CPS188 Term Project Report

Brian Mami (Section 1), Jason Su (Section 7), Saahil Kher (Section 3), and Samuel Azar (Section 1).

Dr. Denis Hamelin
CPS188

TABLE OF CONTENTS

INTRODUCTION

Diabetes is an incurable, long-lasting condition that affects how the body transforms food into energy. The body breaks down these foods into glucose - otherwise known as sugar. The glucose enters the bloodstream and raises a person's blood sugar levels, which prompts the pancreas to release a hormone known as insulin. Insulin allows for the blood sugar to enter the cells at a regulated rate, which in turn allows the body to use the cells as energy. With diabetes, the pancreas cannot produce insulin, or fails to produce an efficient amount for the body to use. With no insulin, blood sugar levels within the body cannot be regulated, which can be fatal and cause serious health effects in the future. This report will analyze the statistics of people living with diabetes in Canada, as well as provide calculations and conclusions on the population of diabetics in Canada.

The explanation of each question and portion of the assignment will be provided in singular paragraphs throughout the report.

EXPLANATION OF MAIN CODE

The purpose of the main code is to read the data from the file to analyze the average percentage of people who have diabetes within each province, the year the data was recorded, as well as the age group of those with diabetes. The code then determines the highest and lowest averages for each province and year of data publication. The code defines two structs; one to store the data obtained from the file and one to store the average data.

The ‘question1’ struct has five members: ‘ref_date’, ‘geo’, ‘age_group’, ‘sex’, and ‘value’. These members represent the year of the data, the geographic location, age group of those with diabetes, sex of the population recorded, and the percentage of population with diabetes.

The ‘average’ struct has three members: ‘avg’, ‘values’, and ‘num’. These members correspond to the average of all percentages for a particular geographic location and year, the sum of all percentages for a particular geographic location and year, and the number of data values for a particular geographic location and year.

This code defines multiple different functions, including ‘removeQuotes’ which removes the quotes from a string; ‘getAverage’ which counts the number of data values and their sum for each geographical location and year to calculate the average; ‘sortAvg’ which sorts the provincial average array to obtain the highest and lowest averages; and ‘sortArrAvg’, which determines the lowest and highest average for each year and province.

The code stores the data from the file in an array of ‘question1’ structs. It then calculates the average diabetes prevalence for each year and location using the ‘getAverage’ function and stores the data in an array of ‘average’ structs. Ultimately, it determines the highest and lowest averages for each province and year using the ‘sortArrAvg’ function and prints the information to the console.

Within the main code, this part can be seen in red.

QUESTION ONE

The first part of this question required us to output various average percentages of those who are diabetic based on provincial location, age, and date. The input portion for this particular part consisted of several structs that grouped data such as numerical values (year, percentage of population, etc.), and chars/strings (locations and genders).

There is a simple file opening statement which opens the csv file containing the data, followed by listing variables needed to store data consisting of the necessary average values for provinces or year which are to be computed. The data file was then read line by line with the use of a while loop that stopped once the total number of entries was read. The line of data read would include the necessary values however it would also include punctuation and unwanted numbers. Another while loop would then be used to remove each quotation mark in the line with the "removeQuotes" function. This function would also obtain the token in between the quotes and assign the column in which it was located. With a series of conditional statements, the wanted values would then be stored in its respective structure.

After the average values of each province are calculated (By taking the sum of the values and dividing it by the number of values in the array), the numbers are then outputted through the use of 4 print statements for each different provincial average, which is the answer for question 1a. 1b is solved in a similar manner and insteads displays and calculates the average for Canada.

1c asks for the yearly average of each province and this is done with the 2D array of structures. The 2D array of structures arrAverage[][], was used to determine the specific average percentages of diabetic Canadians, along with a combination of conditional statements in order to separate data based on the parameters provided from the first question (province, age, and year). The first dimension of the 2D array stored the province (first index: for Canada) associated with the data collected while the second dimension stored the year (first index: all years combined). Since the array was also a structure, the sum of percentages, total amount of percents, and the average were able to be accessed for given province and year. The averages were calculated by sending the values obtained in part a to the function “getAverage" which would perform the calculations and assign the values into its respective index using a for loop nested within another for loop.

Finally, for 1d, the data is collected and calculated in the same way as 1a and 1b, and is then displayed.

Within the main code, this part can be seen in Orange.

QUESTION TWO

For loops are used for this part of the question, and the average for each province is stored in the array “provAvg”. The for loop compares each of the provincial averages, and sorts them from lowest to highest by comparing each value and switching their index if one is greater than the other. The main function then runs the “provAvg” array through two conditional statements to determine the first and last values in the array. Depending on the order, the main function then outputs the province with the lowest and highest percentages.

Within the main code, this part can be seen in Green.

QUESTION THREE

This question requires an analysis of which countries are above or below the national average of the diabetes population in Canada. The code consists of if statements analyzing the averages for each province's diabetes population, and comparing them to the national average in Canada. If the province's averages are above the national average, the program prints the province’s names indicating that the province is above the national average. If the province’s averages are below the national average, the program prints the province’s names indicating that they are below the national average.

Within the main code, this part can be seen in blue.

QUESTION FOUR

This question requires a determination of which year and province has the highest population percentage of diabetes compared to the other provinces. This code calls a function called “sortArrAvg()”, and this function contains five arguments; “arrAverage” is an array of the average values of diabetes in different provinces over different years; “provH” is a character array that stores the name of the province with the highest average; “&yearH” is a pointer to an integer variable that stores the year with the highest average; “provL” is a character array that stores the name of the province with the lowest average; and “&yearL” which is a pointer to an integer variable that stores the year with the lowest average. This function will sort the “arrAverage” array and determine the lowest and highest values, and their corresponding province and year and store these values in “provH”, “&yearH”, “provL” and “&yearL” variables. The next few lines of the code print the values of the year and province with the lowest percentage of diabetes, as well as the province with the highest percentage of diabetes.

Within the main code, this part can be seen in purple.

QUESTION FIVE

The graph consists of labels which indicate the geographical area in which the data is from, as well as outlining the various features of the graph such as the title, increment of increase for each of the axes, and positioning of the graph legend. The final component involves getting the information from the text files previously determined in order to output them via the graph, which consists of a statement for Canada and each province respectively, along with specifications for the points on the graph by using the plot function.

Within the GNUPlot code, this part can be seen in pink.

QUESTION SIX

This GNUplot code creates a plot of the average percentages of people with diabetes differentiating between several age groups. The plot has a title, x and y axis labels and a y-axis tick interval of 2. The boxes in the plot are 0.5 units wide and have a solid fill style. The y-axis has grid lines and ranges from 0-30 which is the percentage of the population that has diabetes. The plot has three bars, each with a different color determining the percentage of the population within a certain age group. The information from the text files previously determined are then integrated into the bar graph and the bars demonstrate the percentage.

Within the GNUPlot code, this part can be seen in Yellow.

CONCLUSION

The overall experience of completing the project was enjoyable. The group analyzed our respective strengths and weaknesses to determine who was better at coding, and assigned them to compile the statistics in which the assignment was based on. To maximize efficiency, two members were responsible for creating the code, while the remaining two members would organize the findings within the report. Of the two members assigned to complete the code, one member was tasked with completing the main code - which consisted of gathering statistics from the txt file - and another member created the code to solve the assigned questions. The remaining two members split the questions within the report and explained them based on the code. The division of work ultimately contributed to the positive experience of the project.

If the group were to reattempt the project, one thing that could be done differently is compiling the report while simultaneously creating the code. Within the project, the two members creating the code worked separately from the two students creating the report. If the project were to be repeated, it would be more methodical for the members coding to create notes to provide to the members creating the report, which would alleviate confusion between the two parties. The notes given on behalf of the members creating the code would explain the code to the members who had to create the report, which would allow them to understand the code and thus provide proper explanations for each question.

MAIN CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_RECORDS 211

typedef struct question1 //Struct to store all values from file
{
        int ref_date; //year;
        char geo[50]; //country/province
        char age_group[50]; //the age group
        char sex[50]; //gender
        double value; //% of pop. that has diabetes
} question1;

typedef struct average //Struct to store all values for the average
{
        double avg;  //average for all years and age groups
        double values;  //sum of all percentages
        double num;  //total number of values entered
} average;

//Function to remove quotes from the data in the file
void removeQuotes(char* raw, char* result){
        int len = strlen(raw);
        strncpy(result, raw + 1, len - 2);
        result[len - 2] = '\0';
}

//Function to count the number of data values and their sum for each geo and year
//values / num = average
void
 getAverage(average arrAverage[5][8], int x, int ref_date, double value){
        arrAverage[x][0].num++;
        arrAverage[x][0].values += value;

        if(ref_date == 2015){
                arrAverage[x][1].num++;
                arrAverage[x][1].values += value;
        }
        else if(ref_date == 2016){
                arrAverage[x][2].num++;
                arrAverage[x][2].values += value;
        }
        else if(ref_date == 2017){
                arrAverage[x][3].num++;
                arrAverage[x][3].values += value;
        }
        else if(ref_date == 2018){
                arrAverage[x][4].num++;
                arrAverage[x][4].values += value;
        }
        else if(ref_date == 2019){
                arrAverage[x][5].num++;
                arrAverage[x][5].values += value;
        }
        else if(ref_date == 2020){
                arrAverage[x][6].num++;
                arrAverage[x][6].values += value;
        }
        else if(ref_date == 2021){
                arrAverage[x][7].num++;
                arrAverage[x][7].values += value;
        }
}
```

```
//Sorts the provincial average array to obtain the highest and lowest averages
void
sortAvg (double provAvg[]) {
  double temp;

  for(int j = 0; j < 3; j++){
          for(int i = 0; i < 3; i++){
                        if(provAvg[i+1] < provAvg[i]){
                                temp = provAvg[i];
                                provAvg[i] = provAvg[i+1];
                                provAvg[i+1] = temp;
                        }
                }
        }
}

//Determines the lowest and highest average for each year and province
void
sortArrAvg(average arrAverage[5][8], char provH[20], int *yearH, char provL[20], int *yearL){

        double highestAvg = arrAverage[1][1].avg;
        double lowestAvg = arrAverage[1][1].avg;

        for(int i = 1; i < 5; i++){
                for(int j = 1; j < 8; j++){

                        //Highest Average
                        if(arrAverage[i][j].avg > highestAvg){
                                highestAvg = arrAverage[i][j].avg;

                                if(i == 1)
                                        strncpy (provH, "- Quebec", 20);
                                else if(i == 2)
                                        strncpy (provH, "- Ontario", 20);
                                else if(i == 3)
                                        strncpy (provH, "- Alerta", 20);
                                else if(i == 4)
                                        strncpy (provH, "- British Columbia", 20);
		if(j == 1)
			*yearH = 2015;
		else if(j == 2)
			*yearH = 2016;
		else if(j == 3)
			*yearH = 2017;
		else if(j == 4)
			*yearH = 2018;
		else if(j == 5)
			*yearH = 2019;
		else if(j == 6)
			*yearH = 2020;
		else if(j == 7)
			*yearH = 2021;
	}

	//Lowest average
	if(arrAverage[i][j].avg < lowestAvg){
		lowestAvg = arrAverage[i][j].avg;

		if(i == 1)
			strncpy (provL, "- Quebec", 20);
		else if(i == 2)
			strncpy (provL, "- Ontario", 20);
		else if(i == 3)
			strncpy (provL, "- Alerta", 20);
		else if(i == 4)
			strncpy (provL, "- British Columbia", 20);

		if(j == 1)
			*yearL = 2015;
		else if(j == 2)
			*yearL = 2016;
		else if(j == 3)
			*yearL = 2017;
		else if(j == 4)
			*yearL = 2018;
		else if(j == 5)
			*yearL = 2019;
		else if(j == 6)
```

```
                                        *yearL = 2020;
                                else if(j == 7)
                                        *yearL = 2021;

                        }
                }
        }
}

int main(){
        FILE* file; //opens file
        file = fopen ("statscan_diabetes.csv", "r");//open file;

        //if file doesnt open
        if (file == NULL){
                perror("Error");
                return 1;
        }

        //Defining all variables
        question1 arr[MAX_RECORDS];
        int records = 0;
        int initalize = 0;
        int ref_date_num = 0;
        int geo_num = 1;
        int age_num = 3;
        int sex_num = 4;
        int value_num = 13;
        char line[4096];
        char provH[20];
        char provL[20];
        int yearH;
        int yearL;

        //Creating an average struct for each average needed
        //Geos
        average ca;
        average on;
        average qu;
        average bc;
        average al;

        //Age groups
        average age1; //35 to 49 years
        average age2; //50 to 64 years
        average age3; //65 years and over

        //Years
        average total;
        average y15;
        average y16;
        average y17;
        average y18;
        average y19;
        average y20;
        average y21;

        //makes a 2D array of structs
  for averages of all years for each province
        average arrAverage[5][8] = {{ca, qu, on, al, bc},{total, y15, y16, y17, y18, y19, y20,
y21}};

        //Loop to obtain data from file
        while (fgets(line, 4096, file)){

                //exit loop if file is read
                if(records == MAX_RECORDS)
                   break;

                // printf("%s\n", buf);
                if (!initalize){
                        initalize++;
                        records++;
                        continue;
                }

                question1 temp;

                // loop through all columns until equal to "," signifying end of column
                // value at current column put into item
		// item = "2015"
		char *token;
		token = strtok(line, ",");
		int col = 0;
		while (token != NULL)
		{// do whatever on the string
			char result[1000];
			removeQuotes(token, result);

			if (col == geo_num) //if in the geo column
				strcpy(temp.geo, result); //store in geo struct
			else if (col == ref_date_num) //if in the ref date column
				temp.ref_date = atoi(result); //store in the ref date struct
			else if (col == age_num)
				strcpy(temp.age_group, result);
			else if (col == sex_num)
				strcpy(temp.sex, result);
			else if (col == value_num)
				temp.value = atof(result);

			token = strtok(NULL, ",");
			col++;
			// points to start of next column
		}

		arr[records] = temp;
		records++;

		//Un-comment line below to print all raw data values from the file
		//printf("%d | %s | %s | %s | %.2lf \n", temp.ref_date, temp.age_group, temp.geo,
temp.sex, temp.value);

		//averages for country/province -- 1a, 1b, 1c
		//arrAverage[0][0] = Canada for all years
		//arrAverage[i][j]
		//i = each geo
		//j = each year
		if(strcmp( temp.geo, "Canada (excluding territories)") == 0)
			getAverage(arrAverage, 0, temp.ref_date, temp.value);
		//For Quebec
		else if(strcmp( temp.geo, "Quebec") == 0)
			getAverage(arrAverage, 1, temp.ref_date, temp.value);

		//For Ontario
		else if(strcmp( temp.geo, "Ontario") == 0)
			getAverage(arrAverage, 2, temp.ref_date, temp.value);

		//For Alberta
		else if(strcmp( temp.geo, "Alberta") == 0)
			getAverage(arrAverage, 3, temp.ref_date, temp.value);

		//For British Columbia
		else if(strcmp( temp.geo, "British Columbia") == 0)
			getAverage(arrAverage, 4, temp.ref_date, temp.value);


		//averages for age group -- 1d
		if(strcmp( temp.age_group, "35 to 49 years") == 0){
			age1.num ++;
			age1.values += temp.value;
		}
		else if(strcmp( temp.age_group, "50 to 64 years") == 0){
			age2.num ++;
			age2.values += temp.value;
		}
		else if(strcmp( temp.age_group, "65 years and over") == 0){
			age3.num ++;
			age3.values += temp.value;
		}

	}
	fclose(file);

	//calculating averages of each geo for all years
	ca.avg = arrAverage[0][0].values / arrAverage[0][0].num;
	qu.avg = arrAverage[1][0].values / arrAverage[1][0].num;
	on.avg = arrAverage[2][0].values / arrAverage[2][0].num;
	al.avg = arrAverage[3][0].values / arrAverage[3][0].num;
	bc.avg = arrAverage[4][0].values / arrAverage[4][0].num;
```

```
        //calculating averages of each age group
        age1.avg = age1.values / age1.num;
        age2.avg = age2.values / age2.num;
        age3.avg = age3.values / age3.num;

        //calculating averages of each geo for each year
        for(int i = 0; i < 5; i++){
                for(int j = 1; j < 8; j++){
                        arrAverage[i][j].avg = arrAverage[i][j].values / arrAverage[i][j].num;
                        //printf("%.2lf\n",arrAverage[i][j].avg);
                }
        }

//Outputting all averages for each question:
//Question 1
        //Printing averages
        printf("-All Values are Percentages-\n");
        printf("\nQUESTION 1\n\n");

        //Averages for each province
        printf("1a) Provincial Averages \n");
        printf("- Quebec: %.2lf\n", qu.avg);
        printf("- Ontario: %.2lf\n", on.avg);
        printf("- Alberta: %.2lf\n", al.avg);
        printf("- British Columbia: %.2lf\n", bc.avg);
        printf("\n");

        // Average for all of Canada
        printf("1b) National Average\n");
        printf("- Canada: %.2lf\n\n", ca.avg);


        //Averages of each province for every year

        //opens files for graph 1
        FILE* data1ca;
        data1ca = fopen("data1ca.txt", "w");

        FILE* data1qu;
```

```
data1qu = fopen("data1qu.txt", "w");

FILE* data1on;
data1on = fopen("data1on.txt", "w");

FILE* data1al;
data1al = fopen("data1al.txt", "w");

FILE* data1bc;
data1bc = fopen("data1bc.txt", "w");

printf("1c) Yearly Average for each Province");
char geo[20];
int year;
for(int i = 0; i < 5; i++){
        printf("\n");
        for(int j = 1; j < 8; j++){

                if(i == 0)
                        strcpy(geo, "Canada");
                else if(i == 1)
                        strcpy(geo, "Quebec");
                else if(i == 2)
                        strcpy(geo, "Ontario");
                else if(i == 3)
                        strcpy(geo, "Alberta");
                else if(i == 4)
                        strcpy(geo, "British Columbia");

                if(j == 1)
                        year = 2015;
                else if(j== 2)
                        year = 2016;
                else if(j== 3)
                        year = 2017;
                else if(j== 4)
                        year = 2018;
                else if(j== 5)
                        year = 2019;
                else if(j== 6)
```

```
                        year = 2020;
                else if(j== 7)
                        year = 2021;

                printf("- %s, %i: %.2lf\n", geo, year, arrAverage[i][j].avg);

                //Adds the average per year for each graph
                if(i == 0)
                        fprintf(data1ca, "%s %i %.2lf\n", geo, year, arrAverage[i][j].avg);
                else if(i == 1)
                        fprintf(data1qu, "%s %i %.2lf\n", geo, year, arrAverage[i][j].avg);
                else if(i == 2)
                        fprintf(data1on, "%s %i %.2lf\n", geo, year, arrAverage[i][j].avg);
                else if(i == 3)
                        fprintf(data1al, "%s %i %.2lf\n", geo, year, arrAverage[i][j].avg);
                else if(i == 4)
                        fprintf(data1bc, "%s %i %.2lf\n", geo, year, arrAverage[i][j].avg);

        }
    }
    printf("\n");

    //average for each age group
    FILE* data2;
    data2 = fopen("data2.txt", "w");

    printf("1d) Average per Age Group\n");
    printf("- 35-49: %.2lf\n", age1.avg);
    printf("- 60-64: %.2lf\n", age2.avg);
    printf("- 65+: %.2lf\n", age3.avg);

    fprintf(data2, "0 35-49 %.2lf\n", age1.avg);
    fprintf(data2, "1 60-64 %.2lf\n", age2.avg);
    fprintf(data2, "2 65+ %.2lf\n", age3.avg);

// end of Question 1

//Question 2
    printf("\nQUESTION 2\n");
    double provAvg[4] = {qu.avg, on.avg, al.avg, bc.avg};
```

```
	sortAvg(provAvg);

	//Prov with lowest percent
	printf("\nProvince with lowest percentage of diabetes: \n");
	if(provAvg[0] == qu.avg)
		printf("- Quebec\n");

	else if(provAvg[0] == on.avg)
		printf("- Ontario\n");

	else if(provAvg[0] == al.avg)
		printf("- Alberta\n");

	else if(provAvg[0] == bc.avg)
		printf("- British Columbia\n");


	//Prov with highest percent
	printf("\nProvince with highest percentage of diabetes: \n");
	if(provAvg[3] == qu.avg)
		printf("- Quebec\n");

	else if(provAvg[3] == on.avg)
		printf("- Ontario\n");

	else if(provAvg[3] == al.avg)
		printf("- Alberta\n");

	else if(provAvg[3] == bc.avg)
		printf("- British Columbia\n");

// end of Question 2

//Question 3
	printf("\nQUESTION 3\n");

	printf("\nProvinces above national average:\n");
	if(qu.avg > ca.avg)
		printf("- Quebec\n");
	if(on.avg > ca.avg)
```

```
        printf("- Ontario\n");
    if(al.avg > ca.avg)
        printf("- Alberta\n");
    if(bc.avg > ca.avg)
        printf("- British Columbia\n");

    printf("\nProvinces below national average:\n");
    if(qu.avg < ca.avg)
        printf("- Quebec\n");
    if(on.avg < ca.avg)
        printf("- Ontario\n");
    if(al.avg < ca.avg)
        printf("- Alberta\n");
    if(bc.avg < ca.avg)
        printf("- British Columbia\n");
// end of Question 3

//Question 4
    printf("\nQUESTION 4\n");
    sortArrAvg(arrAverage, provH, &yearH, provL, &yearL);

    printf("\nProvince and year with higest percent of diabetes:\n");
    printf("%s, %i\n", provH, yearH);

    printf("\nProvince and year with lowest percent of diabetes:\n");
    printf("%s, %i\n", provL, yearL);
// end of Question 4

    return (0);
}
```

GNUPLOT CODES

Graph 1-Using file "data1ca.txt", "data1qu.txt", "data1on.txt", "data1al.txt", "data1bc.txt"

```
set terminal svg enhanced size 600,480

label1 = "Canada Excluding Territories"
label2 = "Ontario"
label3 = "Quebec"
```

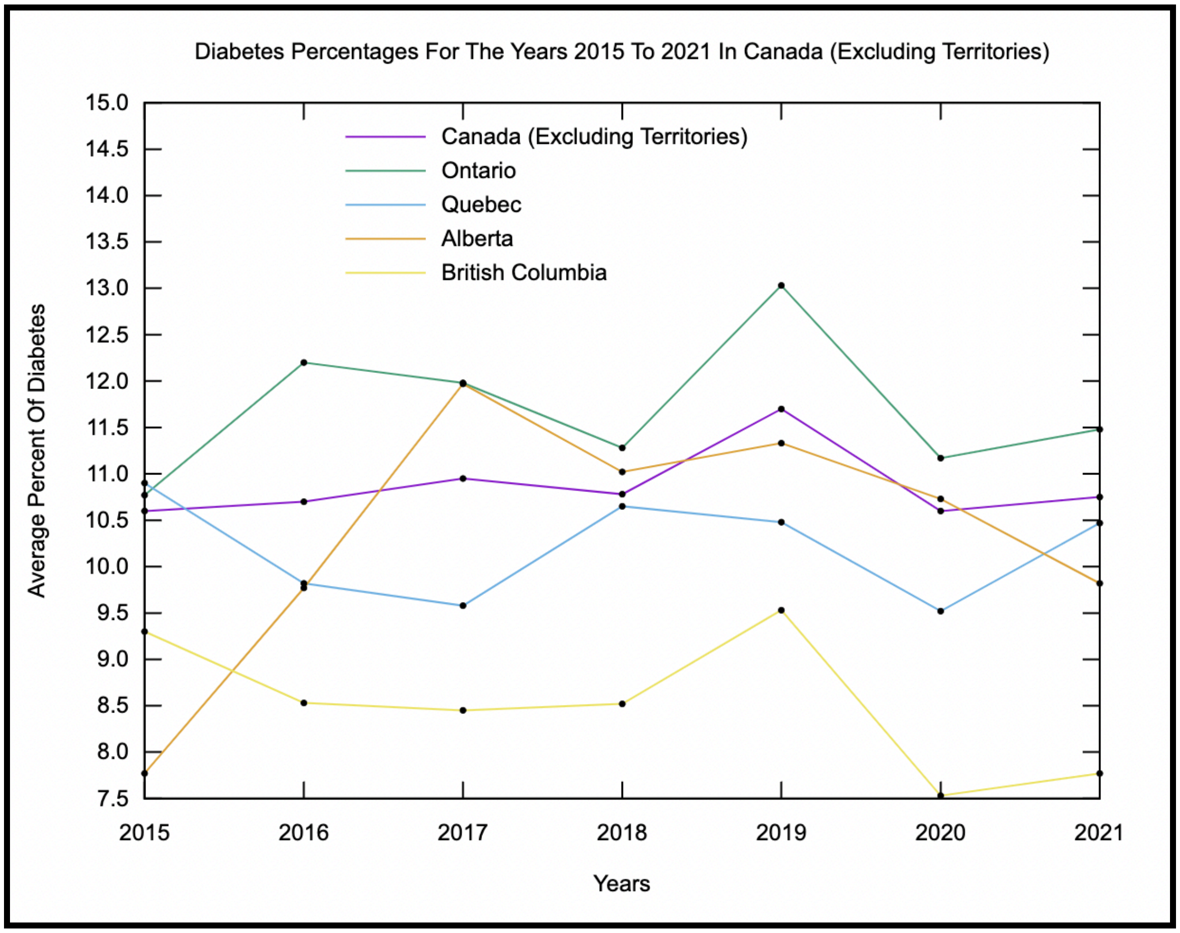
```
label4 = "Alberta"
label5 = "British Columbia"

set termoption enhanced
save_encoding = GPVAL_ENCODING
set encoding utf8

# the title
set title "Diabetes Percentages For The Years 2015 To 2021 In Canada (Excluding Territories)"

#centring the legend
set key Left center top reverse

#setting the labels for each side
set xlabel 'Years'
set ylabel 'Average Percent Of Diabetes'

#setting the range of each side and how much to go up by
set xrange [ 2015 : 2021 ]
set xtics 1
set ytics 0.5
set yrange [ 7.5 : 15 ]
set format y "%.1f"
set samples 500
set style fill solid 0.4 noborder

#Calling the information from the txt files to print
plot     "data1ca.txt" using 2:3 with lines title "Canada (Excluding Territories)", \
          "data1on.txt" using 2:3 with lines title "Ontario", \
          "data1qu.txt" using 2:3 with lines title "Quebec", \
          "data1al.txt" using 2:3 with lines title "Alberta", \
          "data1bc.txt" using 3:4 with lines title "British Columbia", \
          "data1ca.txt" using 2:3 with points pointtype 7 pointsize 0.35 lc rgb "black" notitle, \
          "data1on.txt" using 2:3 with points pointtype 7 pointsize 0.35 lc rgb "black" notitle, \
          "data1qu.txt" using 2:3 with points pointtype 7 pointsize 0.35 lc rgb "black" notitle, \
          "data1al.txt" using 2:3 with points pointtype 7 pointsize 0.35 lc rgb "black" notitle, \
          "data1bc.txt" using 3:4 with points pointtype 7 pointsize 0.35 lc rgb "black" notitle
```

Graph 2 - Using file "data2.txt"

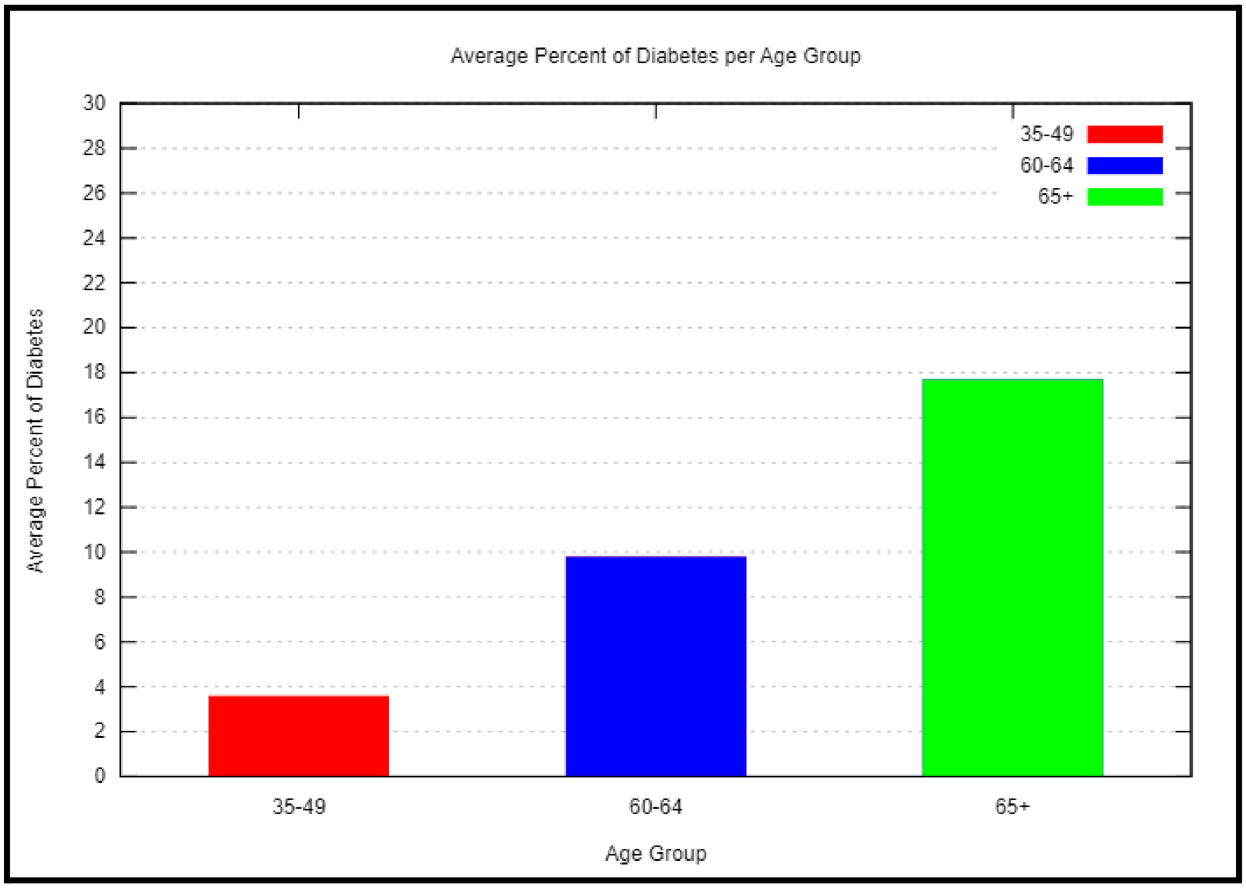
```
set terminal svg size 700,500

set title 'Average Percent of Diabetes per Age Group'
set xlabel 'Age Group'
set ylabel 'Average Percent of Diabetes'
set boxwidth 0.5
set style fill solid
set grid ytics
set yrange [0:30]

set style line 1 lc rgb "red"
set style line 2 lc rgb "blue"
set style line 3 lc rgb "green"

set ytics 2

plot "data2.txt" every ::0::1 using 1:3:xtic(2) with boxes ls 1 title "35-49", \
     "data2.txt" every ::1::2 using 1:3:xtic(2) with boxes ls 2 title "60-64", \
     "data2.txt" every ::2::3 using 1:3:xtic(2) with boxes ls 3 title "65+"
```

GNUPLOT GRAPHS

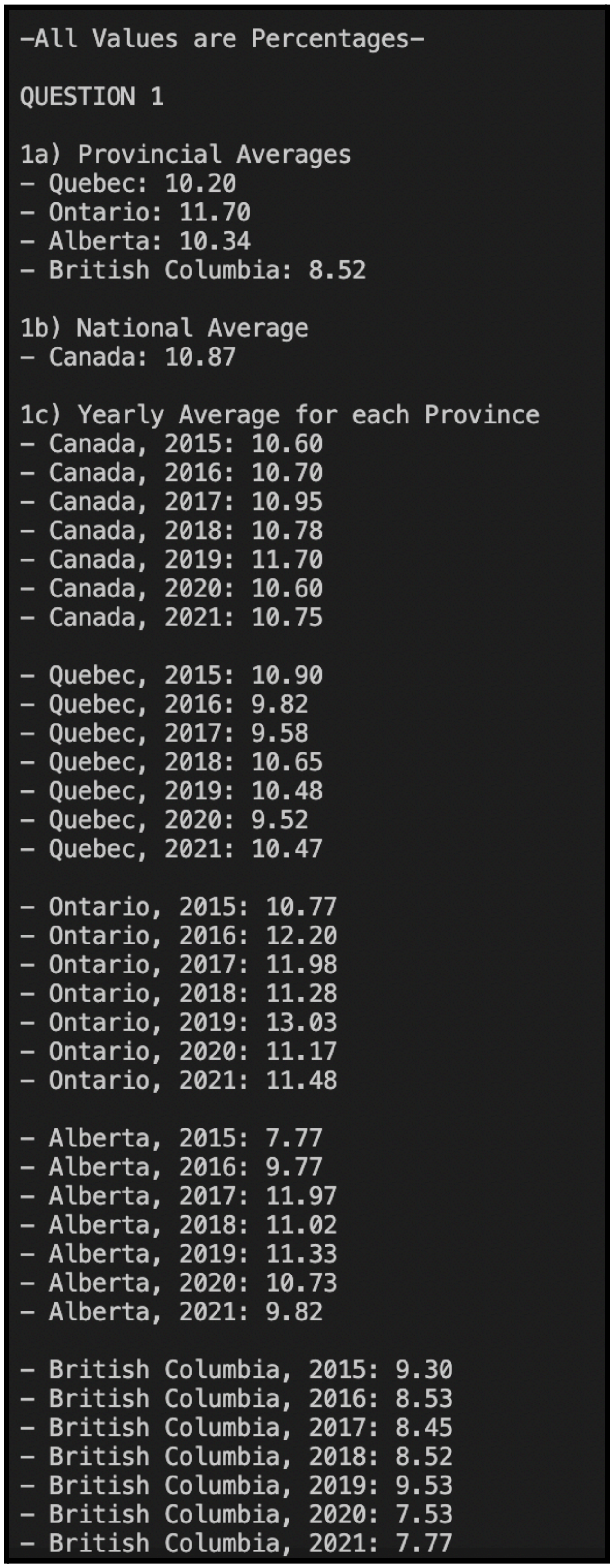
Graph 1:



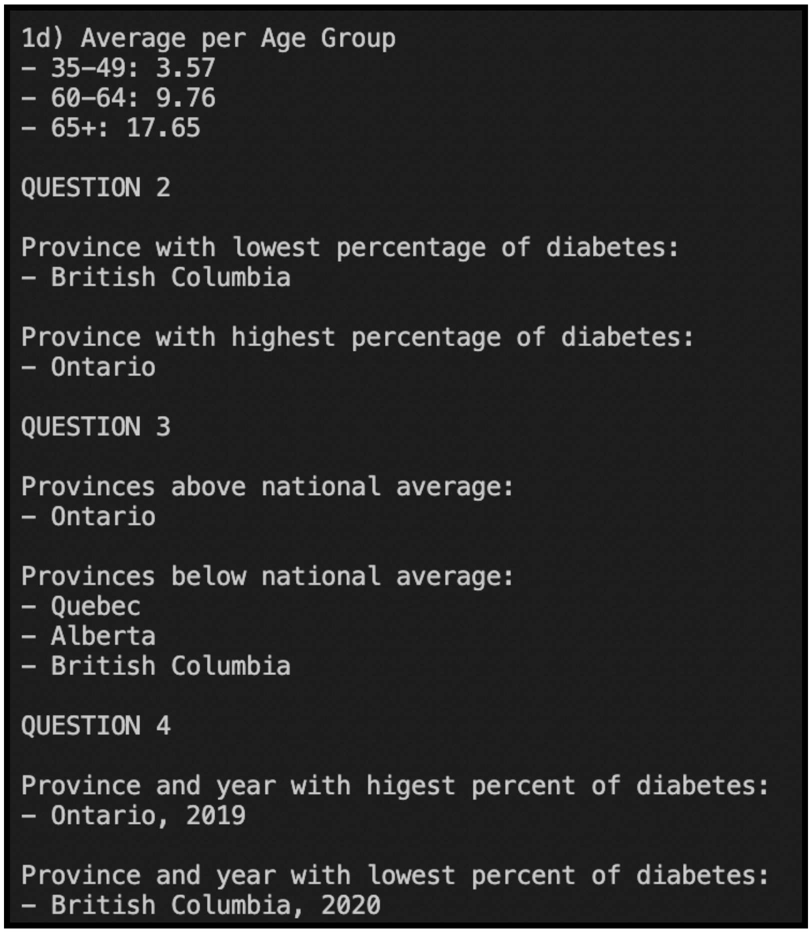
Graph 2:


Average Percent of Diabetes per Age Group
Average Percent of Diabetes
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
35-49
60-64
65+
Age Group
35-49 🟥
60-64 🟦
65+ 🟩

MAIN CODE OUTPUT (ANSWERS TO QUESTIONS)



```
-All Values are Percentages-

QUESTION 1

1a) Provincial Averages
- Quebec: 10.20
- Ontario: 11.70
- Alberta: 10.34
- British Columbia: 8.52

1b) National Average
- Canada: 10.87

1c) Yearly Average for each Province
- Canada, 2015: 10.60
- Canada, 2016: 10.70
- Canada, 2017: 10.95
- Canada, 2018: 10.78
- Canada, 2019: 11.70
- Canada, 2020: 10.60
- Canada, 2021: 10.75

- Quebec, 2015: 10.90
- Quebec, 2016: 9.82
- Quebec, 2017: 9.58
- Quebec, 2018: 10.65
- Quebec, 2019: 10.48
- Quebec, 2020: 9.52
- Quebec, 2021: 10.47

- Ontario, 2015: 10.77
- Ontario, 2016: 12.20
- Ontario, 2017: 11.98
- Ontario, 2018: 11.28
- Ontario, 2019: 13.03
- Ontario, 2020: 11.17
- Ontario, 2021: 11.48

- Alberta, 2015: 7.77
- Alberta, 2016: 9.77
- Alberta, 2017: 11.97
- Alberta, 2018: 11.02
- Alberta, 2019: 11.33
- Alberta, 2020: 10.73
- Alberta, 2021: 9.82

- British Columbia, 2015: 9.30
- British Columbia, 2016: 8.53
- British Columbia, 2017: 8.45
- British Columbia, 2018: 8.52
- British Columbia, 2019: 9.53
- British Columbia, 2020: 7.53
- British Columbia, 2021: 7.77
```



```
1d) Average per Age Group
- 35-49: 3.57
- 60-64: 9.76
- 65+: 17.65

QUESTION 2

Province with lowest percentage of diabetes:
- British Columbia

Province with highest percentage of diabetes:
- Ontario

QUESTION 3

Provinces above national average:
- Ontario

Provinces below national average:
- Quebec
- Alberta
- British Columbia

QUESTION 4

Province and year with higest percent of diabetes:
- Ontario, 2019

Province and year with lowest percent of diabetes:
- British Columbia, 2020
```