Introduction

When considering diseases in the world, diabetes is one of the most prevalent, and affects the livelihoods of millions. Furthermore, the aim of this report is to present various findings of an analysis of the prevalence of diabetes in Canada's four most populated provinces. These provinces being, Ontario, Quebec, British Columbia as well as Alberta (*CPS188 : Term Project : Winter 2023* n.d.). Moreover, the analysis that is conducted is based around the real data collected by Statistics Canada from the time period of 2015 to 2021 (*CPS188 : Term Project : Winter 2023* n.d.). When tackling this report and its coding process, the primary objective was to use C programming language to read data provided from a CSV file and then conduct several computations.

Moving on, when completing the computations, the relevant columns of the data file are REF_DATE (year), GEO (country/province), Sex (Females/Males), Age group (35-49 years, 50-64 years, 65 years and over), and VALUE (the % of the population that has diabetes) (*CPS188 : Term Project : Winter 2023* n.d.). In addition, this report is constructed in such a manner to allow for a greater and clear understanding of the data analysis process. This process ranges from data processing, data visualization and data analysis as well. As expected, the report also utilizes various graphs and tables to assist in the explanation of the code and thought process taken place.

Alongside the use of the C programming language, GNUPlot and its functionalities are also used to create the tables and graphs (*CPS188 : Term Project : Winter 2023* n.d.).

Lastly, this project and report aims to be useful for anyone interested in understanding the prevalence of diabetes in Canada and the possible implications it may present to the health system.

Discussion of Calculations

Part 1.

Section (A and B) output:

```
Total Provincial and National percent averages:
Ontario: 11.70%
Quebec: 10.45%
British Columbia 9.67%
Alberta: 10.86%
National: 10.87%
```

This part of the code and its output is aimed to answer those questions presented in section A and B. As seen through the output for the total provincial and national percent averages, it is evident that although the numbers for each province are similar to one another, slight discrepancies are still present. However, to firstly discuss the results. The provincial averages for the percentage of the population diagnosed with diabetes in Canada are as follows: Ontario has a total provincial average of 11.70%, Quebec with 10.45%, British Columbia with 9.67%, and finally Ontario with the highest provincial percent average of 10.86%. In addition, all of the values for this output are calculated with the data from all year and age groups.

The process of this code was done by doing Part D first, which saved the age groups and the corresponding years to arrays. So to get the total average which was needed for this part, there was a for loop that was made to go through the elements of all these arrays and add them to a fourth array which contains the entirety of the values respective to each location. After the arrays which contained all the values were created, they were put through the `get_average` function which then calculated the average for the arrays. Finally, the values were printed in a tabulated format.

```
--------------------------------------------------
The Provincial and National averages for 2015:
Ontario: 10.77%
Quebec: 10.90%
British Columbia: 9.30%
Alberta: 9.32%
National: 10.60%
--------------------------------------------------
The Provincial and National averages for 2016:
Ontario: 12.20%
Quebec: 9.82%
British Columbia: 8.63%
Alberta: 9.77%
National: 10.70%
--------------------------------------------------
The Provincial and National averages for 2017:
Ontario: 11.98%
Quebec: 9.58%
British Columbia: 10.14%
Alberta Average for 2017: 11.97%
National Average for 2017: 10.95%
--------------------------------------------------
The Provincial and National averages for 2018:
Ontario : 11.28%
Quebec: 10.65%
British Columbia: 8.52%
Alberta: 11.02%
National: 10.78%
--------------------------------------------------
The Provincial and National averages for 2019:
Ontario: 13.03%
Quebec: 10.48%
British Columbia: 11.44%
Alberta: 11.33%
National: 11.70%
--------------------------------------------------
The Provincial and National averages for 2020:
Ontario: 11.17%
Quebec: 11.42%
British Columbia: 9.04%
Alberta: 12.88%
National: 10.60%
--------------------------------------------------
The Provincial and National averages for 2021:
Ontario: 11.48%
Quebec: 10.47%
British Columbia: 11.70%
Alberta: 9.82%
National: 10.75%
```

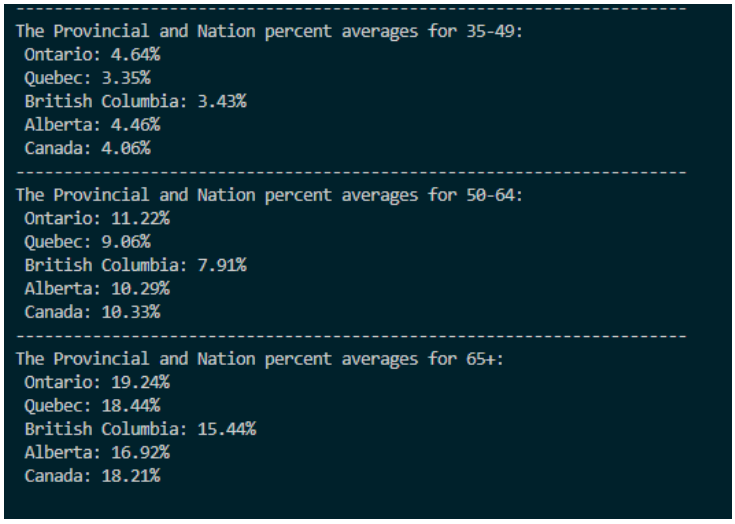
Section (C) output:

The aim of this section of the code and output is to highlight and show the provincial and national averages in Canada between 2015 to 2021. From this vast range of computational results, many educated observations can be made. Firstly, there are several interesting variations in diabetes diagnosis rates across the provinces in Canada. Most notably, Ontario was able to consistently present higher provincial averages than the national average in every single year between 2015-2021 except for 2016. Furthermore, Quebec's provincial average through the years fluctuated between higher-than-average some years and lower-than-average in others. On the other hand, the province of British Columbia consistently had lower-than-average percentages. This trend in British Columbia is not an anomaly as the province also had the lowest percentage by far in the results of section 1a) of this report. Moreover, Alberta presented higher-than-average percentages in every year between 2015-2021, except for 2015. However, the trend that would likely be of the most importance to statisticians, the government of Canada and the healthcare system is one that shows an overall steady increase in the provincial averages of diabetics from 2015-2021. To provide an

example, the national average for diabetics in Canada increased from 10.60% in 2015 to 10.75% in 2021. Although this difference may not seem rather large, it can be of use to the healthcare industry as it can allow for more precise and accurate approaches to rectifying diabetes in Canada.

The way the code was processed to find out the information needed to calculate the averages is through a positioning system. Part D was done first in this project, so there were 15 arrays that contained the values for each location in association with each respective age group. For example: Ontario with the age groups of 35 - 49. This meant that in those arrays each index referenced a specific year. So, in index 0, it was 2015, index 1 was 2016, index 2 was 2017 and so on. Using this setup, the corresponding indexes for all the age groups associated with a certain province were collected and added to an array. For example, the index 0 of the array `canada35`, `canada50 & canada65` were added to the array `canada2015`. This was done for all years and all locations. After that the `get_average` function was called to produce the averages for all these arrays and finally they were printed in an orderly manner.

Section (D) output:



The purpose of this section of the code is to provide an output highlighting the average percentage of diabetes among various age groups (35-49, 60-64 and 65+). Throughout this output we can determine a number of things. This output clearly shows how the percentage of diabetes increases in each set of age groups, which is supported by the fact of how diabetes affects increase as you get older. Because as you get older your body gets weaker and can't produce enough insulin for the body so your blood glucose levels spike therefore leading to heart and kidney problems etc. Not only does this output show the percentage of diabetes overtime, it also shows us the percentage of overweight people as diabetes can be a very serious side effect of being overweight due to high fat intake which then causes insulin resistance. It is also evident

from these results that in each age group, Ontario has the highest percentage of people dealing with diabetes.

The logic for this part was the most important since all the other values for the other questions relied on it to be correct and done properly. The code started by creating a file reader. Then the first while loop: `while (fgets(line, sizeof(line), reader))` goes through the entire file, it gets the values until the end of the file is reached. Then the second while loop: `while (token != NULL)` checks if the tokens (the values taken from the file) are even there. The while loop continues until there are no more tokens. Then the if statement checks if the tokens are from the relevant columns (columns 1, 2, 4,5 and 14). Then it checks if it is column 14, which is the most important column because it is the column that hosts the data. Then it parses the token from a string into a double value that is usable in calculations in the line: `if (sscanf(token, "\"%lf\"", & value) == 1)`. This line also removes the quotation marks to make sure that there are only numerical values being taken in. After that, according to the positions of the values it adds them to the corresponding array. For example, the values of canada from the ages of 35-49 are in the rows 2-15, so using this information it goes through and adds them ignoring any values that are 0. It calls upon the function `add_to_list` which adds them to the list by taking in the parameters of the value, the list and the size of the list. This process for all age groups and locations. Next, the averages are taken by calling the `calculate_average` function. Finally the averages are printed in an organized fashion.

Part 2.

```
------------------------------------------------------------------------
Province with the highest percentage:
Ontario
------------------------------------------------------------------------
Province with the lowest percentage:
British Columbia
------------------------------------------------------------------------
```

For part 2, the aim is for the code and output to highlight those findngs from part 1a). Furthermore, it is important to mention the fact that these findings were calculated with the use of data provided from all year and age groups. This code takes into the account the findings from part 1a) and mention the province with the lowest percentage of diabetics, and highest percentage of diabetics. Furthermore, as evident in the output of this part, the province with the highest percentage is Ontario, and the one with the lowest is British Columbia. As mentioned earlier in the report, it is common to see British Columbia with the lowest calculated percentages throughout the range of outputs in this report. This once again reinforces the fact that British Columbia's provincial government is likely implementing various livelihood changes and regulations that allow for this low percentage. These regulations could possibly include stricter restrictions regarding food retail in the province. This percentage could also possibly be accredited to the fact that British Columbia may have wealthier residents than the rest of Canada,

which would allow them to spend more on their body and health. Moreover, this part of the code's output could be useful to the government as it would indicate which province needs the most medical attention in this field, and which one is not as much of a priority.

For this part, the code was done by simply comparing the values to each other and printing out the largest part. It goes through the averages of the sum arrays one by one until a condition is met and then it prints it out. For example: `if (ontariosumavg < quebecsumavg && ontariosumavg < bcsumavg && ontariosumavg < albertasumavg)` compares if ontario is larger than the other provinces. The same process is done but in reverse to check which is the smallest.

Part 3.

```
------------------------------------------------------------------------
Province(s) higher than the national average:
Ontario
------------------------------------------------------------------------
Province(s) lower than the national average:
Quebec
British Columbia
Alberta
```

In part 3, the aim is to highlight and summarize the percentage findings from the provinces and compare them to the national average. This is for all years and age groups and the output displays the province(s) with percentages higher than the national average, and province(s) with percentages lower than the national average. Moreover, if the output for this part is read, it beccomes evident that Ontario is the only province in all of Canada with diabetic percentages higher than the national average. Additionally, there are only two provinces with percentages lower than the national average, those being British Columbia and Alberta.

This code of this part compares the sum averages of the provinces and prints out the ones that are greater than the national average and less than the national average. For example:

```
if (ontariosumavg > canadasumavg) {
    printf("Ontario \n");
  }
```

checks if ontario is greater than the provincial average and if it is it prints it. This process is done for all the provinces in the same manner.

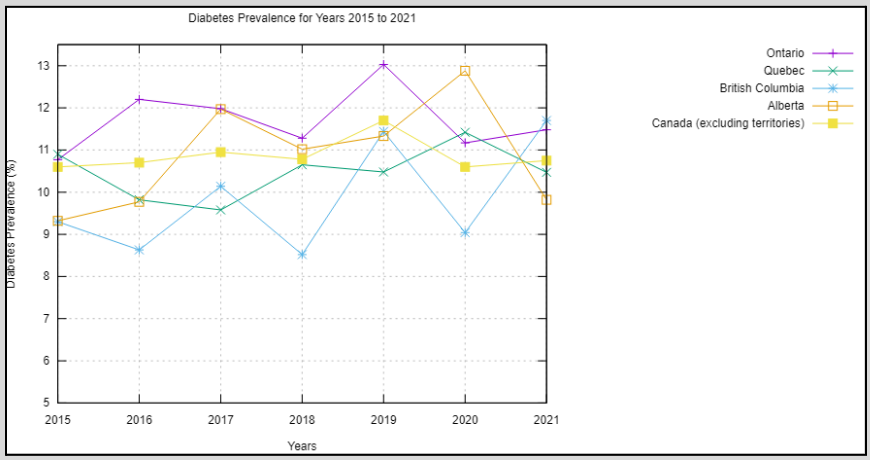
Part 4.

```
------------------------------------------------------------------------
Province(s) and year with the highest percentage
 Ontario 2019 | 13.03%
------------------------------------------------------------------------
Province(s) and year with the lowest percentage
 British Columbia 2018 | 8.52%
```

To begin part 4, the findings from the output from Section C are analyzed, from which the code then outputs the highest and lowest percentage of diabetes. In addition, these percentages are not only for the province but for the year as well. Therefore, judging by these results it can be seen that the highest percentage of diabetes were before 2020 and the COVID-19 pandemic. Moreso, this is important as it showcases how the lockdown affected peoples livelihoods as most restaurants and fast food chains who normally serve food containing high fats were temporarily closed. Throughout all outputs it is once again evident that British Columbia consitently has the lowest calculated percentage of diabetes, thus leading to a conclusion that they promote healthy living. As a result, if the Canadian government wanted to improve health protocol in so the government should look over the type of lifestyle British Columbia provides in order to romote to find solutions for places which have a high diabetes percentage.

The code for this part saves all the sum averages to an array and goes through them and checks which is the largest and prints it.

Part 5.

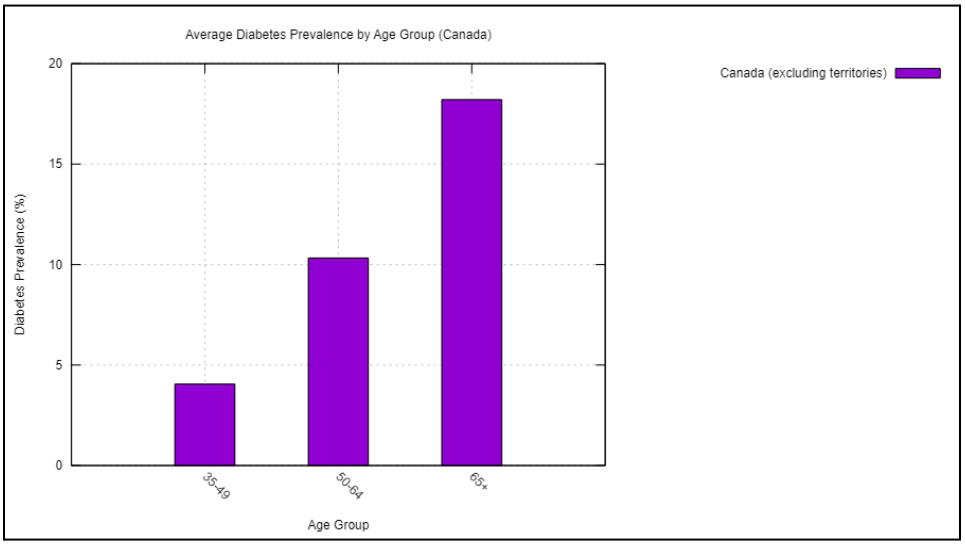


To create the plots and calculate the averages, the group wrote a function in an IDE which was then downloaded and uploaded to the GNU plot software. Furthermore,the 'plot' function was utilized to display the data, and cded the title, legend, and other components according to the instructions to present the information in a well-organized and visually appealing manner.

Moreover, the GNU plot was used to discover the diabetes diagnosis percents from the years 2015 to 2021. To begin, in 2015, Quebec had the highest provincial average surpassing the national diagnoses average whereas British Columbia displayed the lowest provincial average.

The following year, 2016, Ontario had the greatest percentage of provincial diagnoses, while British Columbia had the lowest. Moreover, Ontario also exceeded the national average in 2016. 2017, however, Alberta and Ontario had the highest provincial averages, while Quebec had the lowest. Both Ontario and Alberta had suffered more diagnoses than the average in 2017. Again in 2018, Ontario had the highest rate of diagnoses, while British columbia had the lowest. Both Alberta and Onatrio surpassed the national percentage again in 2018. In 2019, Ontario had the highest percentage of provincial diagnoses while Quebec had the lowest. British Columbia, Alberta and Ontario all exceeded the national average in 2019. 2020, Alberta suffered the highest provincial percentage of diagnoses, while British Columbia had the lowest. Quebec, Ontario and Alberta all surpassed the national average in 2020. Finally in 2021, British Columbia barely surpasses Ontario in highest percentage of provincial diagnoses, while Alberta has the lowest.

Part 6.



To explain this bar graph, it is a combination of several computed values from earlier in the report which are then expected to be presented graphically. Furthermore, the graphical representation needs to be in the format of a bar graph, that represents the percentage of diabetes for three specifed age groups within Canada. Moreover, these values will be similar to those written in part 1.d), although the difference is that this bargraph accounts for the average diabetes prevlance for only three age groups.

For question 6, we calculated for all years combined instead of each year individually like in question 5. We also changed the graph type by changing the code on question 5's plot data to produce a bar graph graph instead of a line plot. The lowest age group, ages 35-49, had the

lowest diabetes prevalence at 4%. Ages 50-64, had a higher diabetes averages from the lower age group but was still much lower than the next age group ranging from years 65+, the 50-64 age group had a diabetes average of 10.5%. Finally, the oldest age range from years 65+, have a diabetes prevalence of 18.5%. The first line which sets the output to 'svg' which provides us with the specifications of the graph, such as width and height of the graph. Command, 'set title', 'set x label', 'set y label', 'set key', provides the user to adjust the x and y-axis labels, the legend and the title to their liking.

The code for this plot sets the size and creates a title. It creates the setup for the bar graph and uses the data form result.txt and uses that to create the graph.

Conclusion

In conclusion, this project was able to provide a great opportunity to showcase our application of C programming skills to analyze real data from Statistics Canada. As seen through the report and analysis, it was evident that diabetes is a significant health concern, especially for those aged 65 and over. Furthermore, the report was able to convey various information through tables and graphs that provided a clear visualization of the data and helped in drawing meaningful conclusions.

In addition, whilst primarily being unproblematic, the project presented some obstacles. Some examples of the obstacles are assessing the mistakes in the early versions of the code and analyzing how to fix them, and finding time in our busy schedules to work together as a group. However, these challenges were all overcome with perseverance, and the team was able to produce accurate results by the end.

Moreover, if given the opportunity to do this project again, the group would consider discussing our scheduling from the start and improve the communication between group members regarding the code as to prevent any misunderstandings. Additionally, we would also refer more to our previously completed CPS labs as each lab covers a different area of code that may be useful to the project. All in all, this project was a valuable learning experience, and the group hopes that this report is of use to anyone interested in understanding the prevalence and affect of diabetes in Canada.

GNU Plot Written Code

```
set terminal svg enhanced size 900,480
set title "Diabetes Prevalence for Years 2015 to 2021"
set xlabel "Years"
set ylabel "Diabetes Prevalence (%)"
set key outside
set yrange [5: 13.5]
set grid ytics mytics
set grid xtics mytics
plot "output.txt" using 1:2 title "Ontario" with linespoints linestyle 1, \
"output.txt" using 1:3 title "Quebec" with linespoints linestyle 2, \
"output.txt" using 1:4 title "British Columbia" with linespoints linestyle 3, \
"output.txt" using 1:5 title "Alberta" with linespoints linestyle 4, \
"output.txt" using 1:6 title "Canada (excluding territories)" with linespoints linestyle 5

set terminal svg enhanced size 900,500
set title 'Average Diabetes Prevalence by Age Group (Canada)'
set xlabel 'Age Group'
set ylabel 'Diabetes Prevalence (%)'
set style data histogram
set style histogram cluster gap 1
set style fill solid border -1
set boxwidth 0.9
set xtics rotate by -45
set yrange[0:20]
set grid ytics mytics
set grid xtics mytics
plot 'result.txt' using 2:xtic(1) title 'Canada (excluding territories)'
```

Written Code

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <float.h>

#define MAX_SIZE 14


//function to add a value to an array
void add_to_list(double num, double * list, int * size) {
  if (num != 0.0) { // check if number is non-blank
    if ( * size < MAX_SIZE) { // check if array list is not full
      list[ * size] = num; // add number to array list
      ( * size) ++; // increase size of array list
    } else {
```

```
      //printf("Array list is full\n"); // error message for full array list
    }
  }
}

//function to calculate the average
double calculate_average(double arr[], int size) {
  double sum = 0.0;
  for (int i = 0; i < size; i++) {
    sum += arr[i];
  }
  return sum / size;
}


int main(void) {

//initialize the sizes for the arrays
  int size1 = 0;
  int size2 = 0;
  int size3 = 0;
  int size4 = 0;
  int size5 = 0;
  int size6 = 0;
  int size7 = 0;
  int size8 = 0;
  int size9 = 0;
  int size10 = 0;
  int size11 = 0;
  int size12 = 0;
  int size13 = 0;
  int size14 = 0;
  int size15 = 0;

  int sizetest = 0;
  int sizea = 0;
  int sizeb = 0;
  int sizec = 0;
  int sized = 0;
  int sizee = 0;
  int sizef = 0;

  int sizeg = 0;
  int sizeh = 0;
  int sizei = 0;
  int sizej = 0;
  int sizek = 0;
  int sizel = 0;
  int sizem = 0;

  int sizen = 0;
  int sizeo = 0;
  int sizep = 0;
  int sizeq = 0;
  int sizer = 0;
  int sizes = 0;
  int sizet = 0;

  int sizeu = 0;
  int sizev = 0;
  int sizew = 0;
  int sizex = 0;
```

```
  int sizey = 0;
  int sizez = 0;
  int sizeaa = 0;

  int sizebb = 0;
  int sizecc = 0;
  int sizedd = 0;
  int sizeee = 0;
  int sizeff = 0;
  int sizegg = 0;
  int sizehh = 0;

//counter to determine which row you're on
  int counter = 1;

//initialize the arrays that are going to host the data
  double canada35[MAX_SIZE] = {
    0
  };
  double canada50[MAX_SIZE] = {
    0
  };
  double canada65[MAX_SIZE] = {
    0
  };

  double quebec35[MAX_SIZE] = {
    0
  };
  double quebec50[MAX_SIZE] = {
    0
  };
  double quebec65[MAX_SIZE] = {
    0
  };

  double ontario35[MAX_SIZE] = {
    0
  };
  double ontario50[MAX_SIZE] = {
    0
  };
  double ontario65[MAX_SIZE] = {
    0
  };

  double alberta35[MAX_SIZE] = {
    0
  };
  double alberta50[MAX_SIZE] = {
    0
  };
  double alberta65[MAX_SIZE] = {
    0
  };

  double bc35[MAX_SIZE] = {
    0
  };
  double bc50[MAX_SIZE] = {
    0
  };
```

```
double bc65[MAX_SIZE] = {
  0
};

double ontario2015[MAX_SIZE] = {
  0
};
double ontario2016[MAX_SIZE] = {
  0
};
double ontario2017[MAX_SIZE] = {
  0
};
double ontario2018[MAX_SIZE] = {
  0
};
double ontario2019[MAX_SIZE] = {
  0
};
double ontario2020[MAX_SIZE] = {
  0
};
double ontario2021[MAX_SIZE] = {
  0
};

double quebec2015[MAX_SIZE] = {
  0
};
double quebec2016[MAX_SIZE] = {
  0
};
double quebec2017[MAX_SIZE] = {
  0
};
double quebec2018[MAX_SIZE] = {
  0
};
double quebec2019[MAX_SIZE] = {
  0
};
double quebec2020[MAX_SIZE] = {
  0
};
double quebec2021[MAX_SIZE] = {
  0
};

double bc2015[MAX_SIZE] = {
  0
};
double bc2016[MAX_SIZE] = {
  0
};
double bc2017[MAX_SIZE] = {
  0
};
double bc2018[MAX_SIZE] = {
  0
};
double bc2019[MAX_SIZE] = {
  0
```

```
};
double bc2020[MAX_SIZE] = {
  0
};
double bc2021[MAX_SIZE] = {
  0
};

double alberta2015[MAX_SIZE] = {
  0
};
double alberta2016[MAX_SIZE] = {
  0
};
double alberta2017[MAX_SIZE] = {
  0
};
double alberta2018[MAX_SIZE] = {
  0
};
double alberta2019[MAX_SIZE] = {
  0
};
double alberta2020[MAX_SIZE] = {
  0
};
double alberta2021[MAX_SIZE] = {
  0
};

double canada2015[MAX_SIZE] = {
  0
};
double canada2016[MAX_SIZE] = {
  0
};
double canada2017[MAX_SIZE] = {
  0
};
double canada2018[MAX_SIZE] = {
  0
};
double canada2019[MAX_SIZE] = {
  0
};
double canada2020[MAX_SIZE] = {
  0
};
double canada2021[MAX_SIZE] = {
  0
};

double canadasum[MAX_SIZE * 3] = {
  0
};
double quebecsum[MAX_SIZE * 3] = {
  0
};
double ontariosum[MAX_SIZE * 3] = {
  0
};
double albertasum[MAX_SIZE * 3] = {
```

```
    0
  };
  double bcsum[MAX_SIZE * 3] = {
    0
  };

  int quebecsize = 0;


//read the file
  FILE * reader = fopen("statscan_diabetes.csv", "r");
  if (reader == NULL) {
    //if it does not open the file then it returns an error
    perror("Unable to open the file.");
    exit(1);
  }

  char line[200];
  //this is a while loop that takes in the values one by one from the file
  while (fgets(line, sizeof(line), reader)) {
    char * token;
    token = strtok(line, ",");
    int column_count = 1;

    double value;
    //the code only proceeds if the token has a value associated with it
    while (token != NULL) {
      //checks for colomns 1, 2, 4, 5, 14
      if (column_count == 1 || column_count == 2 || column_count == 4 || column_count == 5 || column_count ==
14) {
        //colomn 14 which hold the actual values have the processes associated with it
        if (column_count == 14) {
          //this pareses the string in the token into a double while removing the quotations
          if (sscanf(token, "\"%lf\"", & value) == 1) {

            //this adds the values to canada and for the age group 35-49 (PART D)
            if (counter >= 1 && counter <= 15) {
              add_to_list(value, canada35, & size1);

            //this adds the values to canada and for the age group 35-49 (PART D)
              if (counter == 2 || counter == 9) {
                add_to_list(value, canada2015, & sizetest);
              }
              //this adds the values to canada and for the year 2016 (PART C)
              if (counter == 3 || counter == 10) {
                add_to_list(value, canada2016, & sizea);
              }
              //this adds the values to canada and for the year 2017 (PART C)
              if (counter == 4 || counter == 11) {
                add_to_list(value, canada2017, & sizeb);
              }
              //this adds the values to canada and for the year 2018 (PART C)
              if (counter == 5 || counter == 12) {
                add_to_list(value, canada2018, & sizec);
              }
              //this adds the values to canada and for the year 2019 (PART C)
              if (counter == 6 || counter == 13) {
                add_to_list(value, canada2019, & sized);
              }
              //this adds the values to canada and for the year 2020 (PART C)
              if (counter == 7 || counter == 14) {
                add_to_list(value, canada2020, & sizee);
```

```
            }
            //this adds the values to canada and for the year 2021 (PART C)
            if (counter == 8 || counter == 15) {
              add_to_list(value, canada2021, & sizef);
            }
          }
// THE SAME PROCESS DONE ABOVE IS DONE FOR ALL THE ARRAYS FOR PART C AND PART D
          if (counter >= 16 && counter <= 29) {
            add_to_list(value, canada50, & size2);

            if (counter == 16 || counter == 23) {
              add_to_list(value, canada2015, & sizetest);
            }
            if (counter == 17 || counter == 24) {
              add_to_list(value, canada2016, & sizea);
            }
            if (counter == 18 || counter == 25) {
              add_to_list(value, canada2017, & sizeb);
            }
            if (counter == 19 || counter == 26) {
              add_to_list(value, canada2018, & sizec);
            }
            if (counter == 20 || counter == 27) {
              add_to_list(value, canada2019, & sized);
            }
            if (counter == 21 || counter == 28) {
              add_to_list(value, canada2020, & sizee);
            }
            if (counter == 22 || counter == 29) {
              add_to_list(value, canada2021, & sizef);
            }

          }

          if (counter >= 30 && counter <= 43) {
            add_to_list(value, canada65, & size3);

            if (counter == 30 || counter == 37) {
              add_to_list(value, canada2015, & sizetest);
            }
            if (counter == 31 || counter == 38) {
              add_to_list(value, canada2016, & sizea);
            }
            if (counter == 32 || counter == 39) {
              add_to_list(value, canada2017, & sizeb);
            }
            if (counter == 33 || counter == 40) {
              add_to_list(value, canada2018, & sizec);
            }
            if (counter == 34 || counter == 41) {
              add_to_list(value, canada2019, & sized);
            }
            if (counter == 35 || counter == 42) {
              add_to_list(value, canada2020, & sizee);
            }
            if (counter == 36 || counter == 43) {
              add_to_list(value, canada2021, & sizef);
            }
          }

          if (counter >= 44 && counter <= 57) {
            add_to_list(value, quebec35, & size4);
```

```
  if (counter == 44 || counter == 51) {
    add_to_list(value, quebec2015, & sizeg);
  }
  if (counter == 45 || counter == 52) {
    add_to_list(value, quebec2016, & sizeh);
  }
  if (counter == 46 || counter == 53) {
    add_to_list(value, quebec2017, & sizei);
  }
  if (counter == 47 || counter == 54) {
    add_to_list(value, quebec2018, & sizej);
  }
  if (counter == 48 || counter == 55) {
    add_to_list(value, quebec2019, & sizek);
  }
  if (counter == 49 || counter == 56) {
    add_to_list(value, quebec2020, & sizel);
  }
  if (counter == 50 || counter == 57) {
    add_to_list(value, quebec2021, & sizem);
  }
}
if (counter >= 58 && counter <= 71) {
  add_to_list(value, quebec50, & size5);

  if (counter == 58 || counter == 65) {
    add_to_list(value, quebec2015, & sizeg);
  }
  if (counter == 59 || counter == 66) {
    add_to_list(value, quebec2016, & sizeh);
  }
  if (counter == 60 || counter == 67) {
    add_to_list(value, quebec2017, & sizei);
  }
  if (counter == 61 || counter == 68) {
    add_to_list(value, quebec2018, & sizej);
  }
  if (counter == 62 || counter == 69) {
    add_to_list(value, quebec2019, & sizek);
  }
  if (counter == 63 || counter == 70) {
    add_to_list(value, quebec2020, & sizel);
  }
  if (counter == 64 || counter == 71) {
    add_to_list(value, quebec2021, & sizem);
  }
}
if (counter >= 72 && counter <= 85) {
  add_to_list(value, quebec65, & size6);

  if (counter == 72 || counter == 79) {
    add_to_list(value, quebec2015, & sizeg);
  }
  if (counter == 73 || counter == 80) {
    add_to_list(value, quebec2016, & sizeh);
  }
  if (counter == 74 || counter == 81) {
    add_to_list(value, quebec2017, & sizei);
  }
  if (counter == 75 || counter == 82) {
    add_to_list(value, quebec2018, & sizej);
```

```
  }
  if (counter == 76 || counter == 83) {
    add_to_list(value, quebec2019, & sizek);
  }
  if (counter == 77 || counter == 84) {
    add_to_list(value, quebec2020, & sizel);
  }
  if (counter == 78 || counter == 85) {
    add_to_list(value, quebec2021, & sizem);
  }
}

if (counter >= 86 && counter <= 99) {
  add_to_list(value, ontario35, & size7);
  if (counter == 86 || counter == 93) {
    add_to_list(value, ontario2015, & sizen);
  }
  if (counter == 87 || counter == 94) {
    add_to_list(value, ontario2016, & sizeo);
  }
  if (counter == 88 || counter == 95) {
    add_to_list(value, ontario2017, & sizep);
  }
  if (counter == 89 || counter == 96) {
    add_to_list(value, ontario2018, & sizeq);
  }
  if (counter == 90 || counter == 97) {
    add_to_list(value, ontario2019, & sizer);
  }
  if (counter == 91 || counter == 98) {
    add_to_list(value, ontario2020, & sizes);
  }
  if (counter == 92 || counter == 99) {
    add_to_list(value, ontario2021, & sizet);
  }
}
if (counter >= 100 && counter <= 113) {
  add_to_list(value, ontario50, & size8);
  if (counter == 100 || counter == 107) {
    add_to_list(value, ontario2015, & sizen);
  }
  if (counter == 101 || counter == 108) {
    add_to_list(value, ontario2016, & sizeo);
  }
  if (counter == 102 || counter == 109) {
    add_to_list(value, ontario2017, & sizep);
  }
  if (counter == 103 || counter == 110) {
    add_to_list(value, ontario2018, & sizeq);
  }
  if (counter == 104 || counter == 111) {
    add_to_list(value, ontario2019, & sizer);
  }
  if (counter == 105 || counter == 112) {
    add_to_list(value, ontario2020, & sizes);
  }
  if (counter == 106 || counter == 113) {
    add_to_list(value, ontario2021, & sizet);
  }
}
if (counter >= 114 && counter <= 127) {
  add_to_list(value, ontario65, & size9);
```

```
  if (counter == 114 || counter == 121) {
    add_to_list(value, ontario2015, & sizen);
  }
  if (counter == 115 || counter == 122) {
    add_to_list(value, ontario2016, & sizeo);
  }
  if (counter == 116 || counter == 123) {
    add_to_list(value, ontario2017, & sizep);
  }
  if (counter == 117 || counter == 124) {
    add_to_list(value, ontario2018, & sizeq);
  }
  if (counter == 118 || counter == 125) {
    add_to_list(value, ontario2019, & sizer);
  }
  if (counter == 119 || counter == 126) {
    add_to_list(value, ontario2020, & sizes);
  }
  if (counter == 120 || counter == 127) {
    add_to_list(value, ontario2021, & sizet);
  }
}

if (counter >= 128 && counter <= 141) {
  add_to_list(value, alberta35, & size10);
  if (counter == 128 || counter == 135) {
    add_to_list(value, alberta2015, & sizeu);
  }
  if (counter == 129 || counter == 136) {
    add_to_list(value, alberta2016, & sizev);
  }
  if (counter == 130 || counter == 137) {
    add_to_list(value, alberta2017, & sizew);
  }
  if (counter == 131 || counter == 138) {
    add_to_list(value, alberta2018, & sizex);
  }
  if (counter == 132 || counter == 139) {
    add_to_list(value, alberta2019, & sizey);
  }
  if (counter == 133 || counter == 140) {
    add_to_list(value, alberta2020, & sizez);
  }
  if (counter == 134 || counter == 141) {
    add_to_list(value, alberta2021, & sizeaa);
  }
}
if (counter >= 142 && counter <= 155) {
  add_to_list(value, alberta50, & size11);
  if (counter == 142 || counter == 149) {
    add_to_list(value, alberta2015, & sizeu);
  }
  if (counter == 143 || counter == 150) {
    add_to_list(value, alberta2016, & sizev);
  }
  if (counter == 144 || counter == 151) {
    add_to_list(value, alberta2017, & sizew);
  }
  if (counter == 145 || counter == 152) {
    add_to_list(value, alberta2018, & sizex);
  }
  if (counter == 146 || counter == 153) {
```

```
    add_to_list(value, alberta2019, & sizey);
  }
  if (counter == 147 || counter == 154) {
    add_to_list(value, alberta2020, & sizez);
  }
  if (counter == 148 || counter == 155) {
    add_to_list(value, alberta2021, & sizeaa);
  }
}
if (counter >= 156 && counter <= 169) {
  add_to_list(value, alberta65, & size12);
  if (counter == 156 || counter == 163) {
    add_to_list(value, alberta2015, & sizeu);
  }
  if (counter == 157 || counter == 164) {
    add_to_list(value, alberta2016, & sizev);
  }
  if (counter == 158 || counter == 165) {
    add_to_list(value, alberta2017, & sizew);
  }
  if (counter == 159 || counter == 166) {
    add_to_list(value, alberta2018, & sizex);
  }
  if (counter == 160 || counter == 167) {
    add_to_list(value, alberta2019, & sizey);
  }
  if (counter == 161 || counter == 168) {
    add_to_list(value, alberta2020, & sizez);
  }
  if (counter == 162 || counter == 169) {
    add_to_list(value, alberta2021, & sizeaa);
  }
}

if (counter >= 170 && counter <= 183) {
  add_to_list(value, bc35, & size13);
  if (counter == 170 || counter == 177) {
    add_to_list(value, bc2015, & sizebb);
  }
  if (counter == 171 || counter == 178) {
    add_to_list(value, bc2016, & sizecc);
  }
  if (counter == 172 || counter == 179) {
    add_to_list(value, bc2017, & sizedd);
  }
  if (counter == 173 || counter == 180) {
    add_to_list(value, bc2018, & sizeee);
  }
  if (counter == 174 || counter == 181) {
    add_to_list(value, bc2019, & sizeff);
  }
  if (counter == 175 || counter == 182) {
    add_to_list(value, bc2020, & sizegg);
  }
  if (counter == 176 || counter == 183) {
    add_to_list(value, bc2021, & sizehh);
  }

}
if (counter >= 184 && counter <= 197) {
  add_to_list(value, bc50, & size14);
  if (counter == 184 || counter == 191) {
```

```
              add_to_list(value, bc2015, & sizebb);
            }
            if (counter == 185 || counter == 191) {
              add_to_list(value, bc2016, & sizecc);
            }
            if (counter == 186 || counter == 193) {
              add_to_list(value, bc2017, & sizedd);
            }
            if (counter == 187 || counter == 194) {
              add_to_list(value, bc2018, & sizeee);
            }
            if (counter == 188 || counter == 195) {
              add_to_list(value, bc2019, & sizeff);
            }
            if (counter == 189 || counter == 196) {
              add_to_list(value, bc2020, & sizegg);
            }
            if (counter == 190 || counter == 197) {
              add_to_list(value, bc2021, & sizehh);
            }
          }
          if (counter >= 198 && counter <= 211) {
            add_to_list(value, bc65, & size15);
            if (counter == 198 || counter == 205) {
              add_to_list(value, bc2015, & sizebb);
            }
            if (counter == 199 || counter == 206) {
              add_to_list(value, bc2016, & sizecc);
            }
            if (counter == 200 || counter == 207) {
              add_to_list(value, bc2017, & sizedd);
            }
            if (counter == 201 || counter == 208) {
              add_to_list(value, bc2018, & sizeee);
            }
            if (counter == 202 || counter == 209) {
              add_to_list(value, bc2019, & sizeff);
            }
            if (counter == 203 || counter == 210) {
              add_to_list(value, bc2020, & sizegg);
            }
            if (counter == 204 || counter == 210) {
              add_to_list(value, bc2021, & sizehh);
            }
          }

        }
      }
    }
    //this is used to get the next token that is seperated by a comma
    token = strtok(NULL, ",");
  //increases the colomn counter to move on to the next colomn
    column_count++;
  }
  //increases the row counter to move on to the next row
  counter++;
}

//closes the file
  fclose(reader); // always close the file when done with it

//FOR PARTS A
```

```
//adds the arrays associated with canada to one array to help get the total average
  int index = 0;
  for (int i = 0; i < MAX_SIZE; i++) {
    canadasum[index++] = canada35[i];
    canadasum[index++] = canada50[i];
    canadasum[index++] = canada65[i];
  }
//DOES THE SAME AS THE PREVIOUS CODE BUT FOR THE REST OF THE PROVINCES (PART B)
  int index2 = 0;

  for (int i = 0; i < MAX_SIZE; i++) {
    if (quebec35[i] != 0) {
      quebecsum[index2++] = quebec35[i];
    }
    if (quebec50[i] != 0) {
      quebecsum[index2++] = quebec50[i];
    }
    if (quebec65[i] != 0) {
      quebecsum[index2++] = quebec65[i];
    }

  }
  index2 = 0;
  for (int i = 0; i < MAX_SIZE; i++) {

    ontariosum[index2++] = ontario35[i];
    ontariosum[index2++] = ontario50[i];
    ontariosum[index2++] = ontario65[i];
  }

  index2 = 0;
  for (int i = 0; i < MAX_SIZE; i++) {

    if (bc35[i] != 0) {
      bcsum[index2++] = bc35[i];

    }
    if (bc50[i] != 0) {
      bcsum[index2++] = bc50[i];

    }
    if (bc65[i] != 0) {
      bcsum[index2++] = bc65[i];

    }
  }

  index2 = 0;
  for (int i = 0; i < MAX_SIZE; i++) {
    if (alberta35[i] != 0) {
      albertasum[index2++] = alberta35[i];

    }
    if (alberta50[i] != 0) {
      albertasum[index2++] = alberta50[i];
    }
    if (alberta65[i] != 0) {
      albertasum[index2++] = alberta65[i];
    }

  }
//goes to the function calculate average, which calculates the average for all the arrays
```

```
double canadasumavg = calculate_average(canadasum, MAX_SIZE * 3);
double quebecsumavg = calculate_average(quebecsum, size4 + size5 + size6);
double ontariosumavg = calculate_average(ontariosum, MAX_SIZE * 3);
double bcsumavg = calculate_average(bcsum, size13 + size14 + size15);
double albertasumavg = calculate_average(albertasum, size10 + size11 + size12);


double canada35avg = calculate_average(canada35, size1);
double canada50avg = calculate_average(canada50, size2);
double canada65avg = calculate_average(canada65, size3);

double quebec35avg = calculate_average(quebec35, size4);
double quebec50avg = calculate_average(quebec50, size5);
double quebec65avg = calculate_average(quebec65, size6);

double ontario35avg = calculate_average(ontario35, size7);
double ontario50avg = calculate_average(ontario50, size8);
double ontario65avg = calculate_average(ontario65, size9);

double alberta35avg = calculate_average(alberta35, size10);
double alberta50avg = calculate_average(alberta50, size11);
double alberta65avg = calculate_average(alberta65, size12);

double bc35avg = calculate_average(bc35, size13);
double bc50avg = calculate_average(bc50, size14);
double bc65avg = calculate_average(bc65, size15);

double canada2015avg = calculate_average(canada2015, sizetest);
double canada2016avg = calculate_average(canada2016, sizea);
double canada2017avg = calculate_average(canada2017, sizeb);
double canada2018avg = calculate_average(canada2018, sizec);
double canada2019avg = calculate_average(canada2019, sized);
double canada2020avg = calculate_average(canada2020, sizee);
double canada2021avg = calculate_average(canada2021, sizef);

double quebec2015avg = calculate_average(quebec2015, sizeg);
double quebec2016avg = calculate_average(quebec2016, sizeh);
double quebec2017avg = calculate_average(quebec2017, sizei);
double quebec2018avg = calculate_average(quebec2018, sizej);
double quebec2019avg = calculate_average(quebec2019, sizek);
double quebec2020avg = calculate_average(quebec2020, sizel);
double quebec2021avg = calculate_average(quebec2021, sizem);

double ontario2015avg = calculate_average(ontario2015, sizen);
double ontario2016avg = calculate_average(ontario2016, sizeo);
double ontario2017avg = calculate_average(ontario2017, sizep);
double ontario2018avg = calculate_average(ontario2018, sizeq);
double ontario2019avg = calculate_average(ontario2019, sizer);
double ontario2020avg = calculate_average(ontario2020, sizes);
double ontario2021avg = calculate_average(ontario2021, sizet);

double alberta2015avg = calculate_average(alberta2015, sizeu);
double alberta2016avg = calculate_average(alberta2016, sizev);
double alberta2017avg = calculate_average(alberta2017, sizew);
double alberta2018avg = calculate_average(alberta2018, sizex);
double alberta2019avg = calculate_average(alberta2019, sizey);
double alberta2020avg = calculate_average(alberta2020, sizez);
double alberta2021avg = calculate_average(alberta2021, sizeaa);

double bc2015avg = calculate_average(bc2015, sizebb);
double bc2016avg = calculate_average(bc2016, sizecc);
double bc2017avg = calculate_average(bc2017, sizedd);
```

```
  double bc2018avg = calculate_average(bc2018, sizeee);
  double bc2019avg = calculate_average(bc2019, sizeff);
  double bc2020avg = calculate_average(bc2020, sizegg);
  double bc2021avg = calculate_average(bc2021, sizehh);


  //prints all the headers and the averages in a proper format

  printf("------------------------------------------------------------------------ \n");
//QUESTION 1 PART A & B
  printf("Total Provincial and National percent averages: \n");
  printf("Ontario: %.2f%% \n", ontariosumavg);
  printf("Quebec: %.2f%% \n", quebecsumavg);
  printf("British Columbia %.2f%% \n", bcsumavg);
  printf("Alberta: %.2f%% \n", albertasumavg);
  printf("National: %.2f%% \n", canadasumavg);

  printf("------------------------------------------------------------------------ \n");
//QUESTION 1 PART C
  printf("The Provincial and National averages for 2015: \n");
  printf("Ontario: %.2f%% \n", ontario2015avg);
  printf("Quebec: %.2f%% \n", quebec2015avg);
  printf("British Columbia: %.2f%% \n", bc2015avg);
  printf("Alberta: %.2f%% \n", alberta2015avg);
  printf("National: %.2f%% \n", canada2015avg);

  printf("------------------------------------------------------------------------ \n");
  printf("The Provincial and National averages for 2016: \n");
  printf("Ontario: %.2f%% \n", ontario2016avg);
  printf("Quebec: %.2f%% \n", quebec2016avg);
  printf("British Columbia: %.2f%% \n", bc2016avg);
  printf("Alberta: %.2f%% \n", alberta2016avg);
  printf("National: %.2f%% \n", canada2016avg);

  printf("------------------------------------------------------------------------ \n");
  printf("The Provincial and National averages for 2017: \n");
  printf("Ontario: %.2f%% \n", ontario2017avg);
  printf("Quebec: %.2f%% \n", quebec2017avg);
  printf("British Columbia: %.2f%% \n", bc2017avg);
  printf("Alberta Average for 2017: %.2f%% \n", alberta2017avg);
  printf("National Average for 2017: %.2f%% \n", canada2017avg);

  printf("------------------------------------------------------------------------ \n");
  printf("The Provincial and National averages for 2018: \n");
  printf("Ontario : %.2f%% \n", ontario2018avg);
  printf("Quebec: %.2f%% \n", quebec2018avg);
  printf("British Columbia: %.2f%% \n", bc2018avg);
  printf("Alberta: %.2f%% \n", alberta2018avg);
  printf("National: %.2f%% \n", canada2018avg);

  printf("------------------------------------------------------------------------ \n");
  printf("The Provincial and National averages for 2019: \n");

  printf("Ontario: %.2f%% \n", ontario2019avg);
  printf("Quebec: %.2f%% \n", quebec2019avg);
  printf("British Columbia: %.2f%% \n", bc2019avg);
  printf("Alberta: %.2f%% \n", alberta2019avg);
  printf("National: %.2f%% \n", canada2019avg);

  printf("------------------------------------------------------------------------ \n");
  printf("The Provincial and National averages for 2020: \n");
  printf("Ontario: %.2f%% \n", ontario2020avg);
```

```
  printf("Quebec: %.2f%% \n", quebec2020avg);
  printf("British Columbia: %.2f%% \n", bc2020avg);
  printf("Alberta: %.2f%% \n", alberta2020avg);
  printf("National: %.2f%% \n", canada2020avg);

  printf("------------------------------------------------------------------------ \n");
  printf("The Provincial and National averages for 2021: \n");
  printf("Ontario: %.2f%% \n", ontario2021avg);
  printf("Quebec: %.2f%% \n", quebec2021avg);
  printf("British Columbia: %.2f%% \n", bc2021avg);
  printf("Alberta: %.2f%% \n", alberta2021avg);
  printf("National: %.2f%% \n", canada2021avg);

  printf("------------------------------------------------------------------------ \n");
  //Question 1 PART D
  printf("The Provincial and Nation percent averages for 35-49: \n");
  printf(" Ontario: %.2f%% \n", ontario35avg);
  printf(" Quebec: %.2f%% \n", quebec35avg);
  printf(" British Columbia: %.2f%% \n", bc35avg);
  printf(" Alberta: %.2f%% \n", alberta35avg);
  printf(" Canada: %.2f%% \n", canada35avg);

  printf("------------------------------------------------------------------------ \n");
  printf("The Provincial and Nation percent averages for 50-64: \n");

  printf(" Ontario: %.2f%% \n", ontario50avg);
  printf(" Quebec: %.2f%% \n", quebec50avg);
  printf(" British Columbia: %.2f%% \n", bc50avg);
  printf(" Alberta: %.2f%% \n", alberta50avg);
  printf(" Canada: %.2f%% \n", canada50avg);

  printf("------------------------------------------------------------------------ \n");
  printf("The Provincial and Nation percent averages for 65+: \n");

  printf(" Ontario: %.2f%% \n", ontario65avg);
  printf(" Quebec: %.2f%% \n", quebec65avg);
  printf(" British Columbia: %.2f%% \n", bc65avg);
  printf(" Alberta: %.2f%% \n", alberta65avg);
  printf(" Canada: %.2f%% \n", canada65avg);
  printf("\n");
  printf("------------------------------------------------------------------------ \n");
//Question 2
  printf("Province with the highest percentage: \n");

//Goes through each array and checks which array is the largest
  if (ontariosumavg > quebecsumavg && ontariosumavg > bcsumavg && ontariosumavg > albertasumavg) {
    printf("Ontario \n");
  } else if (quebecsumavg > ontariosumavg && quebecsumavg > bcsumavg && quebecsumavg > albertasumavg) {
    printf("Quebec \n");
  } else if (bcsumavg > ontariosumavg && bcsumavg > quebecsumavg && bcsumavg > albertasumavg) {
    printf("British Columbia \n");
  } else {
    printf("Alberta \n");
  }

  printf("------------------------------------------------------------------------ \n");
  printf("Province with the lowest percentage: \n");
//Goes through each array and checks which array is the smallest
  if (ontariosumavg < quebecsumavg && ontariosumavg < bcsumavg && ontariosumavg < albertasumavg) {
    printf("Ontario \n");
  } else if (quebecsumavg < ontariosumavg && quebecsumavg < bcsumavg && quebecsumavg < albertasumavg) {
    printf("Quebec \n");
```

```
  } else if (bcsumavg < ontariosumavg && bcsumavg < quebecsumavg && bcsumavg < albertasumavg) {
    printf("British Columbia \n");
  } else {
    printf("Alberta \n");
  }

  printf("------------------------------------------------------------------------ \n");
   //Question 3 compares all the arrays to the national average and prints all the provinces that are higher
than the national average
  printf("Province(s) higher than the national average: \n");

  if (ontariosumavg > canadasumavg) {
    printf("Ontario \n");
  }
  if (quebecsumavg > canadasumavg) {
    printf("Quebec \n");
  }
  if (bcsumavg > canadasumavg) {
    printf("British Columbia \n");
  }
  if (albertasumavg > canadasumavg) {
    printf("Alberta \n");
  }

  printf("------------------------------------------------------------------------ \n");
     //Question 3 compares all the arrays to the national average and prints all the provinces that are lower
than the national average
  printf("Province(s) lower than the national average: \n");
  //
  if (ontariosumavg < canadasumavg) {
    printf("Ontario \n");
  }
  if (quebecsumavg < canadasumavg) {
    printf("Quebec \n");
  }
  if (bcsumavg < canadasumavg) {
    printf("British Columbia \n");
  }
  if (albertasumavg < canadasumavg) {
    printf("Alberta \n");
  }

  printf("------------------------------------------------------------------------ \n");
  printf("Province(s) and year with the highest percentage\n");

  //QUESTION 4


  double max_value = -DBL_MAX;
  double min_value = DBL_MAX;

//saves all the averages to an array
  double province_avgs[28] = {
    quebec2015avg,
    quebec2016avg,
    quebec2017avg,
    quebec2018avg,
    quebec2019avg,
    quebec2020avg,
    quebec2021avg,
    ontario2015avg,
    ontario2016avg,
```

```
    ontario2017avg,
    ontario2018avg,
    ontario2019avg,
    ontario2020avg,
    ontario2021avg,
    alberta2015avg,
    alberta2016avg,
    alberta2017avg,
    alberta2018avg,
    alberta2019avg,
    alberta2020avg,
    alberta2021avg,
    bc2015avg,
    bc2016avg,
    bc2017avg,
    bc2018avg,
    bc2019avg,
    bc2020avg,
    bc2021avg
  };
//creates names to print in association with the array averages
  const char * province_years[28] = {
    "Quebec 2015",
    "Quebec 2016",
    "Quebec 2017",
    "Quebec 2018",
    "Quebec 2019",
    "Quebec 2020",
    "Quebec 2021",
    "Ontario 2015",
    "Ontario 2016",
    "Ontario 2017",
    "Ontario 2018",
    "Ontario 2019",
    "Ontario 2020",
    "Ontario 2021",
    "Alberta 2015",
    "Alberta 2016",
    "Alberta 2017",
    "Alberta 2018",
    "Alberta 2019",
    "Alberta 2020",
    "Alberta 2021",
    "British Columbia 2015",
    "British Columbia 2016",
    "British Columbia 2017",
    "British Columbia 2018",
    "British Columbia 2019",
    "British Columbia 2020",
    "British Columbia 2021"
  };

  int max_index = -1;
  int min_index = -1;

  //goes through the averages to determine which one is the highest
  for (int i = 0; i < 28; i++) {
    if (province_avgs[i] > max_value) {
      max_value = province_avgs[i];
      max_index = i;
    }
  //goes through the averages to determine which one is the lowest
```

```
    if (province_avgs[i] < min_value) {
      min_value = province_avgs[i];
      min_index = i;
    }
  }

//prints out the highest average
  if (max_index != -1) {
    printf(" %s | %.2f%%\n", province_years[max_index], max_value);
  }

  //prints the lowest average
  printf("------------------------------------------------------------------------ \n");
  printf("Province(s) and year with the lowest percentage\n");
  if (min_index != -1) {
    printf(" %s | %.2f%% \n", province_years[min_index], min_value);
  }

FILE *outputFile;
  outputFile = fopen("output.txt", "w");
  FILE *resultsFile;
  resultsFile = fopen("result.txt", "w");

  fprintf(outputFile,"2015        %.2lf       %.2lf     %.2lf               %.2lf         %.2lf\n",
ontario2015avg, quebec2015avg, bc2015avg, alberta2015avg, canada2015avg);
  fprintf(outputFile,"2016        %.2lf       %.2lf      %.2lf               %.2lf         %.2lf\n",
ontario2016avg, quebec2016avg, bc2016avg, alberta2016avg, canada2016avg);
  fprintf(outputFile,"2017        %.2lf       %.2lf      %.2lf              %.2lf        %.2lf\n",
ontario2017avg, quebec2017avg, bc2017avg, alberta2017avg, canada2017avg);
  fprintf(outputFile,"2018        %.2lf       %.2lf     %.2lf               %.2lf        %.2lf\n",
ontario2018avg, quebec2018avg, bc2018avg, alberta2018avg, canada2018avg);
  fprintf(outputFile,"2019        %.2lf       %.2lf     %.2lf              %.2lf        %.2lf\n",
ontario2019avg, quebec2019avg, bc2019avg, alberta2019avg, canada2019avg);
  fprintf(outputFile,"2020        %.2lf       %.2lf     %.2lf               %.2lf        %.2lf\n",
ontario2020avg, quebec2020avg, bc2020avg, alberta2020avg, canada2020avg);
  fprintf(outputFile,"2021        %.2lf       %.2lf     %.2lf              %.2lf         %.2lf\n",
ontario2021avg, quebec2021avg, bc2021avg, alberta2021avg, canada2021avg);

    fclose(outputFile);

fprintf(resultsFile, "#AgeGroup  Percentage\n");

  // Prints the values in the file
  fprintf(resultsFile, "35-49     %.2lf\n", canada35avg);
  fprintf(resultsFile, "50-64     %.2lf\n", canada50avg);
  fprintf(resultsFile, "65+       %.2lf\n", canada65avg);

  // Closes the file

  fclose(resultsFile);



  return 0;
}
```

References

CPS188 : Term Project : Winter 2023. Google Docs. (n.d.). Retrieved April 1, 2023, from https://docs.google.com/document/d/1LWcFIEftVkho7zBlU2T81ot3SZ_SqdS7IehDcNyBQyo/edit