[Course Title]

CPS188: Term Project - Winter 2023

[Project Title]

Analysis of Diabetes Prevalence in Canadian Provinces (2015 -2021)

[Group Leader]

Dip Tandel

[Group Members]

Haider Al Kassab

Marcus Dove

[Submission Date]

April 2, 2023 11:59 PM

[University/Institution]

Toronto Metropolitan University

Introduction:

This report is to analyze the prevalence of diabetes in Quebec, Ontario, Alberta, and British Columbia from 2015 to 2021. The study includes data on the average proportion of the population with diabetes in each province and across Canada, as well as age-specific prevalence. To completely comprehend and enable comparisons, the findings are shown with a line plot and a bar chart.

Findings:

Q1a) Average Diabetes Prevalence (2015-2021):

- Quebec: 10.451221%
- Ontario: 11.702380%
- Alberta: 10.860000%
- British Columbia: 9.670270%

Q1b) Average Percentage of The Whole Population of Canada (2015-2021):

- Canada: 10.869047%

Q1c) Average Percentage of The whole population with diabetes per year from 2015 to 2021:

Quebec:

- 2015: 10.899999%
- 2016: 9.816667%
- 2017: 9.583333%
- 2018: 10.650001%
- 2019: 10.483334%
- 2020: 11.420000%
- 2021: 10.466667%

Ontario:

- 2015: 10.766666%
- 2016: 12.200000%
- 2017: 11.983334%

- 2018: 11.283333%
- 2019: 13.033333%
- 2020: 11.166667%
- 2021: 11.483334%

Alberta:

- 2015: 9.320000%
- 2016: 9.766666%
- 2017: 11.966667%
- 2018: 11.016666%
- 2019: 11.333333%
- 2020: 12.880000%
- 2021: 9.816667%

British Columbia:

- 2015: 9.300000%
- 2016: 8.533334%
- 2017: 10.140000%
- 2018: 8.516666%
- 2019: 11.440001%
- 2020: 9.040000%
- 2021: 11.650000%

Canada:

- 2015: 10.599999%
- 2016: 10.700001%
- 2017: 10.950000%
- 2018: 10.783333%
- 2019: 11.700001%
- 2020: 10.599999%
- 2021: 10.750000%

Q1d) The average percentage of the population with diabetes by age group for each province and Canada from 2015 to 2021:

- Quebec: 3.353846% (35-49 years), 9.057143% (50-64 years), 18.435715% (65+ years)
- Ontario: 4.642857% (35-49 years), 11.221428% (50-64 years), 19.242857% (65+ years)
- Alberta: 4.458333% (35-49 years), 10.285714% (50-64 years), 16.921429% (65+ years)
- British Columbia: 3.433333% (35-49 years), 7.914286% (50-64 years), 15.435716% (65+ years)
- Canada: 4.064286% (35-49 years), 10.328572% (50-64 years), 18.214285% (65+ years)

2. Highest and Lowest Provincial Prevalence:

- Highest: Ontario (11.702380%)
- Lowest: British Columbia (9.670270%)

3. Provinces Above and Below the National Average:

- Above National Average: Ontario
- Below National Average: Quebec, Alberta, and British Columbia

4. Year and Province with the Highest and Lowest Prevalence:

- Highest: 2019, Ontario
- Lowest: 2018, British Columbia

Explanation of code for each question:

Q1. (reading and saving each element)
Around the first 200 lines of the code is just to read each element of the csv file. A structure was created which consists of all the different categories in the csv file, such as 'value', or 'date'. Then an array of the aforementioned structure was made in order to save each line of the file. This was done by simply using various File IO functions and looping through each line until the end was reached. This was simple since each value is separated by delimiters, commas in this case. However, reading each element using the separating delimiters is not enough, the values are also surrounded by quotes, which is why another function was created specifically to remove quotes around each string, by re-reading and re-writing the string, but ignoring the quotes. Notice that each word/value read is still a string, this is where the atoi() and atof() functions come into use, they simple convert the string into either an integer, or float. After

this is when each value can finally be saved into the structure: s[i].date for example, saves the value read into the date element of the structure. Since the order of the elements/categories is known, each element is saved to its corresponding location in the structure based on the order it is read; date is first, then location, and so forth. The 'i' indicates the index of the array (line of the file). The 'i' value increases each iteration of the loop and stops at the end of the file/final line.

Q1a)

The key to using this 'struct' is knowing which indices (i), belong to which province or location. This was simple since opening the csv file easily show which line corresponds to which province; indices [42] to [83] correspond to Quebec, [84] to [125] correspond to Ontario, [126] to [167] correspond to Alberta, and [168] to [209] correspond to British Columbia. Knowing this information, the rest of the question becomes fairly simple, all that was required was to loop through the struct using a for-loop which loops through the indices corresponding to the specific province and adding the .value element from the struct to an ongoing total. A counter variable was made to keep track of the amount of times a value was added after which the total was divided by the counter to get an average. However, since '0' values are to be ignored, a simple if-statement was made so that the counter only increased if the value was not a zero.

Q1b)

Part b was the in the exact same way as part a, except the indices for Canada ranged from [0] to [41].

Q1c)

At this point, there were so many calculations required that a dedicated function was used to compute them in fewer lines of code. The function requires only four variables, the beginning index, the final index, specific year for which the calculation is required, and the structure array which has each element of the csv file. The function works almost identically to how part a and b was done. An average and counter variable was made and a for-loop was made which loops through the start and end using the indices provided into the function. But now, the if-statement also requires that the .date portion of the struct is equal to the year which was passed into the function, this way only the values needed for the specific year are added. The function then returns the average for that specific year. In the main function, this function is actually called using a loop since calculations were required from 2015-2021, the main loop simply called the function over and over, each time passing through each year for which the calculation was required, then adding each year's average to a float array. This works similar to a double for-loop, except the 'inner' for-loop is another function in this case.

Q1d)

Another function was made specifically for part d and it works similarly to the function created for part c. This function needs the pointer to a char array created in main rather than a specific year. This char array is simply an array of strings (2-D char array) which consists of the strings: "35 to 49 years", "50 to 64 years", "65 years and over". These are the strings which were read and saved into the .agegroup element of the struct, so this can be used by the function to find specific values corresponding to each of the age groups. A for-loop in the main function was created to loop specifically three times since there are only three age groups, and it passes in each age group (string) for which the calculation is required, saving each average returned into another array which was created in the main function.

Q2.

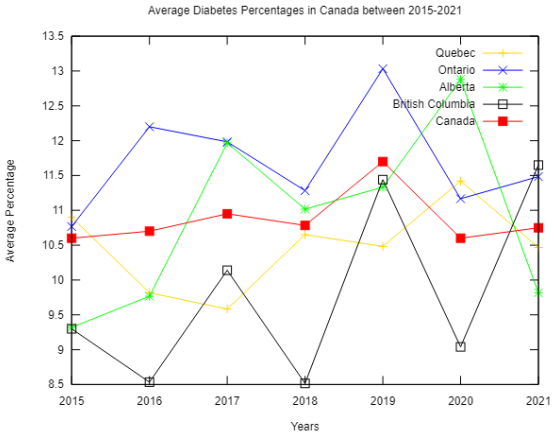
- Initialize variables minPercentage and maxPercentage to qcAvg, and minProvince and maxProvince to "Quebec".
- Compare the average diabetes percentages for Ontario, Alberta, and British Columbia against minPercentage and maxPercentage, updating them and their corresponding province names accordingly.
- Print the provinces with the highest and lowest percentages of diabetes.

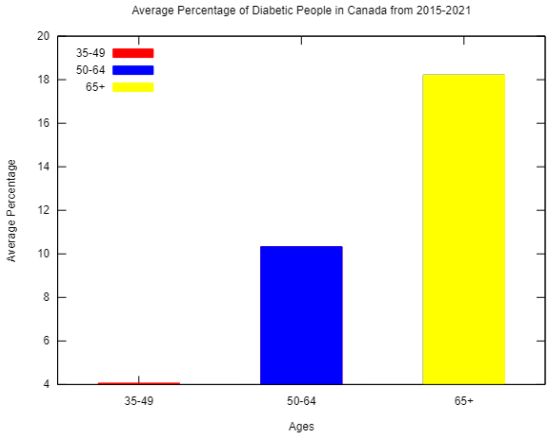
Q3.

- Print the provinces with diabetes percentages above the national average (all years and age groups) by checking if their averages are greater than canAvg.
- Print the provinces with diabetes percentages below the national average (all years and age groups) by checking if their averages are less than canAvg.

Q4.

- Initialize variables for the highest and lowest years, provinces, and percentages.
- Loop through the years from 2015 to 2021 and compare the diabetes percentages for each province in each year to the current highest and lowest percentages.
- Update the highest and lowest years, provinces, and percentages based on the comparison results.
- Print the year and province with the highest and lowest percentages of diabetes (all age groups).





Q5&6.

The gnuplot source scripts are used to generate charts of diabetes percentages in Canada from 2015 to 2021. They use data visualization to give insights into diabetes patterns across time and across different age groups. These are the results of both gnuplots:

5. Plotting Lines (first gnuplot):
   To create the line plot, we used different line styles and colors to differentiate each province and the national average. The data was plotted using the 'plot' command and the 'with lp' (line points) option to show both lines and points on the graph. The x-axis is labeled 'Years' and the y-axis is labeled 'Average Percentage.' The graph has a title and a legend to identify each province and the national average. To read the data for both files, a txt file was created that had the data arranged in columns. The code was written in a way that would identify which columns to take the data from and display on the graph. The colors for both graphs were changed by using the line style command to change the color for each set of data. To separate the lines of code that need to be read for each set of data, we used “, \” to indicate if there is more data to be read underneath each line of code that is going to be displayed.

6. Bar Chart (second gnuplot):
   To separate each age category, we utilized various line colours to construct the bar chart. To change the look of the bars, use the'set boxwidth' and'set style fill solid' commands. The x-axis reads 'Ages,' while the y-axis reads 'Average Percentage.' Each age group is identified with a title and a legend on the graph. The same syntaxes, such as '', '' and the line style, are used in this graph. Because of the distinct nature of the data, a separate txt file was utilized for this graph.

Conclusion:

This analysis shows that Ontario has the greatest prevalence of diabetes among the provinces studied, while British Columbia has the lowest. Diabetes prevalence rates in Quebec, Alberta, and British Columbia are all lower than the national average. Diabetes prevalence rises with age, as demonstrated

by the bar chart. This initiative provided a better knowledge of diabetes trends in Canada and allowed for comparisons across provinces.

If this experiment were to be repeated, a more comprehensive analysis may be undertaken by include new provinces, a greater range of years, and investigating potential links between demographic characteristics and diabetes prevalence. This would offer a more complete picture of the variables driving diabetes prevalence across the country.

Programs:

Project.c:

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

//putting the entire file in a custom structure
typedef struct {
   int date;
   char geo[31];
   char dguid[12];
   char agegroup[18];
   char sex[8];
   char indicators[9];
   char characteristics[8];
   char uom[8];
   int uomid;
   char scalarfactor[6];
   int scalarid;
   char vector[11];
   char coordinate[12];
   float value;
   char status[2];
   char symbol[2];
   char terminated[2];
   int decimals;
} sheet;

//function to remove quotes from string
void cleanint(char *input) {
   int j = 0;
   //goes through the whole string and rewrites it without quotes
   for (int i = 0; input[i] != '\0'; i++) {
      if (input[i] != '"') {
         input[j++] = input[i];
      }
   }
   input[j] = '\0'; //adds \0 at end of string to marks the end
}
```

```
//function to calculate the avg population of a province with diabetes for a specific year
float yearlyavg(int start, int end, int year, sheet s[210]) {
	float avg = 0;
	int counter = 0;
	for (int i = start; i <= end; i++) {
		if (s[i].value != 0 && s[i].date == year) { //dont count empty values and only count the values of the year needed
			avg = avg + s[i].value;
			counter++;
		}
	}
	return (avg/counter);
}

//function to calculate the avg population of a province with diabetes across an age range
float ageavg(int start, int end, char *range, sheet s[210]) {
	float avg = 0;
	int counter = 0;
	for (int i = start; i <= end; i++) {
		//strcmp() works by comparing 2 strings and returning 0 if they are the same
		if (s[i].value != 0 && strcmp(s[i].agegroup,range) == 0) { //dont count empty values and only count the age range needed
			avg = avg + s[i].value;
			counter++;
		}
	}
	return (avg/counter);
}

int main (){

	//the spreadsheet has 210 rows
	sheet s[210];

	FILE* data = fopen("statscan_diabetes.csv", "r");
	//variable of file is now "data"

	int linelength = 255; //arbitrarily chosen length of a single line
	char line[linelength]; //this will hold the line
```

```
//read the header line here to prevent it from being read in the loop
fgets(line, linelength, data);

char *sp; //temporary pointer

//loop which will read data line by line
int i = 0; //index of line

while (i < 210) { //Goes until end of file
        //printf("\n%s", line);
        fgets(line, linelength, data);
        //convert date from string to int and save
        sp = strtok(line, ",");
        cleanint(sp);
        s[i].date = atoi(sp);
        printf("%d - ", s[i].date);

        //save geo
        sp = strtok(NULL, ","); //read from where left of till another comma
        cleanint(sp);
        strcpy(s[i].geo,sp);
        printf("%s - ", s[i].geo);

        //save dguid
        sp = strtok(NULL, ","); //read from where left of till another comma
        cleanint(sp);
        strcpy(s[i].dguid,sp);
        printf("%s - ", s[i].dguid);

        //save agegroup
        sp = strtok(NULL, ","); //read from where left of till another comma
        cleanint(sp);
        strcpy(s[i].agegroup,sp);
        printf("%s - ", s[i].agegroup);

        //save sex
        sp = strtok(NULL, ","); //read from where left of till another comma
        cleanint(sp);
        strcpy(s[i].sex,sp);
        printf("%s - ", s[i].sex);
```

```
//save indicators
sp = strtok(NULL, ","); //read from where left of till another comma
cleanint(sp);
strcpy(s[i].indicators,sp);
printf("%s - ", s[i].indicators);

//save characteristics
sp = strtok(NULL, ","); //read from where left of till another comma
cleanint(sp);
strcpy(s[i].characteristics,sp);
printf("%s - ", s[i].characteristics);

//save uom
sp = strtok(NULL, ","); //read from where left of till another comma
cleanint(sp);
strcpy(s[i].uom,sp);
printf("%s - ", s[i].uom);

//convert uomid from string to int and save
sp = strtok(NULL, ",");
cleanint(sp);
s[i].uomid = atoi(sp);
printf("%d - ", s[i].uomid);

//save scalarfactor
sp = strtok(NULL, ","); //read from where left of till another comma
cleanint(sp);
strcpy(s[i].scalarfactor,sp);
printf("%s - ", s[i].scalarfactor);

//convert scalarid from string to int and save
sp = strtok(NULL, ",");
cleanint(sp);
s[i].scalarid = atoi(sp);
printf("%d - ", s[i].scalarid);

//save vector
sp = strtok(NULL, ","); //read from where left of till another comma
cleanint(sp);
strcpy(s[i].vector,sp);
printf("%s - ", s[i].vector);
```

```
        //save coordinate
        sp = strtok(NULL, ","); //read from where left of till another comma
        cleanint(sp);
        strcpy(s[i].coordinate,sp);
        printf("%s - ", s[i].coordinate);

        //convert value from string to float and save
        sp = strtok(NULL, ",");
        cleanint(sp);
        s[i].value = atof(sp);
        printf("%.2f - ", s[i].value);

        //save status
        sp = strtok(NULL, ","); //read from where left of till another comma
        cleanint(sp);
        strcpy(s[i].status,sp);
        printf("%s - ", s[i].status);

        //save symbol
        sp = strtok(NULL, ","); //read from where left of till another comma
        cleanint(sp);
        strcpy(s[i].symbol,sp);
        printf("%s - ", s[i].symbol);

        //save terminated
        sp = strtok(NULL, ","); //read from where left of till another comma
        cleanint(sp);
        strcpy(s[i].terminated,sp);
        printf("%s - ", s[i].terminated);

        //convert decimals from string to int and save
        sp = strtok(NULL, ",");
        cleanint(sp);
        s[i].decimals = atoi(sp);
        printf("%d \n", s[i].decimals);

        i++;
}
//close file (end)
fclose(data);
```

```
	printf("\n");


	printf("//----------Q1a)----------//\n");
	//average percentage of the whole population of Quebec with diabetes
	float qcAvg = 0;
	int counter1 = 0;
	for (int i = 42; i <= 83; i++) { //lines for Quebec
		if (s[i].value != 0) { //dont count empty values
			qcAvg = qcAvg + s[i].value;
			counter1++;
		}
	}
	qcAvg = qcAvg/counter1;
	printf("The Average percentage of the whole population of Quebec with diabetes from 2015 to 2021 (inclusive): %f\n", qcAvg);
	//average percentage of the whole population of Ontario with diabetes
	float onAvg = 0;
	int counter2 = 0;
	for (int i = 84; i <= 125; i++) { //lines for Ontario
		if (s[i].value != 0) { //dont count empty values
			onAvg = onAvg + s[i].value;
			counter2++;
		}
	}
	onAvg = onAvg/counter2;
	printf("The Average percentage of the whole population of Ontario with diabetes from 2015 to 2021 (inclusive): %f\n", onAvg);
	//average percentage of the whole population of Alberta with diabetes
	float abAvg = 0;
	int counter3 = 0;
	for (int i = 126; i <= 167; i++) { //lines for Alberta
		if (s[i].value != 0) { //dont count empty values
			abAvg = abAvg + s[i].value;
			counter3++;
		}
	}
	abAvg = abAvg/counter3;
	printf("The Average percentage of the whole population of Alberta with diabetes from 2015 to 2021 (inclusive): %f\n", abAvg);
	//average percentage of the whole population of British Columbia with diabetes
```

```
	float bcAvg = 0;
	int counter4 = 0;
	for (int i = 168; i <= 209; i++) { //lines for British Columbia
		if (s[i].value != 0) { //dont count empty values
			bcAvg = bcAvg + s[i].value;
			counter4++;
		}
	}
	bcAvg = bcAvg/counter4;
	printf("The Average percentage of the whole population of British Columbia with diabetes from 2015 to 2021 (inclusive): %f \n \n", bcAvg);


	printf("//----------Q1b)----------//\n");
	//average percentage of the whole population of Canada with diabetes
	float canAvg = 0;
	int counter5 = 0;
	for (int i = 0; i <= 41; i++) { //lines for Canada
		if (s[i].value != 0) { //dont count empty values
			canAvg = canAvg + s[i].value;
			counter5++;
		}
	}
	canAvg = canAvg/counter5;
	printf("The Average percentage of the whole population of Canada with diabetes from 2015 to 2021 (inclusive): %f\n\n", canAvg);


	printf("//----------Q1c)----------//\n");
	//yearly average percentages of the whole population of Quebec with diabetes
	float qcAvgList[7];
	for (int i = 2015; i <= 2021; i++) {
		qcAvgList[i-2015] = yearlyavg(42, 83, i, s);
		printf("The %d Average percentage of the whole population of Quebec with diabetes: %f\n", i, qcAvgList[i-2015]);
	}
	printf("\n");
	//yearly average percentages of the whole population of Ontario with diabetes
	float onAvgList[7];
	for (int i = 2015; i <= 2021; i++) {
		onAvgList[i-2015] = yearlyavg(84, 125, i, s);
```

```
		printf("The %d Average percentage of the whole population of Ontario with diabetes: %f\n", i, onAvgList[i-2015]);
	}
	printf("\n");
	//yearly average percentages of the whole population of Alberta with diabetes
	float abAvgList[7];
	for (int i = 2015; i <= 2021; i++) {
		abAvgList[i-2015] = yearlyavg(126, 167, i, s);
		printf("The %d Average percentage of the whole population of Alberta with diabetes: %f\n", i, abAvgList[i-2015]);
	}
	printf("\n");
	//yearly average percentages of the whole population of British Columbia with diabetes
	float bcAvgList[7];
	for (int i = 2015; i <= 2021; i++) {
		bcAvgList[i-2015] = yearlyavg(168, 209, i, s);
		printf("The %d Average percentage of the whole population of British Columbia with diabetes: %f\n", i, bcAvgList[i-2015]);
	}
	printf("\n");
	//yearly average percentages of the whole population of Canada with diabetes
	float canAvgList[7];
	for (int i = 2015; i <= 2021; i++) {
		canAvgList[i-2015] = yearlyavg(0, 41, i, s);
		printf("The %d Average percentage of the whole population of Canada with diabetes: %f\n", i, canAvgList[i-2015]);
	}
	printf("\n");


	printf("//----------Q1d)----------//\n");
	char ageRangeList[3][18] = {"35 to 49 years","50 to 64 years","65 years and over"}; //will be used to easily pass to function
	//average percentage of the population of Quebec with diabetes within an age group
	float qcAgeList[3];
	for (int i = 0; i < 3; i++) {
		qcAgeList[i] = ageavg(42, 83, ageRangeList[i], s);
		printf("The Average percentage from 2015 to 2021 (inclusive) of the population of Quebec with diabetes from ages %s is: %f\n", ageRangeList[i], qcAgeList[i]);
	}
	printf("\n");
```

```
	//average percentage of the population of Ontario with diabetes within an age group
	float onAgeList[3];
	for (int i = 0; i < 3; i++) {
		onAgeList[i] = ageavg(84, 125, ageRangeList[i], s);
		printf("The Average percentage from 2015 to 2021 (inclusive) of the population of
Ontario with diabetes from ages %s is: %f\n", ageRangeList[i], onAgeList[i]);
	}
	printf("\n");
	//average percentage of the population of Alberta with diabetes within an age group
	float abAgeList[3];
	for (int i = 0; i < 3; i++) {
		abAgeList[i] = ageavg(126, 167, ageRangeList[i], s);
		printf("The Average percentage from 2015 to 2021 (inclusive) of the population of
Alberta with diabetes from ages %s is: %f\n", ageRangeList[i], abAgeList[i]);
	}
	printf("\n");
	//average percentage of the population of British Columbia with diabetes within an age group
	float bcAgeList[3];
	for (int i = 0; i < 3; i++) {
		bcAgeList[i] = ageavg(168, 209, ageRangeList[i], s);
		printf("The Average percentage from 2015 to 2021 (inclusive) of the population of
British Columbia with diabetes from ages %s is: %f\n", ageRangeList[i], bcAgeList[i]);
	}
	printf("\n");
	//average percentage of the population of Canada with diabetes within an age group
	float canAgeList[3];
	for (int i = 0; i < 3; i++) {
		canAgeList[i] = ageavg(0, 41, ageRangeList[i], s);
		printf("The Average percentage from 2015 to 2021 (inclusive) of the population of
Canada with diabetes from ages %s is: %f\n", ageRangeList[i], canAgeList[i]);
	}
	printf("\n");

		printf("//----------Q2)----------//\n");
	// Determine which province has the highest and lowest percentage of diabetes
	float minPercentage = qcAvg;
	float maxPercentage = qcAvg;
	char *minProvince = "Quebec";
	char *maxProvince = "Quebec";

	if (onAvg < minPercentage) {
```

```
            minPercentage = onAvg;
            minProvince = "Ontario";
        } else if (onAvg > maxPercentage) {
            maxPercentage = onAvg;
            maxProvince = "Ontario";
        }

        if (abAvg < minPercentage) {
            minPercentage = abAvg;
            minProvince = "Alberta";
        } else if (abAvg > maxPercentage) {
            maxPercentage = abAvg;
            maxProvince = "Alberta";
        }

        if (bcAvg < minPercentage) {
            minPercentage = bcAvg;
            minProvince = "British Columbia";
        } else if (bcAvg > maxPercentage) {
            maxPercentage = bcAvg;
            maxProvince = "British Columbia";
        }

        printf("The province with the highest percentage of diabetes (all years and age groups): %s\n",
maxProvince);
        printf("The province with the lowest percentage of diabetes (all years and age groups): %s\n\n",
minProvince);

        printf("//----------Q3)----------//\n");
        // Indicate provinces with diabetes percentages above and below the national average
        printf("Provinces with diabetes percentages above the national average (all years and age
groups):\n");
        if (qcAvg > canAvg) printf("Quebec\n");
        if (onAvg > canAvg) printf("Ontario\n");
        if (abAvg > canAvg) printf("Alberta\n");
        if (bcAvg > canAvg) printf("British Columbia\n");

        printf("\nProvinces with diabetes percentages below the national average (all years and age
groups):\n");
        if (qcAvg < canAvg) printf("Quebec\n");
        if (onAvg < canAvg) printf("Ontario\n");
```

```
if (abAvg < canAvg) printf("Alberta\n");
if (bcAvg < canAvg) printf("British Columbia\n");

printf("\n");

printf("//----------Q4)----------//\n");
// Indicate which year and province have the highest and lowest percentage of diabetes
int maxYear = 2015, minYear = 2015;
char *maxYearProvince = "Quebec";
char *minYearProvince = "Quebec";
float maxYearPercentage = qcAvgList[0];
float minYearPercentage = qcAvgList[0];

for (int year = 2015; year <= 2021; year++) {
        int index = year - 2015;
        if (qcAvgList[index] < minYearPercentage) {
                minYearPercentage = qcAvgList[index];
                minYear = year;
                minYearProvince = "Quebec";
        } else if (qcAvgList[index] > maxYearPercentage) {
                maxYearPercentage = qcAvgList[index];
                maxYear = year;
                maxYearProvince = "Quebec";
        }

        if (onAvgList[index] < minYearPercentage) {
                minYearPercentage = onAvgList[index];
                minYear = year;
                minYearProvince = "Ontario";
        } else if (onAvgList[index] > maxYearPercentage) {
                maxYearPercentage = onAvgList[index];
                maxYear = year;
                maxYearProvince = "Ontario";
        }

        if (abAvgList[index] < minYearPercentage) {
                minYearPercentage = abAvgList[index];
                minYear = year;
                minYearProvince = "Alberta";
        } else if (abAvgList[index] > maxYearPercentage) {
                maxYearPercentage = abAvgList[index];
```

```
                maxYear = year;
                maxYearProvince = "Alberta";
            }

            if (bcAvgList[index] < minYearPercentage) {
                minYearPercentage = bcAvgList[index];
                minYear = year;
                minYearProvince = "British Columbia";
            } else if (bcAvgList[index] > maxYearPercentage) {
                maxYearPercentage = bcAvgList[index];
                maxYear = year;
                maxYearProvince = "British Columbia";
            }
        }

        printf("The year and province with the highest percentage of diabetes (all age groups): %d, %s\n", maxYear, maxYearProvince);
        printf("The year and province with the lowest percentage of diabetes (all age groups): %d, %s\n\n", minYear, minYearProvince);

        printf("\n//----------Final Note----------//\nQuestions 5 & 6 are done with GNU Plot");
        return (0);
}
```

GNUPlot 1(Line):

```
set style line 1 lc rgb "gold"
set style line 2 lc rgb "blue"
set style line 3 lc rgb "green"
set style line 4 lc rgb "black"
set style line 5 lc rgb "red"


set title 'Average Diabetes Percentages in Canada between 2015-2021'
set xlabel 'Years'
set ylabel 'Average Percentage'
set xtics 1
plot 'o.txt' using 1:2 title "Quebec" with lp ls 1, \
     'o.txt' using 1:3 title "Ontario" with lp ls 2, \
     'o.txt' using 1:4 title "Alberta" with lp ls 3, \
```

```
'o.txt' using 1:5 title "British Columbia" with lp ls 4, \
'o.txt' using 1:6 title "Canada" with lp ls 5
```

o.txt

2015	10.899999	10.766666	9.320000	9.300000	10.599999
2016	9.816667	12.200000	9.766666	8.533334	10.700001
2017	9.583333	11.983334	11.966667	10.140000	10.950000
2018	10.650001	11.283333	11.016666	8.516666	10.783333
2019	10.483334	13.033333	11.333333	11.440001	11.700001
2020	11.420000	11.166667	12.880000	9.040000	10.599999
2021	10.466667	11.483334	9.816667	11.650000	10.750000

GNUPlot 2 (Bar Chart):

```
set style line 1 lc rgb "red"
set style line 2 lc rgb "blue"
set style line 3 lc rgb "yellow"

set xlabel 'Ages'
set ylabel 'Average Percentage'

set boxwidth 0.5
set style fill solid
set key left top
set title "Average Percentage of Diabetic People in Canada from 2015-2021"
plot "b.txt" every ::0::1 using 1:3:xtic(2) title "35-49" with boxes ls 1, \
        "b.txt" every ::1::2 using 1:3:xtic(2) title "50-64"  with boxes ls 2, \
    "b.txt" every ::2::2 using 1:3:xtic(2) title "65+" with boxes ls 3
```

b.txt

```
0  35-49  4.064286
1  50-64  10.328572
2  65+    18.214285
```