Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Computer Programming Fundamentals
Course Number:	CPS 188
Semester/Year	W2023

Instructor:	Dr. Denis Hamelin

Assignment/Lab Number:	1
Assignment/Lab Title:	CPS 188: Term Project Winter 2023
Group Number	56

Submission Date	April 2, 2023
Due Date:	April 2, 2023

Student LAST Name	**Student FIRST Name**	**Student Number**	**Section**	**Signature***
Dawson	Zaevian	501169957	5	Z. Dawson
Wei	Allen	501162087	5	Allen Wei
Zmak	Petar	501188283	4	Petar Zmak
Ali	Mohammad	501181080	2	[illegible]

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

Introduction

Nearly 8.9% of Canada's population fall under the Diabetic category, along with a 3.3% increase every year. A report made by Statistics Canada shows the prevalence of diabetes in Canada's four most populated regions (Ontario, Quebec, British Columbia, and Alberta) between 2015 and 2021. Based on this, we will use the C programming language and GNUplot to analyze the data. Our data analysis for our report will include calculating the average percentage diagnosed with diabetes for all age categories at the provincial and national levels. Furthermore, there are average percentages of diabetic diagnosis for each age group along with yearly average percentages at the national and provincial levels. From these calculations, we will be able to identify the provincial averages that are both above and below the national average, accompanied by the largest/lowest yearly and provincial averages. Moreover, we will create a line graph that will display the provincial and national averages along with a bar graph that will display the national average percentage of the age groups in Canada. In summary, this report provides a detailed analysis of the diabetic percentage in the four most populated provinces in Canada.

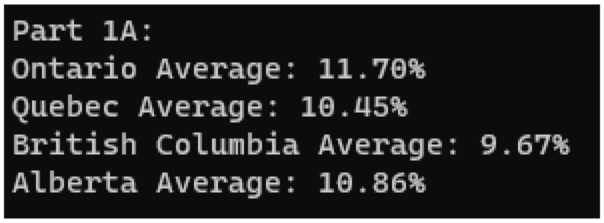
Calculations:

Question 1:

a)

Part A of Question 1 asked for the average provincial percentages to be calculated. This was done using a loop and an if-statement. Since all the percentages were stored in a single array, the loop would cycle through each value in the array called "value". Inside the for-loop are five if-statements, one for every location. These if-statements and else if-statements would check to see the location each value belongs to. This would ensure that all Ontario values get stored in the array for the Ontario average, and so on. Since zero values were to be omitted from each average, an if-statement was required to ensure that the values were not zero. Therefore inside each if-statement that was reading the location, there was an if-statement that checked if the percentage value was zero. If the value was zero, then the value would not be added to the average. To keep track of the number of percentages that were being added, an array that counted the number of times a percentage value was not equal to zero, was added. This array simply would start at zero and increase by one each time a new percentage value was added. The reason this counter was an array was because there are five locations to keep track of. This decreased the number of variables needed.

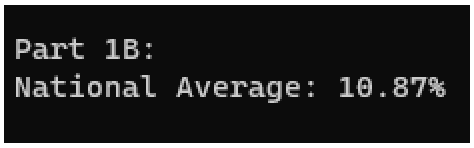
To calculate the average, a for-loop was used to cycle through each final sum and divide it by its respective counter value. The values were then printed using a for-loop that cycled through each of the newly calculated averages.



```
Part 1A:
Ontario Average: 11.70%
Quebec Average: 10.45%
British Columbia Average: 9.67%
Alberta Average: 10.86%
```

b)

Part B of this Question asked for the national average of diabetes. This would include only the values that fall under the location “Canada(excluding territories)”. The national average was calculated at the same time and in the same way as the Provincial averages. There was an address in the array that contained all the provincial averages, that was for the national average. The for-loop would go through all of the percentage values, and if an average was at the location “Canada(excluding territories)” and the value was not equal to zero, then that percentage would be added to the sum. Inside these, if statements were also a counter that would count the number of times a percentage value got summed. This final sum was then divided by the counter, which resulted in the final average, which was printed. The procedure to calculate the national average was done in the exact same way as the provincial averages, which were done in Part A.



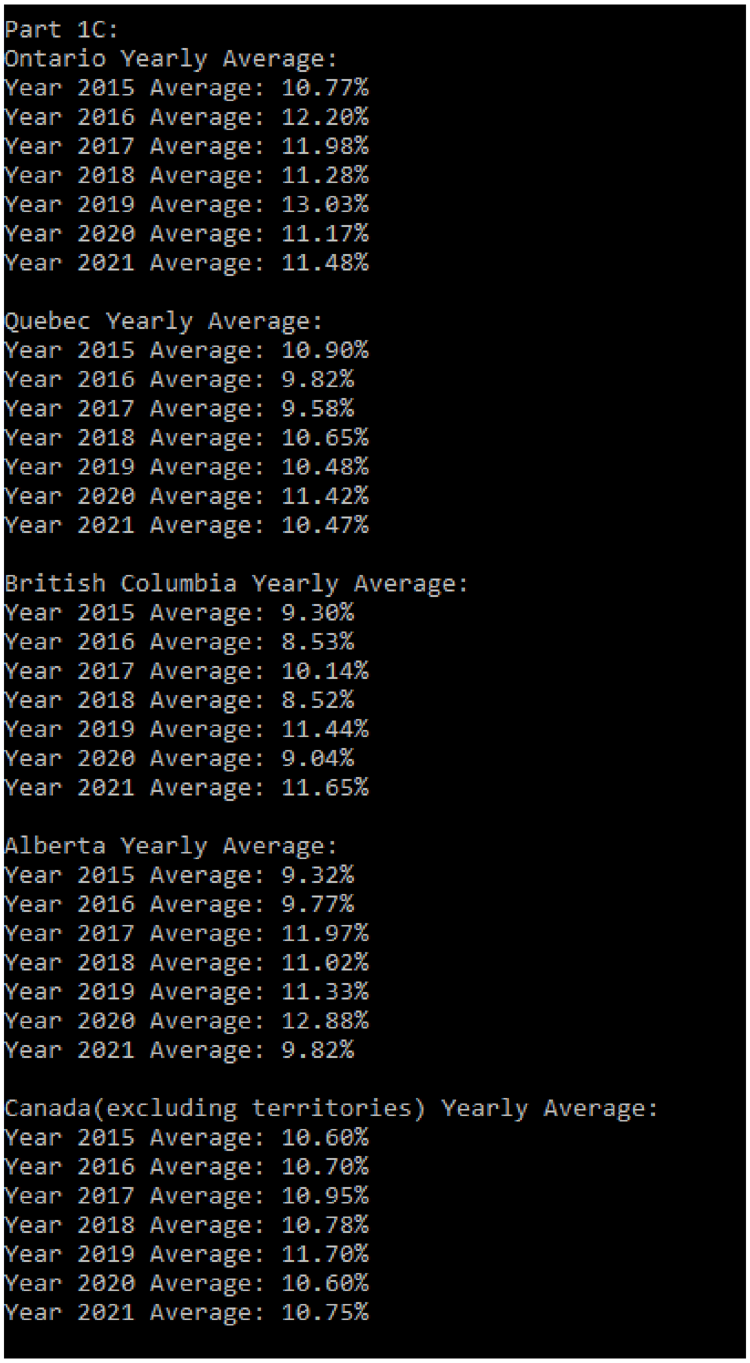
```
Part 1B:
National Average: 10.87%
```

c)

The method used to calculate the average percentage of the population who were diagnosed with diabetes for each year from 2015 to 2021 in Ontario, in Quebec, in Alberta, in British Columbia or in Canada as a whole (excluding the territories) was to make five separate arrays to calculate the percentage averages for every year from 2015 to 2021 for each province and Canada. The five arrays each could store seven integer values. The indices in the array would store the averages in the respective year in ascending order, for example, the average of the values in 2015 for Ontario would be stored in index 0 of the array, the average of the values in 2016 for Ontario would be stored in index 1 of the array, 2017 values for Ontario in index 2, and so on. Five additional counter arrays that stored seven integer values were created to keep track of the number of percentage values that were added to the yearly provincial average arrays. The counter arrays are necessary to calculate the yearly averages later on in the program. The average percentages arrays and the counter arrays were all initialized to contain 0s in every cell.

A for-loop is used to loop through every potential data row. The for-loop contains five if-statements with seven if-statements nested inside. The outer five if-statements checked to see which location the data belongs to, the seven nested if-statements inside check to see which year the data belongs to, and within the seven nested if-statements is one final if-statement that ensures the percentage value for diabetes exists for that specific location and year. Once all conditions are met, the percentage value for diabetes is added to its respective array and array index and the respective counter array and counter array index is increased by one. For example, ont_yearlyavgcount[1] = ont_yearlyavgcount[1] + 1.

After all the values have been added to their respective arrays and the for-loop ends, another for-loop is used to convert the summed values into averages. The for-loop loops through every index in the array and calculates the provincial yearly averages by dividing the sum of the yearly percentage values for diabetes by their respective yearly average counts, for example, ont_yearlyavg[i] = ont_yearlyavg[i] / ont_yearlyavgcount[i].

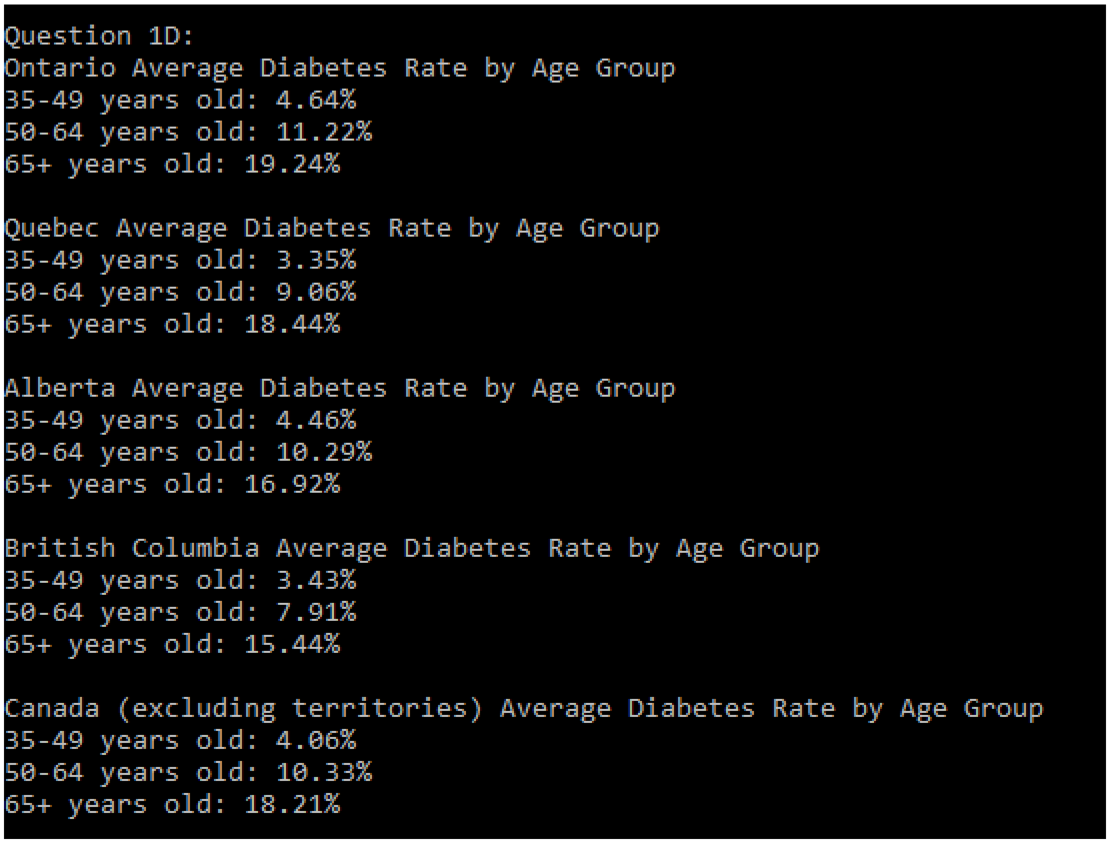


```
Part 1C:
Ontario Yearly Average:
Year 2015 Average: 10.77%
Year 2016 Average: 12.20%
Year 2017 Average: 11.98%
Year 2018 Average: 11.28%
Year 2019 Average: 13.03%
Year 2020 Average: 11.17%
Year 2021 Average: 11.48%

Quebec Yearly Average:
Year 2015 Average: 10.90%
Year 2016 Average: 9.82%
Year 2017 Average: 9.58%
Year 2018 Average: 10.65%
Year 2019 Average: 10.48%
Year 2020 Average: 11.42%
Year 2021 Average: 10.47%

British Columbia Yearly Average:
Year 2015 Average: 9.30%
Year 2016 Average: 8.53%
Year 2017 Average: 10.14%
Year 2018 Average: 8.52%
Year 2019 Average: 11.44%
Year 2020 Average: 9.04%
Year 2021 Average: 11.65%

Alberta Yearly Average:
Year 2015 Average: 9.32%
Year 2016 Average: 9.77%
Year 2017 Average: 11.97%
Year 2018 Average: 11.02%
Year 2019 Average: 11.33%
Year 2020 Average: 12.88%
Year 2021 Average: 9.82%

Canada(excluding territories) Yearly Average:
Year 2015 Average: 10.60%
Year 2016 Average: 10.70%
Year 2017 Average: 10.95%
Year 2018 Average: 10.78%
Year 2019 Average: 11.70%
Year 2020 Average: 10.60%
Year 2021 Average: 10.75%
```

d)

To calculate the average percentage of the population diagnosed with diabetes, for each age group (35-49, 50-64 and 65+), for each of the provinces/Canada, lab members created five arrays that had a size of 3. For example, the array double proavg_on[3] = {0, 0, 0}, stores the average percentage of the population in Ontario diagnosed with diabetes for the age groups 35-49 (index 0), 50-64 (index 1) and 65+ (index 2). The values in the array have an initialized value of 0, but as the calculations were done, those values were updated. An outer for loop was used to check every line in the cleaned text file. Within the for loop, five if statements were used, and within each of those five if statements, three if statements were nested inside. The outer if statements checked that the corresponding location data matched. The next three nested if statements checked that the age group data being read, matched one of the age groups (35-49, 50-64 or 65+). After the age group data was checked, a nested if statement was used to check if the percentage value was 0. If the percentage value was 0, then that value was not added to the values array.

For example, the outer if statement would check if the province was Ontario, and if so, then three nested if statements would check for the age group, in this case, that could be ages 35-49. Finally, another nested if statement would add the given diabetes percentage value to the array (proavg_on) that stores the diabetes percentage values for Ontario if the value was not 0. Also, the array that keeps track of how many values are added to the (proavg_on) array, is increased by 1 each time a new value was added to the (proavg_on) array. Finally, to compute the averages for each province and Canada, the sum of the values added to a provinces/Canada's values array for each age group was divided by the number of values that were added to it. For example, proavg_on[i] = proavg_on[i]/avgcount_on[i] was computed for each of the three age categories using a for loop that looped three times. Five separate for loops were used to compute the averages for Ontario, Quebec, Alberta, British Columbia and Canada.

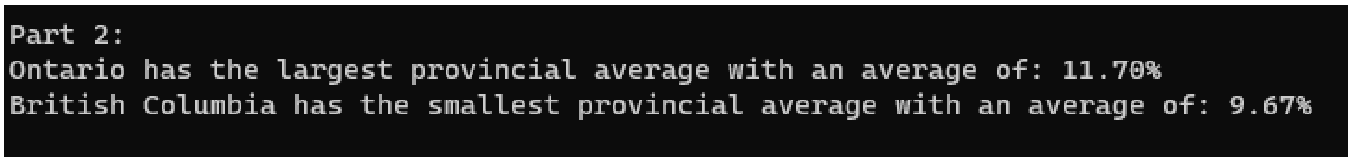


```
Question 1D:
Ontario Average Diabetes Rate by Age Group
35-49 years old: 4.64%
50-64 years old: 11.22%
65+ years old: 19.24%

Quebec Average Diabetes Rate by Age Group
35-49 years old: 3.35%
50-64 years old: 9.06%
65+ years old: 18.44%

Alberta Average Diabetes Rate by Age Group
35-49 years old: 4.46%
50-64 years old: 10.29%
65+ years old: 16.92%

British Columbia Average Diabetes Rate by Age Group
35-49 years old: 3.43%
50-64 years old: 7.91%
65+ years old: 15.44%

Canada (excluding territories) Average Diabetes Rate by Age Group
35-49 years old: 4.06%
50-64 years old: 10.33%
65+ years old: 18.21%
```

Question 2:

Question 2 asked to find which province had the highest percentage average, and which province had the lowest percentage average. To find which value was the largest, the variable "provmax" was set to the first value in the array "proavg". This array contained all of the provincial averages that were calculated in Part 1A. A for-loop was then used to loop through the values of "proavg". Inside the loop was an if-statement that checked if the value of "proavg" at the address specified by the for-loop was larger than the current value of "provmax". If the value was larger, then "provmax" would be set to this new value. If the value was smaller, then the if-statement would be not entered and "provmax" would not be changed. Additionally, depending on which address of "proavg" was being checked, a string containing the name of the province would be copied to the array "provname". The for-loop would continue looping this until all the provincial averages were checked. The final value of "provmax" was then printed along with its corresponding province.

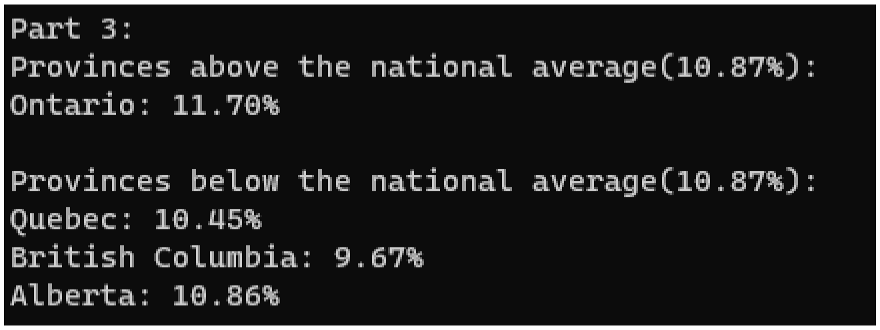
The exact same approach was used to find the smallest provincial average. The only difference in the code being the if-statement. To find the smallest provincial average, the if-statement would check to see if the value of "proavg" at the address specified at the for-loop was smaller than the current value of the variable "provmin". If this was true, then the value of “provmin” would be updated to this new smaller value. Additionally, depending on which address of “proavg” was being checked, a string containing the name of the province would be copied to the array “provname”. The value of “provmin” was then printed along with its corresponding province.



```
Part 2:
Ontario has the largest provincial average with an average of: 11.70%
British Columbia has the smallest provincial average with an average of: 9.67%
```

Question 3:

Question 3 asked to find which provinces had averages that were above the national average, and which provinces were below the national average. To determine this, a for-loop was used to cycle through the calculated average percentages in the array “provavg”. Instead of looping through the entire array, the for-loop was terminated at the address of “3” because the address of “4” contained the national average. In this for-loop were two if-statements. One would check if the address of “proavg” specified by the loop was greater than the value of “proavg” at the address “4”. The second if-statement would then check to see if the value was less than national average which was located in “proavg” at address “4”. Depending on if the value was higher or lower, the provincial average would be sorted into the array “avg_above” and “avg_below” respectively. Additionally, depending on which address of “proavg” was being looked at, their corresponding provincial names would also be sorted into arrays called “pro_above” and “pro_below”. This would allow for one printf-statement to print all of the Provinces above the national average, and one printf-statement to print all of the provinces below the national average.

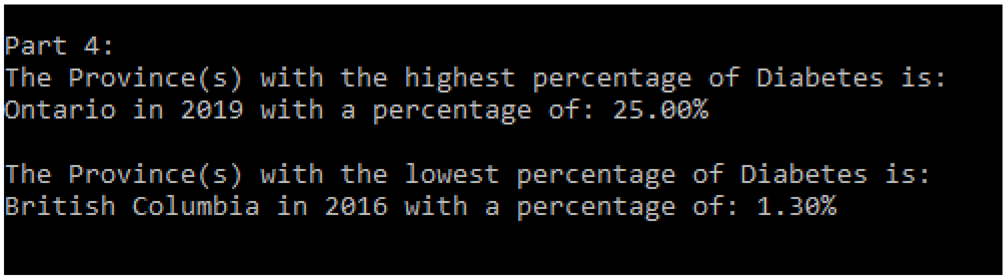


```
Part 3:
Provinces above the national average(10.87%):
Ontario: 11.70%

Provinces below the national average(10.87%):
Quebec: 10.45%
British Columbia: 9.67%
Alberta: 10.86%
```

Question 4:

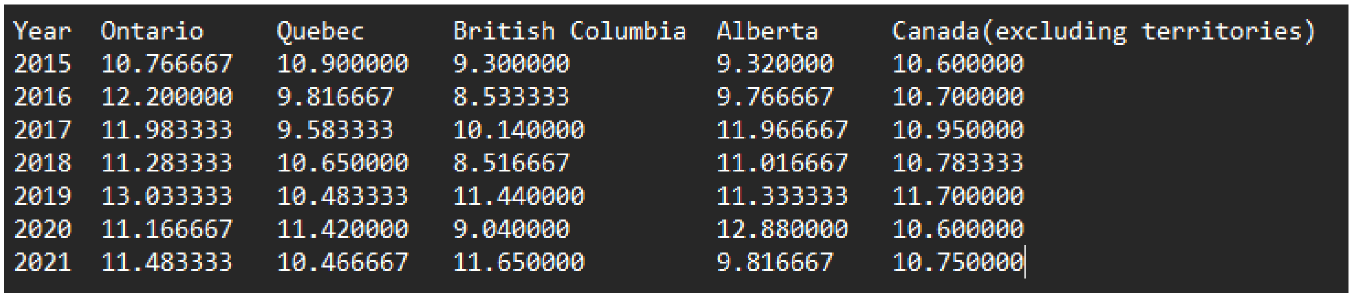
Question 4 asked to find which province and year has the highest percentage of diabetes and which province and year has the lowest percentage of diabetes. Since the array “value” contained all of the percentages, a for-loop was used to cycle through all of the percentages. To find which percentage was the highest, the percentage value specified by the loop would be compared to the variable “avg_max”. If the value was larger, then “avg_max” would be updated to the new value. Inside this if-statement, the year of the highest percentage would be copied from the array “year” to the variable “year_max” and the name of the province would be copied from the array “location” to “provname_max”. Both of these would be printed along with the percentage.

To find the lowest percentage, the variable “avg_min” was set to the value of “avg_max”. Then, similar to finding the largest percentage, the current percentage specified by a for-loop would be compared to the variable “avg_min”. This if-statement would compare to make sure that the percentage was smaller and if the percentage was not zero. Zero was used to replace all missing data. If the value was smaller and not equal to zero, then the value of “avg_min” would be updated to the value of “value” at the address specified by the loop. Inside the if-statement, the year of the lowest percentage would then be copied from the array “year” to the variable “year_min” and the name of the province would be copied from the array “location” to “provname_min”. Both of these would be printed along with the percentage.


```
Part 4:
The Province(s) with the highest percentage of Diabetes is:
Ontario in 2019 with a percentage of: 25.00%

The Province(s) with the lowest percentage of Diabetes is:
British Columbia in 2016 with a percentage of: 1.30%
```

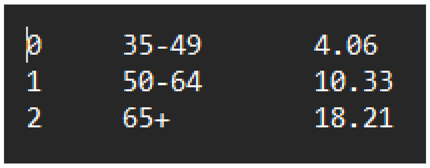
Question 5:

Question 5 asked to create a graph that contained all of the yearly provincial averages. To do this, a file called “Question5.txt” was created. A fprintf-statement was used to print out the headers into the file. Following the headers, a loop and a fprintf-statement was used to print all the yearly averages for all of the provinces from their respective arrays. After everything was printed, the file “graph1” was closed. This resulted in a file with the following output:


```
Year  Ontario     Quebec      British Columbia  Alberta     Canada(excluding territories)
2015  10.766667   10.900000   9.300000          9.320000    10.600000
2016  12.200000   9.816667    8.533333          9.766667    10.700000
2017  11.983333   9.583333    10.140000         11.966667   10.950000
2018  11.283333   10.650000   8.516667          11.016667   10.783333
2019  13.033333   10.483333   11.440000         11.333333   11.700000
2020  11.166667   11.420000   9.040000          12.880000   10.600000
2021  11.483333   10.466667   11.650000         9.816667    10.750000
```

Question 6:

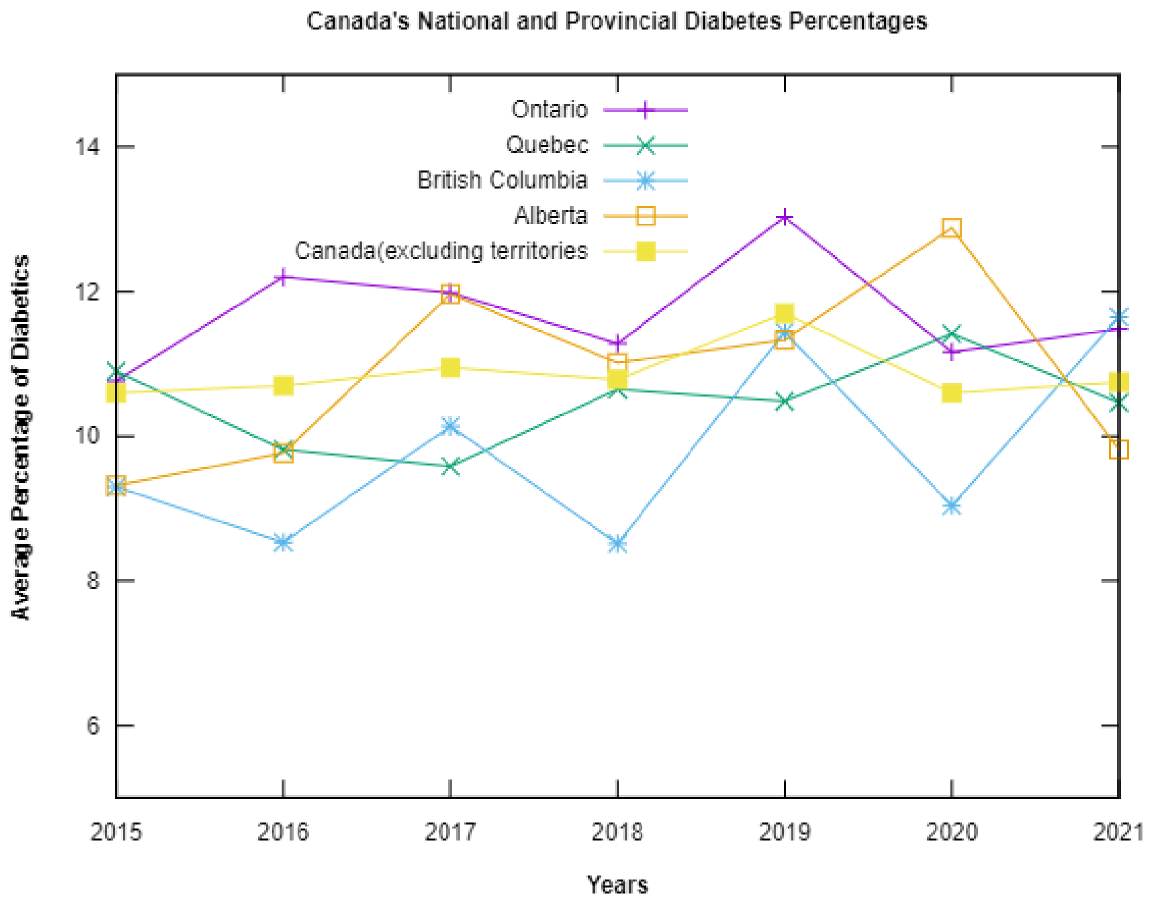
In order to generate a bar graph that shows the average percentage of diabetes among the three age groups for Canada (excluding territories), a text file had to be generated. In order to make this text file, a new file named Question6_bar_graph.txt was created and a pointer variable (graphs2) was also created. A 2D array that stored the three different age groups (35-49, 50-64 and 65+) was used, and an array that stored the numbers 0, 1 and 2 was also used. The numbers 0, 1 and 2 represented the index of each bar on the bar graph. The first bar had an index of 0, the second bar had an index of 1, and the third bar had an index of 2. In addition, the average percentage of the population diagnosed with diabetes in Canada (excluding territories), for each of the three age groups was accessed using the proavg_cad array that stored these values. A for loop was used to print three rows of information, into the new text file. Each row contained the index number of the bar, the age group and the average percentage of the population diagnosed with diabetes value for Canada. Lastly, the new file pointer (graphs2) was closed in the main C file. The resultant text file is shown below.



```
0      35-49        4.06
1      50-64        10.33
2      65+          18.21
```

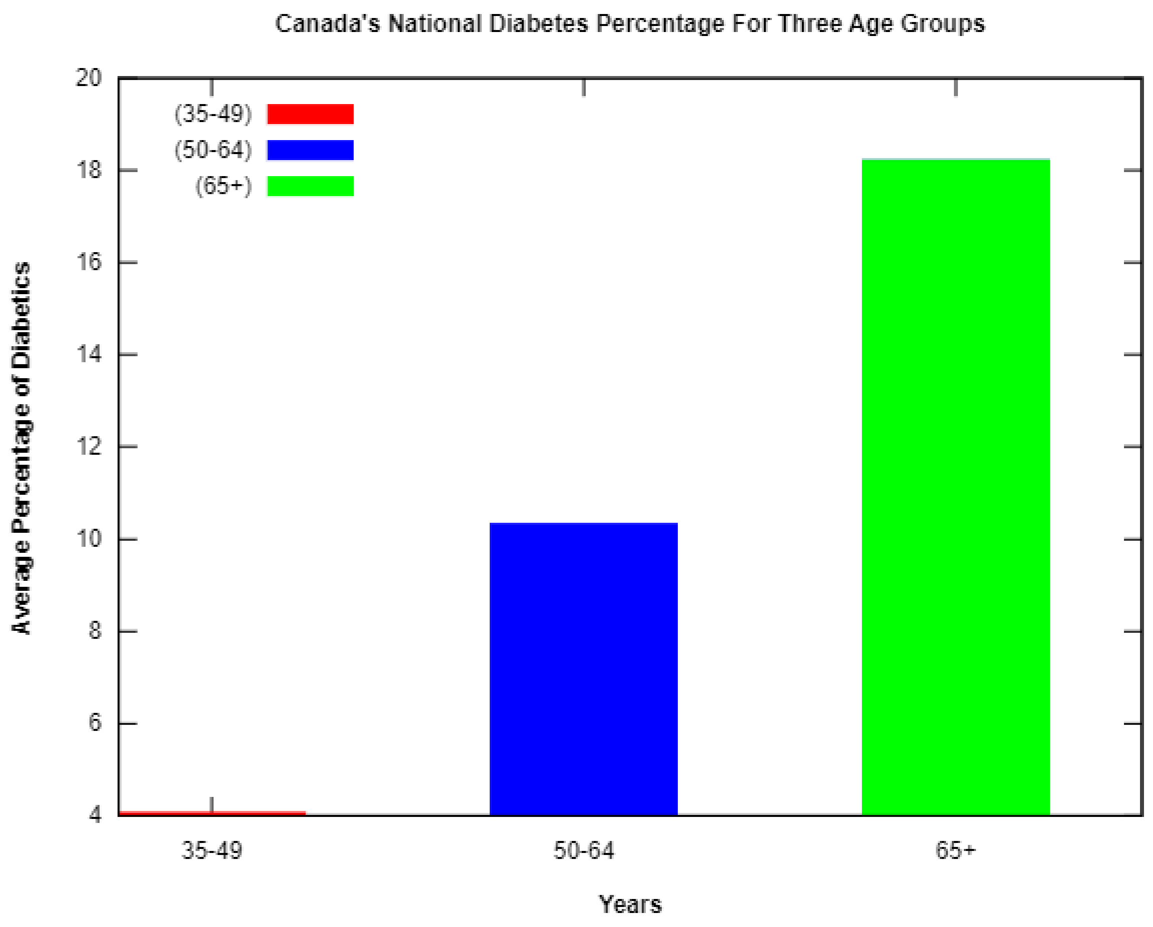
GNUPlot Graphs:

Question 5:



In this assignment, we used the data provided by Stats Canada and put it through our program. This generated a file with the yearly average percent value of diabetics in the four most populated provinces and Canada as a whole. The file was then imported into GNUplot, where the titles and axis labels are coded, with the ranges being manually set to make the graph more comprehensive. The data is then plotted into a line graph. Five lines are plotted, each being distinctly colour coded to represent the four provinces, with one representing Canada as a whole. Finally, a legend is created with the correct labels to represent the provinces and Canada as a whole. The resultant line graph shows a clear visual representation of the data.

Question 6:



In this assignment, we used the data provided by Stats Canada and put it through our program. This generated a file with the average percent value of diabetics in three distinct age groups. The file is then imported to GNUplot, where the titles and axis labels are coded, with the ranges on both axes set automatically. The data is then used to plot the bar graph with colour coding. Finally, a legend is added to the graph, with the correct labels. The resultant bar graph shows a clear visual representation of the data.

Conclusion

In conclusion, the program was written coherently and concisely, allowing us to correctly import the Stats Canada data file. The file was imported into the program using a series of loops and if statements, which created a file inside the program after ridding it of unnecessary punctuation and spaces so it could readily be accessed throughout the program. Using our program, we can see that out of the four most populated provinces, Ontario has the greatest percentage of diabetics at 11.70%, and in contrast, British Columbia has the lowest percentage at

9.67%. But if we look at Canada as a whole, the average percentage of diabetics is 10.87%. Next, we calculated the yearly average percent of diabetics in the four provinces and Canada as a whole, and even with slight fluctuations, it is clearly evident that the percentage is on the rise, indicating that as the years progress, there will be more people reported with diabetes. We also looked at the ages of people that have been diagnosed with diabetes, and it can be observed that the older a person is, the more likely it is for them to be diabetic. We can see in our graph that approximately 18% of people at the age of 65 and above have diabetes, whereas approximately 4% of people at the age of 35-49 have diabetes.

This project provided a unique opportunity for all group members to experience what working with a large set of data would be like. Additionally, this program pushed our limits in terms of organization and cohesiveness. Since multiple people were working on one project, the group needed to be well coordinated in order to make sure that all parts of the program would work together. The program is also over 800 lines long, the largest program any of us have coded before, this required the group to be well organized so that all individual parts of the code would be organized in the file. If this group were to do the project again, we would try to reduce the length of our program by exploring other shorter options to solve the problems posed to us. This would improve organization and efficiency. In the code, there are large quantities of repetitive code, if more time was spent on finding alternative methods to perform the same function, but with reduced lines of code, perhaps the program would be more neat and optimized.

<u>Code</u>

C Program

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

void
csvtotxt(){
	char clean[3];
	FILE *in, *out;

	in = fopen("statscan_diabetes.csv", "r");
	out = fopen("clean.txt", "w");


	while (!feof(in)){

		fscanf(in, "%c", &clean[0]);

		//Replaces all empty values with a ~ character
		if (clean[0] == '"' && clean[1] == '"'){
			clean[1] = '~';
			fprintf(out, "%c", clean[1]);
		}

		// Gets rid of all " and unecessary spaces
		if(clean[0] == '"'){
		}
		else if(clean[0] == ' '){
		}
		else if(clean[0] == ','){
			clean[0] = ' ';
			fprintf(out, "%c", clean[0]);
		}
		else
			fprintf(out, "%c", clean[0]);

		// Necessary for the first if statement
		clean[1] = clean[0];

	}
	fclose(in);
	fclose(out);
}

int
main (void){

	char yearchar[300][5];
char headers[18][17];
char location[300][30];
char age[300][20];
char sex[300][10];
char valuechar[300][5];
char empty[300][50];

int i, k, g;
int year[300];
int ls_years[7];
double value[300];

//Part 1A and 1B
double proavg[5] = {0, 0, 0, 0, 0};
int avgcount[5] = {0, 0, 0, 0, 0};

//Part 1C
double ont_yearlyavg[7] = {0, 0, 0, 0, 0, 0, 0};
double que_yearlyavg[7] = {0, 0, 0, 0, 0, 0, 0};
double bc_yearlyavg[7] = {0, 0, 0, 0, 0, 0, 0};
double alb_yearlyavg[7] = {0, 0, 0, 0, 0, 0, 0};
double cad_yearlyavg[7] = {0, 0, 0, 0, 0, 0, 0};
int ont_yearlyavgcount[7] = {0, 0, 0, 0, 0, 0, 0};
int que_yearlyavgcount[7] = {0, 0, 0, 0, 0, 0, 0};
int bc_yearlyavgcount[7] = {0, 0, 0, 0, 0, 0, 0};
int alb_yearlyavgcount[7] = {0, 0, 0, 0, 0, 0, 0};
int cad_yearlyavgcount[7] = {0, 0, 0, 0, 0, 0, 0};

//Part 1D
double proavg_on[3] = {0, 0, 0};
double proavg_que[3] = {0, 0, 0};
double proavg_ab[3] = {0, 0, 0};
double proavg_bc[3] = {0, 0, 0};
double proavg_cad[3] = {0, 0, 0};

int avgcount_on[3] = {0, 0, 0};
int avgcount_que[3] = {0, 0, 0};
int avgcount_ab[3] = {0, 0, 0};
int avgcount_bc[3] = {0, 0, 0};
int avgcount_cad[3] = {0, 0, 0};

//Part 2
double provmax, provmin;
char provname[5][40];
char provmaxmin[2][30];

//Part 3
double avg_above[4];
double avg_below[4];
char pro_above[4][25];
char pro_below[4][25];

//Part 4
```

```
        double avg_max;
        int year_max;
        char provname_max[5][40];

        double avg_min;
        int year_min;
        char provname_min[11][40];



        FILE *cleaned;

        //Calls function that converts csv file to txt file
        csvtotxt();

        cleaned = fopen("clean.txt", "r");

        // Reads headers row
        for (i = 0; i < 18; i++){
                fscanf(cleaned, "%s", headers[i]);
                }

        // Reads all the necessary values and places them into their respective arrays
        k = 0;
        while (!feof(cleaned)){

                for (i = 0; i < 18; i++){
                        if(i == 0){
                                fscanf(cleaned, "%s", yearchar[k]);
                                year[k] = atof(yearchar[k]);
                        }
                        else if(i == 1)
                                fscanf(cleaned, "%s", location[k]);
                        else if(i == 3)
                                fscanf(cleaned, "%s", age[k]);
                        else if(i == 4)
                                fscanf(cleaned, "%s", sex[k]);
                        else if(i == 13){
                                fscanf(cleaned, "%s", valuechar[k]);
                                value[k] = atof(valuechar[k]);
                        }
                        else
                                fscanf(cleaned, "%s", empty[k]);
                                }
                        k++;
        }
        fclose(cleaned);

//Fills Array that contains all of the years
        g = 0;
        for (i = 0; i < k; i++){
                if(g < 7){
                        if(year[i] != ls_years[g])
                        ls_years[g] = year[i];
                g++;
            }
        }

//Part 1A and Calculation for 1B

        //checks for the location, and then calculates the sum of all the percentages at that
location
        //Note: if the percentage is zero, the program skips over it
        for (i = 0; i < k; i++){
                if(strcmp(location[i], "Ontario") == 0){
                        if(value[i] != 0){
                                proavg[0] = proavg[0] + value[i];
                                avgcount[0] = avgcount[0] + 1;
                                strcpy(provname[0], "Ontario");
                        }
                }
                else if(strcmp(location[i], "Quebec") == 0){
                        if(value[i] != 0){
                                proavg[1] = proavg[1] + value[i];
                                avgcount[1] = avgcount[1] + 1;
                                strcpy(provname[1], "Quebec");
                        }
                }
                else if(strcmp(location[i], "BritishColumbia") == 0){
                        if(value[i] != 0){
                                proavg[2] = proavg[2] + value[i];
                                avgcount[2] = avgcount[2] + 1;
                                strcpy(provname[2], "British Columbia");
                        }
                }
                else if(strcmp(location[i], "Alberta") == 0){
                        if(value[i] != 0){
                                proavg[3] = proavg[3] + value[i];
                                avgcount[3] = avgcount[3] + 1;
                                strcpy(provname[3], "Alberta");
                        }
                }
                //Part 1B Calculation
                else if(strcmp(location[i], "Canada(excludingterritories)") == 0){
                        if(value[i] != 0){
                                proavg[4] = proavg[4] + value[i];
                                avgcount[4] = avgcount[4] + 1;
                                strcpy(provname[4], "Canada(excluding territories)");
                        }
                }
        }

        // Calculates the average of each sum
        for (i = 0; i < 5; i++){
                proavg[i] = proavg[i]/avgcount[i];
        }
        //Prints output
        printf("Part 1A:\n");
        for (i = 0; i < 4; i++)
                printf("%s Average: %.2lf%%\n", provname[i], proavg[i]);


//Part 1B
        //Code for Part 1B is same as Part 1A
        //Prints National Average
        printf("\n\nPart 1B:\n");
        printf("National Average: %.2lf%%\n", proavg[4]);


// Part 1C
        // loops through every row
        for (int i = 0; i < 300; i++)
        {
                // checks if the location is Ontario
                if(strcmp(location[i], "Ontario") == 0){
                        /* depending on what year it is, if the percentage value exists,
                                adds the percentage value to its respective index in the
ont_yearlyavg array,
                                as well as, add +1 to its respective index in the
ont_yearlyavgcount
                                array which counts the number of values added to the ont_yearlyavg
array. */
                        if (year[i] == 2015)
                        {
                                if (value[i] != 0)
                                {
                                        ont_yearlyavgcount[0] = ont_yearlyavgcount[0] + 1;
                                        ont_yearlyavg[0] = ont_yearlyavg[0] + value[i];
                                }
                        }
                        else if (year[i] == 2016)
                        {
                                if (value[i] != 0)
                                {
                                        ont_yearlyavgcount[1] = ont_yearlyavgcount[1] + 1;
                                        ont_yearlyavg[1] = ont_yearlyavg[1] + value[i];
                                }
                        }
                        else if (year[i] == 2017)
                        {
                                if (value[i] != 0)
                                {
                                        ont_yearlyavgcount[2] = ont_yearlyavgcount[2] + 1;
                                        ont_yearlyavg[2] = ont_yearlyavg[2] + value[i];
                                }
                        }
                        else if (year[i] == 2018)
                        {
```

```
                if (value[i] != 0)
                {
                    ont_yearlyavgcount[3] = ont_yearlyavgcount[3] + 1;
                    ont_yearlyavg[3] = ont_yearlyavg[3] + value[i];
                }
            }
            else if (year[i] == 2019)
            {
                if (value[i] != 0)
                {
                    ont_yearlyavgcount[4] = ont_yearlyavgcount[4] + 1;
                    ont_yearlyavg[4] = ont_yearlyavg[4] + value[i];
                }
            }
            else if (year[i] == 2020)
            {
                if (value[i] != 0)
                {
                    ont_yearlyavgcount[5] = ont_yearlyavgcount[5] + 1;
                    ont_yearlyavg[5] = ont_yearlyavg[5] + value[i];
                }
            }
            else if (year[i] == 2021)
            {
                if (value[i] != 0)
                {
                    ont_yearlyavgcount[6] = ont_yearlyavgcount[6] + 1;
                    ont_yearlyavg[6] = ont_yearlyavg[6] + value[i];
                }
            }
        }
        // checks if the location is Quebec
        else if(strcmp(location[i], "Quebec") == 0){
            /* depending on what year it is, if the percentage value exists,
                adds the percentage value to its respective index in the
que_yearlyavg array,
                as well as, add +1 to its respective index in the
que_yearlyavgcount
                array which counts the number of values added to the que_yearlyavg
array. */
            if (year[i] == 2015)
            {
                if (value[i] != 0)
                {
                    que_yearlyavgcount[0] = que_yearlyavgcount[0] + 1;
                    que_yearlyavg[0] = que_yearlyavg[0] + value[i];
                }
            }
            else if (year[i] == 2016)
            {
                if (value[i] != 0)
                {
                    que_yearlyavgcount[1] = que_yearlyavgcount[1] + 1;
```

```
                                        que_yearlyavg[1] = que_yearlyavg[1] + value[i];
                                }
                        }
                        else if (year[i] == 2017)
                        {
                                if (value[i] != 0)
                                {
                                        que_yearlyavgcount[2] = que_yearlyavgcount[2] + 1;
                                        que_yearlyavg[2] = que_yearlyavg[2] + value[i];
                                }
                        }
                        else if (year[i] == 2018)
                        {
                                if (value[i] != 0)
                                {
                                        que_yearlyavgcount[3] = que_yearlyavgcount[3] + 1;
                                        que_yearlyavg[3] = que_yearlyavg[3] + value[i];
                                }
                        }
                        else if (year[i] == 2019)
                        {
                                if (value[i] != 0)
                                {
                                        que_yearlyavgcount[4] = que_yearlyavgcount[4] + 1;
                                        que_yearlyavg[4] = que_yearlyavg[4] + value[i];
                                }
                        }
                        else if (year[i] == 2020)
                        {
                                if (value[i] != 0)
                                {
                                        que_yearlyavgcount[5] = que_yearlyavgcount[5] + 1;
                                        que_yearlyavg[5] = que_yearlyavg[5] + value[i];
                                }
                        }
                        else if (year[i] == 2021)
                        {
                                if (value[i] != 0)
                                {
                                        que_yearlyavgcount[6] = que_yearlyavgcount[6] + 1;
                                        que_yearlyavg[6] = que_yearlyavg[6] + value[i];
                                }
                        }
                }
                // checks if the location is British Columbia
                else if(strcmp(location[i], "BritishColumbia") == 0){
                        /* depending on what year it is, if the percentage value exists,
                                adds the percentage value to its respective index in the
bc_yearlyavg array,
                                as well as, add +1 to its respective index in the bc_yearlyavgcount
                                array which counts the number of values added to the bc_yearlyavg
array. */
                        if (year[i] == 2015)
```

```
{
        if (value[i] != 0)
        {
                bc_yearlyavgcount[0] = bc_yearlyavgcount[0] + 1;
                bc_yearlyavg[0] = bc_yearlyavg[0] + value[i];
        }
}
else if (year[i] == 2016)
{
        if (value[i] != 0)
        {
                bc_yearlyavgcount[1] = bc_yearlyavgcount[1] + 1;
                bc_yearlyavg[1] = bc_yearlyavg[1] + value[i];
        }
}
else if (year[i] == 2017)
{
        if (value[i] != 0)
        {
                bc_yearlyavgcount[2] = bc_yearlyavgcount[2] + 1;
                bc_yearlyavg[2] = bc_yearlyavg[2] + value[i];
        }
}
else if (year[i] == 2018)
{
        if (value[i] != 0)
        {
                bc_yearlyavgcount[3] = bc_yearlyavgcount[3] + 1;
                bc_yearlyavg[3] = bc_yearlyavg[3] + value[i];
        }
}
else if (year[i] == 2019)
{
        if (value[i] != 0)
        {
                bc_yearlyavgcount[4] = bc_yearlyavgcount[4] + 1;
                bc_yearlyavg[4] = bc_yearlyavg[4] + value[i];
        }
}
else if (year[i] == 2020)
{
        if (value[i] != 0)
        {
                bc_yearlyavgcount[5] = bc_yearlyavgcount[5] + 1;
                bc_yearlyavg[5] = bc_yearlyavg[5] + value[i];
        }
}
else if (year[i] == 2021)
{
        if (value[i] != 0)
        {
                bc_yearlyavgcount[6] = bc_yearlyavgcount[6] + 1;
                bc_yearlyavg[6] = bc_yearlyavg[6] + value[i];
```

```
                                }
                        }
                }
                // checks if the location is Alberta
                else if(strcmp(location[i], "Alberta") == 0){
                        /* depending on what year it is, if the percentage value exists,
                                adds the percentage value to its respective index in the
alb_yearlyavg array,
                                as well as, add +1 to its respective index in the
alb_yearlyavgcount
                                array which counts the number of values added to the alb_yearlyavg
array. */
                        if (year[i] == 2015)
                        {
                                if (value[i] != 0)
                                {
                                        alb_yearlyavgcount[0] = alb_yearlyavgcount[0] + 1;
                                        alb_yearlyavg[0] = alb_yearlyavg[0] + value[i];
                                }
                        }
                        else if (year[i] == 2016)
                        {
                                if (value[i] != 0)
                                {
                                        alb_yearlyavgcount[1] = alb_yearlyavgcount[1] + 1;
                                        alb_yearlyavg[1] = alb_yearlyavg[1] + value[i];
                                }
                        }
                        else if (year[i] == 2017)
                        {
                                if (value[i] != 0)
                                {
                                        alb_yearlyavgcount[2] = alb_yearlyavgcount[2] + 1;
                                        alb_yearlyavg[2] = alb_yearlyavg[2] + value[i];
                                }
                        }
                        else if (year[i] == 2018)
                        {
                                if (value[i] != 0)
                                {
                                        alb_yearlyavgcount[3] = alb_yearlyavgcount[3] + 1;
                                        alb_yearlyavg[3] = alb_yearlyavg[3] + value[i];
                                }
                        }
                        else if (year[i] == 2019)
                        {
                                if (value[i] != 0)
                                {
                                        alb_yearlyavgcount[4] = alb_yearlyavgcount[4] + 1;
                                        alb_yearlyavg[4] = alb_yearlyavg[4] + value[i];
                                }
                        }
                        else if (year[i] == 2020)
                        {
                                if (value[i] != 0)
                                {
                                        alb_yearlyavgcount[5] = alb_yearlyavgcount[5] + 1;
                                        alb_yearlyavg[5] = alb_yearlyavg[5] + value[i];
                                }
                        }
                        else if (year[i] == 2021)
                        {
                                if (value[i] != 0)
                                {
                                        alb_yearlyavgcount[6] = alb_yearlyavgcount[6] + 1;
                                        alb_yearlyavg[6] = alb_yearlyavg[6] + value[i];
                                }
                        }
                }
                // checks if the location is Canada (excluding territories)
                else if(strcmp(location[i], "Canada(excludingterritories)") == 0){
                        /* depending on what year it is, if the percentage value exists,
                                adds the percentage value to its respective index in the
cad_yearlyavg array,
                                as well as, add +1 to its respective index in the
cad_yearlyavgcount
                                array which counts the number of values added to the cad_yearlyavg
array. */
                        if (year[i] == 2015)
                        {
                                if (value[i] != 0)
                                {
                                        cad_yearlyavgcount[0] = cad_yearlyavgcount[0] + 1;
                                        cad_yearlyavg[0] = cad_yearlyavg[0] + value[i];
                                }
                        }
                        else if (year[i] == 2016)
                        {
                                if (value[i] != 0)
                                {
                                        cad_yearlyavgcount[1] = cad_yearlyavgcount[1] + 1;
                                        cad_yearlyavg[1] = cad_yearlyavg[1] + value[i];
                                }
                        }
                        else if (year[i] == 2017)
                        {
                                if (value[i] != 0)
                                {
                                        cad_yearlyavgcount[2] = cad_yearlyavgcount[2] + 1;
                                        cad_yearlyavg[2] = cad_yearlyavg[2] + value[i];
                                }
                        }
                        else if (year[i] == 2018)
                        {
                                if (value[i] != 0)
                                {
```

```c
                                cad_yearlyavgcount[3] = cad_yearlyavgcount[3] + 1;
                                cad_yearlyavg[3] = cad_yearlyavg[3] + value[i];
                        }
                }
                else if (year[i] == 2019)
                {
                        if (value[i] != 0)
                        {
                                cad_yearlyavgcount[4] = cad_yearlyavgcount[4] + 1;
                                cad_yearlyavg[4] = cad_yearlyavg[4] + value[i];
                        }
                }
                else if (year[i] == 2020)
                {
                        if (value[i] != 0)
                        {
                                cad_yearlyavgcount[5] = cad_yearlyavgcount[5] + 1;
                                cad_yearlyavg[5] = cad_yearlyavg[5] + value[i];
                        }
                }
                else if (year[i] == 2021)
                {
                        if (value[i] != 0)
                        {
                                cad_yearlyavgcount[6] = cad_yearlyavgcount[6] + 1;
                                cad_yearlyavg[6] = cad_yearlyavg[6] + value[i];
                        }
                }
        }
}

/* this for-loop calculates the provincial yearly averages
 by dividing the sum of the yearly percentage values by their
 respective yearly average counts */
for (i = 0; i < 7; i++)
{
        ont_yearlyavg[i] = ont_yearlyavg[i] / ont_yearlyavgcount[i];
        que_yearlyavg[i] = que_yearlyavg[i] / que_yearlyavgcount[i];
        bc_yearlyavg[i] = bc_yearlyavg[i] / bc_yearlyavgcount[i];
        alb_yearlyavg[i] = alb_yearlyavg[i] / alb_yearlyavgcount[i];
        cad_yearlyavg[i] = cad_yearlyavg[i] / cad_yearlyavgcount[i];
}

// displays the yearly percentage averages for diabetes in Ontario
printf("\n\nPart 1C:\n");
printf("Ontario Yearly Average: \n");
printf("Year 2015 Average: %.2lf%%\n", ont_yearlyavg[0]);
printf("Year 2016 Average: %.2lf%%\n", ont_yearlyavg[1]);
printf("Year 2017 Average: %.2lf%%\n", ont_yearlyavg[2]);
printf("Year 2018 Average: %.2lf%%\n", ont_yearlyavg[3]);
printf("Year 2019 Average: %.2lf%%\n", ont_yearlyavg[4]);
printf("Year 2020 Average: %.2lf%%\n", ont_yearlyavg[5]);
printf("Year 2021 Average: %.2lf%%\n", ont_yearlyavg[6]);
```

```
        // displays the yearly percentage averages for diabetes in Quebec
        printf("\nQuebec Yearly Average: \n");
        printf("Year 2015 Average: %.2lf%%\n", que_yearlyavg[0]);
        printf("Year 2016 Average: %.2lf%%\n", que_yearlyavg[1]);
        printf("Year 2017 Average: %.2lf%%\n", que_yearlyavg[2]);
        printf("Year 2018 Average: %.2lf%%\n", que_yearlyavg[3]);
        printf("Year 2019 Average: %.2lf%%\n", que_yearlyavg[4]);
        printf("Year 2020 Average: %.2lf%%\n", que_yearlyavg[5]);
        printf("Year 2021 Average: %.2lf%%\n", que_yearlyavg[6]);

        // displays the yearly percentage averages for diabetes in British Columbia
        printf("\nBritish Columbia Yearly Average: \n");
        printf("Year 2015 Average: %.2lf%%\n", bc_yearlyavg[0]);
        printf("Year 2016 Average: %.2lf%%\n", bc_yearlyavg[1]);
        printf("Year 2017 Average: %.2lf%%\n", bc_yearlyavg[2]);
        printf("Year 2018 Average: %.2lf%%\n", bc_yearlyavg[3]);
        printf("Year 2019 Average: %.2lf%%\n", bc_yearlyavg[4]);
        printf("Year 2020 Average: %.2lf%%\n", bc_yearlyavg[5]);
        printf("Year 2021 Average: %.2lf%%\n", bc_yearlyavg[6]);

        // displays the yearly percentage averages for diabetes in Alberta
        printf("\nAlberta Yearly Average: \n");
        printf("Year 2015 Average: %.2lf%%\n", alb_yearlyavg[0]);
        printf("Year 2016 Average: %.2lf%%\n", alb_yearlyavg[1]);
        printf("Year 2017 Average: %.2lf%%\n", alb_yearlyavg[2]);
        printf("Year 2018 Average: %.2lf%%\n", alb_yearlyavg[3]);
        printf("Year 2019 Average: %.2lf%%\n", alb_yearlyavg[4]);
        printf("Year 2020 Average: %.2lf%%\n", alb_yearlyavg[5]);
        printf("Year 2021 Average: %.2lf%%\n", alb_yearlyavg[6]);

        /* displays the yearly percentage averages for diabetes in Canada (excluding territories)
*/
        printf("\nCanada(excluding territories) Yearly Average: \n");
        printf("Year 2015 Average: %.2lf%%\n", cad_yearlyavg[0]);
        printf("Year 2016 Average: %.2lf%%\n", cad_yearlyavg[1]);
        printf("Year 2017 Average: %.2lf%%\n", cad_yearlyavg[2]);
        printf("Year 2018 Average: %.2lf%%\n", cad_yearlyavg[3]);
        printf("Year 2019 Average: %.2lf%%\n", cad_yearlyavg[4]);
        printf("Year 2020 Average: %.2lf%%\n", cad_yearlyavg[5]);
        printf("Year 2021 Average: %.2lf%%\n", cad_yearlyavg[6]);


//Part 1d
                //loops through every row in the cleaned text file
                for (i = 0; i < k; i++){
                        /*if statements for Ontario*/
                        /*checking if the location is Ontario*/
                        if(strcmp(location[i], "Ontario") == 0){
                                /*depending on which of the 3 age groups is read, and if the
diabetes percentage value exists,
                                that percentage value will be added to the array (proavg_on) that
contains the values for Ontario,
                              also +1 will be added to the array (avgcount_on) which tracks the
number of elements added to the array (proavg_on) */

                              if (strcmp(age[i], "35to49years") == 0) {
                                     if(value[i] != 0){
                                            proavg_on[0] = proavg_on[0] + value[i];
                                            avgcount_on[0] = avgcount_on[0] + 1;
                                     }
                              } else if (strcmp(age[i], "50to64years") == 0) {
                                     if(value[i] != 0){
                                            proavg_on[1] = proavg_on[1] + value[i];
                                            avgcount_on[1] = avgcount_on[1] + 1;
                                     }
                              } else if (strcmp(age[i], "65yearsandover") == 0) {
                                     if(value[i] != 0){
                                            proavg_on[2] = proavg_on[2] + value[i];
                                            avgcount_on[2] = avgcount_on[2] + 1;
                                     }
                              }

                       } /*end of the outer if statement for Ontario*/

                       /*if statements for Quebec*/
                       /*checking if the location is Quebec*/
                       if(strcmp(location[i], "Quebec") == 0){
                              /*depending on which of the 3 age groups is read, and if the
diabetes percentage value exists,
                              that percentage value will be added to the array (proavg_que) that
contains the values for Quebec,
                              also +1 will be added to the array (avgcount_que) which tracks the
number of elements added to the array (proavg_que) */

                              if (strcmp(age[i], "35to49years") == 0) {
                                     if(value[i] != 0){
                                            proavg_que[0] = proavg_que[0] + value[i];
                                            avgcount_que[0] = avgcount_que[0] + 1;
                                     }
                              } else if (strcmp(age[i], "50to64years") == 0) {
                                     if(value[i] != 0){
                                            proavg_que[1] = proavg_que[1] + value[i];
                                            avgcount_que[1] = avgcount_que[1] + 1;
                                     }
                              } else if (strcmp(age[i], "65yearsandover") == 0) {
                                     if(value[i] != 0){
                                            proavg_que[2] = proavg_que[2] + value[i];
                                            avgcount_que[2] = avgcount_que[2] + 1;
                                     }
                              }
                       } /*end of the outer if statement for Quebec*/

                       /*if statements for Alberta*/
                       /*checking if the location is Alberta*/
                       if(strcmp(location[i], "Alberta") == 0){
```

```
                                /*depending on which of the 3 age groups is read, and if the
diabetes percentage value exists,
                                that percentage value will be added to the array (proavg_ab) that
contains the values for Alberta,
                                also +1 will be added to the array (avgcount_ab) which tracks the
number of elements added to the array (proavg_ab) */

                                if (strcmp(age[i], "35to49years") == 0) {
                                        if(value[i] != 0){
                                                proavg_ab[0] = proavg_ab[0] + value[i];
                                                avgcount_ab[0] = avgcount_ab[0] + 1;
                                        }
                                } else if (strcmp(age[i], "50to64years") == 0) {
                                        if(value[i] != 0){
                                                proavg_ab[1] = proavg_ab[1] + value[i];
                                                avgcount_ab[1] = avgcount_ab[1] + 1;
                                        }
                                } else if (strcmp(age[i], "65yearsandover") == 0) {
                                        if(value[i] != 0){
                                                proavg_ab[2] = proavg_ab[2] + value[i];
                                                avgcount_ab[2] = avgcount_ab[2] + 1;
                                        }
                                }
                        } /*end of the outer if statement for Alberta*/

                        /*if statements for British Columbia*/
                        /*checking if the location is British Columbia*/
                        if(strcmp(location[i], "BritishColumbia") == 0){
                                /*depending on which of the 3 age groups is read, and if the
diabetes percentage value exists,
                                that percentage value will be added to the array (proavg_bc) that
contains the values for British Columbia,
                                also +1 will be added to the array (avgcount_bc) which tracks the
number of elements added to the array (proavg_bc) */

                                if (strcmp(age[i], "35to49years") == 0) {
                                        if(value[i] != 0){
                                                proavg_bc[0] = proavg_bc[0] + value[i];
                                        avgcount_bc[0] = avgcount_bc[0] + 1;
                                        }
                                } else if (strcmp(age[i], "50to64years") == 0) {
                                        if(value[i] != 0){
                                                proavg_bc[1] = proavg_bc[1] + value[i];
                                                avgcount_bc[1] = avgcount_bc[1] + 1;
                                        }
                                } else if (strcmp(age[i], "65yearsandover") == 0) {
                                        if(value[i] != 0){
                                                proavg_bc[2] = proavg_bc[2] + value[i];
                                                avgcount_bc[2] = avgcount_bc[2] + 1;
                                        }
                                }
                        } /*end of the outer if statement for British Columbia*/
                        /*if statements Canada(excluding territories)*/
                        /*checking if the location is Canada(excluding territories)*/
                        if(strcmp(location[i], "Canada(excludingterritories)") == 0){
                                /*depending on which of the 3 age groups is read, and if the
diabetes percentage value exists,
                                that percentage value will be added to the array (proavg_cad) that
contains the values for Canada(excluding territories),
                                also +1 will be added to the array (avgcount_cad) which tracks the
number of elements added to the array (proavg_cad) */

                                if (strcmp(age[i], "35to49years") == 0) {
                                        if(value[i] != 0){
                                                proavg_cad[0] = proavg_cad[0] + value[i];
                                                avgcount_cad[0] = avgcount_cad[0] + 1;
                                        }
                                } else if (strcmp(age[i], "50to64years") == 0) {
                                        if(value[i] != 0){
                                                proavg_cad[1] = proavg_cad[1] + value[i];
                                                avgcount_cad[1] = avgcount_cad[1] + 1;
                                        }
                                } else if (strcmp(age[i], "65yearsandover") == 0) {
                                        if(value[i] != 0){
                                                proavg_cad[2] = proavg_cad[2] + value[i];
                                                avgcount_cad[2] = avgcount_cad[2] + 1;
                                        }
                                }
                        } /*end of the outer if statement for Canada*/

                } /*end of the outer for loop*/

        /*5 separate for loops that calculate the average diabetes percentage for each age group,
for each of the 5 locations,
        the average is calculated by dividing the sum of the provinal/national average percentage
values by their
    respective yearly average counts */
        for (i = 0; i < 3; i++)
                proavg_on[i] = proavg_on[i]/avgcount_on[i];

        for (i = 0; i < 3; i++)
                proavg_que[i] = proavg_que[i]/avgcount_que[i];

        for (i = 0; i < 3; i++)
                proavg_ab[i] = proavg_ab[i]/avgcount_ab[i];

        for (i = 0; i < 3; i++)
                proavg_bc[i] = proavg_bc[i]/avgcount_bc[i];

        for (i = 0; i < 3; i++)
                proavg_cad[i] = proavg_cad[i]/avgcount_cad[i];


        //outputting the averages for the 3 different age categories, for each province and Canada
as a whole
```

```
        printf("\n\nQuestion 1D:\n");
        printf("Ontario Average Diabetes Rate by Age Group\n");
        printf("35-49 years old: %.2lf%%\n", proavg_on[0]);
        printf("50-64 years old: %.2lf%%\n", proavg_on[1]);
        printf("65+ years old: %.2lf%%\n", proavg_on[2]);

        printf("\n");
        printf("Quebec Average Diabetes Rate by Age Group\n");
        printf("35-49 years old: %.2lf%%\n", proavg_que[0]);
        printf("50-64 years old: %.2lf%%\n", proavg_que[1]);
        printf("65+ years old: %.2lf%%\n", proavg_que[2]);

        printf("\n");
        printf("Alberta Average Diabetes Rate by Age Group\n");
        printf("35-49 years old: %.2lf%%\n", proavg_ab[0]);
        printf("50-64 years old: %.2lf%%\n", proavg_ab[1]);
        printf("65+ years old: %.2lf%%\n", proavg_ab[2]);

        printf("\n");
        printf("British Columbia Average Diabetes Rate by Age Group\n");
        printf("35-49 years old: %.2lf%%\n", proavg_bc[0]);
        printf("50-64 years old: %.2lf%%\n", proavg_bc[1]);
        printf("65+ years old: %.2lf%%\n", proavg_bc[2]);

        printf("\n");
        printf("Canada (excluding territories) Average Diabetes Rate by Age Group\n");
        printf("35-49 years old: %.2lf%%\n", proavg_cad[0]);
        printf("50-64 years old: %.2lf%%\n", proavg_cad[1]);
        printf("65+ years old: %.2lf%%\n", proavg_cad[2]);


// Part 2

        //Finds the largest provincial average
        provmax = proavg[0];
        for (i = 0; i < 4; i++){
                if (proavg[i] >= provmax){
                        provmax = proavg[i];
                        strcpy(provmaxmin[0], provname[i]);
                }
        }

        //Finds the smallest provincial average
        provmin = proavg[0];
        for (i = 0; i < 4; i++){
                if (proavg[i] <= provmin){
                        provmin = proavg[i];
                        strcpy(provmaxmin[1], provname[i]);
                }
        }
        //Prints result
        printf("\n\nPart 2:\n");
        printf("%s has the largest provincial average with an average of: %.2lf%%\n",
provmaxmin[0], provmax);
        printf("%s has the smallest provincial average with an average of: %.2lf%%\n",
provmaxmin[1], provmin);


// Part 3

        // Compares which provincial averages are above the national average
        for (i = 0; i < 4; i++){
                if (proavg[i] > proavg[4]){
                        avg_above[i] = proavg[i];
                        strcpy(pro_above[i], provname[i]);
                }
        // Compares which provincial averages are below the national average
                if (proavg[i] <= proavg[4]){
                        avg_below[i] = proavg[i];
                        strcpy(pro_below[i], provname[i]);
                }
        }

        //Prints Output
        printf("\n\nPart 3:\n");
        printf("Provinces above the national average(%.2lf%%):\n", proavg[4]);
        for (i = 0; i < 4; i++){
                if(avg_above[i] != 0)
                        printf("%s: %.2lf%%\n", pro_above[i], avg_above[i]);
        }

        printf("\nProvinces below the national average(%.2lf%%):\n", proavg[4]);
        for (i = 0; i < 4; i++){
                if(avg_below[i] != 0)
                        printf("%s: %.2lf%%\n", pro_below[i], avg_below[i]);
                }


// Part 4

        //Finds the largest provincial percentage and records which year and province it is in
        //Note: excluded National Percentages
        for (i = 0; i < k; i++){
                if(strcmp(location[i], "Canada(excludingterritories)") == 1){
                        if (value[i] >= avg_max){
                                avg_max = value[i];
                                year_max = year[i];
                                strcpy(provname_max[0], location[i]);
                        }
                }
        }
        //Finds the smallest provincial percentage, records year and province
        avg_min = avg_max;
        for (i = 0; i < k; i++){
                        if (value[i] < avg_min && value[i] != 0){
                              avg_min = value[i];
                              year_min = year[i];

                              // BC from file is replaced with BC from array(includes a space)
                              if(strcmp(location[i], "BritishColumbia") == 0)
                                     strcpy(provname_min[0], provname[2]);
                              else
                                     strcpy(provname_min[0], location[i]);
                       }
        }

        //Prints Reseult
        printf("\n\nPart 4:\n");
        printf("The Province(s) with the highest percentage of Diabetes is:\n");
        printf("%s in %d with a percentage of: %.2lf%%\n", provname_max[0], year_max, avg_max);

        printf("\nThe Province(s) with the lowest percentage of Diabetes is:\n");
        printf("%s in %d with a percentage of: %.2lf%%\n", provname_min[0], year_min, avg_min);


// Part 5

        //Creates file for Line Graph contataing all provincial averages for all years
        FILE *graphs1;
        graphs1 = fopen("Question5.txt", "w");

        //Prints headers into file
        fprintf(graphs1, "Year\t");
        for(i = 0; i < 5; i++)
               fprintf(graphs1, "%s\t", provname[i]);
        fprintf(graphs1, "\n");

        //Prints all required values into the file
        for(i = 0; i < 7; i++)
               fprintf(graphs1, "%d\t%lf\t%lf\t%lf\t\t%lf\t%lf\n", ls_years[i], ont_yearlyavg[i],
que_yearlyavg[i], bc_yearlyavg[i], alb_yearlyavg[i], cad_yearlyavg[i]);

        fclose(graphs1);


// Part 6

        //creating the text file that will be used to produce the bar graph
    FILE *graphs2;
    graphs2 = fopen("Question6_bar_graph.txt", "w");

        //arrays that contain the age group categories and their corresponding x-axis labels
    char age_group[3][6]= {"35-49", "50-64", "65+"};
    int nums[3] = {0, 1, 2};

        //printing the values into the file bar_graph.txt
    for (int i = 0; i < 3; i++)
        fprintf(graphs2, "%d\t%s\t\t%.2lf\n", nums[i], age_group[i], proavg_cad[i]);
    fclose(graphs2);


       return 0;
}
```

GnuPlot Code

Graph 1

```
set title "{/:Bold Canada's National and Provincial Diabetes Percentages}"
set xlabel "{/:Bold Years}"
set ylabel "{/:Bold Average Percentage of Diabetics}"
set xtics 1
set yrange [5:15]
set xrange [2015:2021]
set key top left

plot "Question5.txt" using 1:2 title "Ontario" with lp, \
     "Question5.txt" using 1:3 title "Quebec" with lp, \
     "Question5.txt" using 1:4 title "British Columbia" with lp, \
     "Question5.txt" using 1:5 title "Alberta" with lp, \
     "Question5.txt" using 1:6 title "Canada(excluding territories" with lp
```

Graph 2

```
set title "{/:Bold Canada's National Diabetes Percentage For Three Age Groups}"
set xlabel "{/:Bold Years}"
set ylabel "{/:Bold Average Percentage of Diabetics}"
set xtics 1
set key top left
set style line 1 lc rgb "red"
set style line 2 lc rgb "blue"
set style line 3 lc rgb "green"
set style fill solid
set boxwidth 0.5

plot "Question6_bar_graph.txt" every ::0::0 using 1:3:xtic(2) title "(35-49)" with boxes ls 1, \
     "Question6_bar_graph.txt" every ::1::2 using 1:3:xtic(2) title "(50-64)" with boxes ls 2, \
     "Question6_bar_graph.txt" every ::2::4 using 1:3:xtic(2) title "(65+)" with boxes ls 3
```