the overall winner. If there is a tie for the winner, the function should return -1.

The input array is in the following format:

- it has a row for each of the 338 ridings (0 through 337)
- each row has a column for each party:
  - Column 0 is the number of votes for NDP
  - Column 1 is the number of votes for Conservative
  - Column 2 is the number of votes for Liberal
  - Column 3 is the number of votes for Green
- The cells contain the total number of votes in the input array that were made for the corresponding party(column) in the corresponding riding (row)

After the function completes, the result array should be in the following format:

- it has an index for each of the 338 ridings (0 through 337)
- each index will hold a number representing the winner of the corresponding riding:
  - the number representing the party with the most votes
  - -1 if there are two parties with the most votes

**Example:**

| | `winners_per_riding` | |
|---|---|---|
| `counts_per_riding` | **values** | **Explanation…** |
| 10021 16876 18768 12892<br>27897 18876 23909 20868<br>13897 10832 12909 22876<br>14887 11212 11459 14887<br>… | 2<br>0<br>3<br>-1<br>… | From looking at each row of `counts_per_riding` …<br>Liberal (2) wins (column 2 of row 0 is the biggest (18768 votes))<br>NDP (0) wins (column 0 of row 1 is the biggest (27897 votes))<br>Green (3) wins (column 3 of row 2 is the biggest (22876 votes))<br>No winner (columns 0 and 3 of row 3 are the biggest (14887 votes))<br>… |
| The function returns the number of the party having the most wins in `winners_per_riding`. | | |

**`calculate_popular_vote`**

This function takes a 2D input array of integers that contains the vote counts for each party in each riding. The function should compute the sum of votes for each party across all ridings and determines the winner as the party with the largest number of votes. The function will return the number corresponding to the winning party or -1 if there are two parties with the most votes.

The input array is in the following format:

- it has a row for each of the 338 ridings (0 through 337)
- each row has a column for each party:
  - Column 0 is the number of votes for NDP
  - Column 1 is the number of votes for Conservative
  - Column 2 is the number of votes for Liberal
  - Column 3 is the number of votes for Green
- The cells contain the total number of votes in the input array that were made for the corresponding party(column) in the corresponding riding (row)

**Examples:**

| `counts_per_riding` | **Explanation** |
|---|---|
| 0 0 0 1<br>2 0 1 1<br>1 0 0 0<br>// to simplify the example, the remaining 334 rows contain all zeros as there were not votes in those ridings yet | The function should return 0 (for NDP) since NDP received the most total votes, with the vote totals being:<br>- 3 for NDP column<br>- 0 for Conservative column<br>- 1 for Liberal column<br>- 2 for Green column |

| `counts_per_riding` | **Explanation** |
|---|---|
| 0 0 0 1<br>2 0 1 1<br>1 0 0 1<br>// to simplify the example, the remaining 334 rows contain all zeros as there were not votes in those ridings yet | The function should return -1 since there was a tie between the NDP and Green parties for the most total votes, with the vote totals being:<br>- 3 for NDP column<br>- 0 for Conservative column<br>- 1 for Liberal column<br>- 3 for Green column |

**`to_party_name`**

This function takes 2 arguments: a 1D char array as the destination array an integer representing the party. You can assume the number representing the party will be one of: -1, 0, 1, 2 or 3. The function should copy the following strings to the destination array.

- -1, use string: "NOT NAMED"
- 0, use string: "NDP"
- 1, use string: "CONSERVATIVE"
- 2, use string: "LIBERAL"
- 3, use string: "GREEN"

When the function is complete, the destination array should contain the correct NULL terminated string. You MUST use the EXACT strings provided above (case, spelling and spacing) or you will receive zero on this function.

**`to_file`**
This function takes 4 arguments (described below) and writes the election results to the given filename. The results should be written in the following order (see the sample output provided for an example):

- The winner of the election
- The winner of the popular vote
- A table reporting the votes per riding with
    - A header row with column headings
    - A row for each riding (0 through 337) that contains the following on each row:
        - The riding number
        - The number of votes for NDP
        - The number of votes for Conservative
        - The number of votes for Liberal
        - The number of votes for Green
        - The name of the winning party which will be one of:
            - "NOT NAMED"
            - "NDP"
            - "CONSERVATIVE"
            - "LIBERAL"
            - "GREEN"

**Argument descriptions**:

- `filename`: a char array holding a NULL terminated string that is the name of the file to write the results to
- `counts_per_riding`: a 2D integer array holding the vote counts for each party in each riding
    - it has a row for each of the 338 ridings (0 through 337)
    - each row has a column for each party:
        - Column 0 is the number of votes for NDP
        - Column 1 is the number of votes for Conservative
        - Column 2 is the number of votes for Liberal
        - Column 3 is the number of votes for Green
    - The cells contain the total number of votes in the input array that were made for the corresponding party(column) in the corresponding riding (row)

- `election_winner`: a NULL terminate string holding the name of the winner of the election which is one of:
    - "NOT NAMED"
    - "NDP"
    - "CONSERVATIVE"
    - "LIBERAL"
    - "GREEN"
- `popular_winner`: a NULL terminate string holding the name of the winner of the popular vote in the election which is one of:
    - "NOT NAMED"
    - "NDP"
    - "CONSERVATIVE"
    - "LIBERAL"
    - "GREEN"