Term Project

Subhan Sajadi, Aananth Kuganendra, Joshua Savunthararajah, Sami Matar

CPS188

Dr. Denis Hamelin

April 2, 2023

Introduction:

This term project's objective is to use a C program to calculate and draw conclusions regarding the prevalence of diabetes in Canada's four most populous provinces (Ontario, Quebec, British Columbia, and Alberta) based on actual data gathered by Statistics Canada. Since it is more effective, C code can be used to manage the information collected by Statistics Canada and compute different statistical measures like mean, median, and standard deviation. Moreover, graphs displaying patterns in the incidence of diabetes in the chosen provinces can be made using this data.

By running a C code analysis on the data, we can identify the prevalence of diabetes in each of these provinces as well as potential risk factors and patterns associated with the disease. The public health policies and efforts that aim to control and prevent diabetes as well as provide support and resources to individuals who currently have the condition can be guided by these data.

Question 1: *Calculations*

The tasks in question 1 required the calculation of averages using C programming operations. All of the calculations were performed by functions that were declared before the main function. These functions were then called upon within the main function. The main function also outputs the average to the user.

The average is simply the sum of inputted values divided by the number of inputted values. In some cases, there are less inputted values than expected due to blank cells being present in the data. The blank cells are a result of missing data and they can impact the average if they are taken into account. The program in this report assigns a value of 0 to each blank.

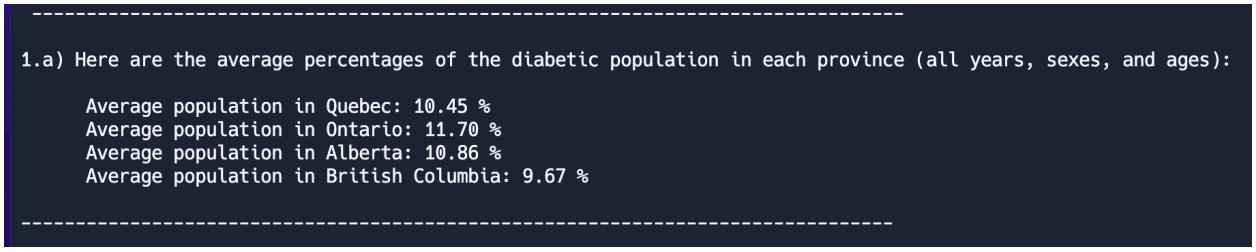
Part a: Average Per Province

Question 1a) asks for the average percentage of diabetics in each province. This includes data from all years, all age groups, both males and females in the four provinces: Quebec, Ontario, Alberta, British Columbia.

Table 1: Pre-calculated values for Part a

Province	Average of Population with Dlabetes
Quebec	10.45121951
Ontario	11.70238095
Alberta	10.86
British Columbia	9.67027027

Figure 1: Output from Part a



The output from **Image 1** matches the precalculated values in **Table 1**. The averages were determined using the function, `provavg`. This function took an array with all the population values, the number of rows in the csv file, and the specified province. The function consists of a for loop which runs until it goes through all of the population values. During this process, if the location matches with the area of the population value, and the population is non-zero (this ignores any blanks already set to 0), the population value is added to the double variable, **`total`**, and integer variable, **`num_entries`**, increases. After the iteration is finished, the program returns the result of **`total/num_entries`**. This is the average for the location. Multiple averages were outputted using a loop in the main function.

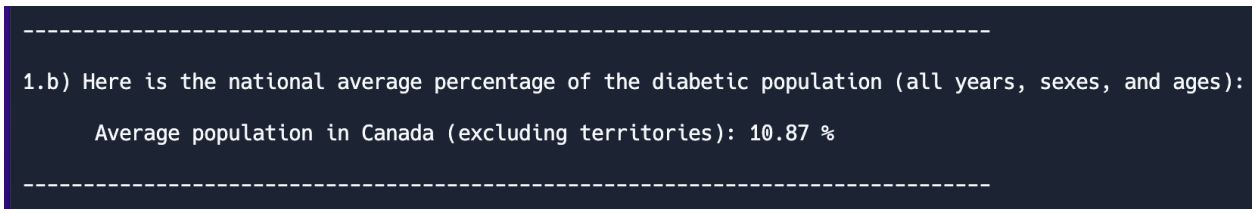
Part b: National Average

Question 1b) asks for the exact same information as question 1a) except for the national average.

Table 2: Pre-calculated values for Part b

Area	Average of Population with Dlabetes
Canada	10.86905

Figure 2: Output from Part b



The output from **Figure 2** matches the precalculated value in **Table 2**. The exact same approach in question 1a) was used here. The averages were also determined using the function, `provavg`. Instead of subbing in the provinces as the location, Canada was subbed in. Every population value associated with the location, “Canada (excluding all territories)”, in the csv file was taken and divided by the number of values – in this case 7.

Part c: Yearly Average per Area

Question 1c) asks for the yearly average diabetic percentage for each province. This includes all the population values for all three age groups, for both male and females, over seven years (2015-2021). A total of 35 averages will have to be calculated. This particular problem requires a slightly different approach than what was used in questions 1a) and 1b). The slight difference is in the function used to find the average.

Table 3: Precalculated values for Part c

Year	Area	Average of Population with Dlabetes
2015	Quebec	10.9
	Ontario	10.77
	Alberta	9.32
	BC	9.3
	Canada	10.6
2016	Quebec	9.816666667
	Ontario	12.2
	Alberta	9.766666667
	BC	8.533333333
	Canada	10.7
2017	Quebec	9.583333333
	Ontario	11.98333333
	Alberta	11.96666667
	BC	10.14
	Canada	10.95
2018	Quebec	10.65
	Ontario	11.28333333
	Alberta	11.01666667
	BC	8.516666667
	Canada	10.78333333
2019	Quebec	10.48333333
	Ontario	13.03333333
	Alberta	11.33333333
	BC	11.44
	Canada	11.7

2020	Quebec	11.42
	Ontario	11.16666667
	Alberta	12.88
	BC	9.04
	Canada	10.6
2021	Quebec	10.46666667
	Ontario	11.48333333
	Alberta	9.816666667
	BC	11.65
	Canada	10.75

Figure 3: Output from Part c

```
--------------------------------------------------------------------------------

1.c) Here are the yearly average percentages of the diabetic population in each province (all sexes and ages):

      In the year 2015
       Average population in Canada (excluding territories): 10.60 %
       Average population in Quebec: 10.90 %
       Average population in Ontario: 10.77 %
       Average population in Alberta: 9.32 %
       Average population in British Columbia: 9.30 %

      In the year 2016
       Average population in Canada (excluding territories): 10.70 %
       Average population in Quebec: 9.82 %
       Average population in Ontario: 12.20 %
       Average population in Alberta: 9.77 %
       Average population in British Columbia: 8.53 %

      In the year 2017
       Average population in Canada (excluding territories): 10.95 %
       Average population in Quebec: 9.58 %
       Average population in Ontario: 11.98 %
       Average population in Alberta: 11.97 %
       Average population in British Columbia: 10.14 %

      In the year 2018
       Average population in Canada (excluding territories): 10.78 %
       Average population in Quebec: 10.65 %
       Average population in Ontario: 11.28 %
       Average population in Alberta: 11.02 %
       Average population in British Columbia: 8.52 %

      In the year 2019
       Average population in Canada (excluding territories): 11.70 %
       Average population in Quebec: 10.48 %
       Average population in Ontario: 13.03 %
       Average population in Alberta: 11.33 %
       Average population in British Columbia: 11.44 %
```

```
In the year 2020
 Average population in Canada (excluding territories): 10.60 %
 Average population in Quebec: 11.42 %
 Average population in Ontario: 11.17 %
 Average population in Alberta: 12.88 %
 Average population in British Columbia: 9.04 %

In the year 2021
 Average population in Canada (excluding territories): 10.75 %
 Average population in Quebec: 10.47 %
 Average population in Ontario: 11.48 %
 Average population in Alberta: 9.82 %
 Average population in British Columbia: 11.65 %

--------------------------------------------------------------------------------
```

The output from **Figure 3** matches the precalculated value in **Table 3**. The averages were determined using the function, `provavg_peryear`. Similar to the function used in 1a) and 1b), the program returns the result of **`total/num_entries`**. However this function took the year as an additional variable along with all the values used in `provavg`. The function still consists of a for loop which runs until it goes through all of the population values. During this process, if the location matches with the area of the population value, and the year matches the year with the population value, and the population is non-zero, the population value is added to the double variable, **`total`**, and integer variable, **`num_entries`**, increases. Multiple averages were outputted using a loop in the main function.

Part d: Average of Each Age Group per Area

Question 1d), similarly to 1c) asks for the averages with an extra condition. This time, the population averages of each age group for each province is required. This includes the population averages of all three age groups for all years, both male and female, in each of the five specified areas. This totals to 15 averages.

Table 4: Precalculated values from Part d

Age Group	Area	Average of Population with Dlabetes
35-49	Quebec	3.353846154
	Ontario	4.642857143
	Alberta	4.458333333
	BC	3.433333333
	Canada	4.064285714

49-65	Quebec	9.057142857
	Ontario	11.22142857
	Alberta	10.28571429
	BC	7.914285714
	Canada	10.32857143
65+	Quebec	18.43571429
	Ontario	19.24285714
	Alberta	16.92142857
	BC	15.43571429
	Canada	18.21428571

Figure 4: Output from Part d

```
--------------------------------------------------------------------------------

1.d) Here is are the average percentages of the diabetic population of each age group in each province (all years and sexes):

      For people aged 35 to 49 years
       Average population in Canada (excluding territories): 4.06 %
       Average population in Quebec: 3.35 %
       Average population in Ontario: 4.64 %
       Average population in Alberta: 4.46 %
       Average population in British Columbia: 3.43 %

      For people aged 50 to 64 years
       Average population in Canada (excluding territories): 10.33 %
       Average population in Quebec: 9.06 %
       Average population in Ontario: 11.22 %
       Average population in Alberta: 10.29 %
       Average population in British Columbia: 7.91 %

      For people aged 65 years and over
       Average population in Canada (excluding territories): 18.21 %
       Average population in Quebec: 18.44 %
       Average population in Ontario: 19.24 %
       Average population in Alberta: 16.92 %
       Average population in British Columbia: 15.44 %

--------------------------------------------------------------------------------
```

The output from **Figure 4** matches the precalculated values in **Table 4**. The averages were determined using the function, `provavg_perage`. Similar to the function used in 1c), the program returns the result of **`total/num_entries`**. However this function took the age as an additional variable along with all the values used in `provavg` and `proavg_year`. The function still consists of a for loop which runs until it goes through all of the population values. During this process, if the location matches with the area of the population value, and the age matches the age group with the population value, and the population is non-zero, the population value is added to the double variable, **`total`,** and integer variable, **`num_entries`**, increases. Multiple averages were outputted using a loop in the main function.

Question 2: *Highest and Lowest Percentage of Diabetes (Provincial)*

This question was done using a nested for loop. Inside the nested loop, the current province average is compared to the next province. If it is greater, it is then stored in a temporary variable. The purpose of the nested for loop is to sort the provinces in ascending order so they can be the highest and lowest average can be printed out.

Figure 5: Output from Question 2

```
--------------------------------------------------------------------------------

2. The province with the lowest diabetic percentage average for all years and age groups is British Columbia with 9.67 %

   The province with the highest diabetic percentage average for all years and age groups is Ontario with 11.70 %

--------------------------------------------------------------------------------
```

Question 3: *Percentages Below and Above National Average*

This question was done using two for loops to iterate through every province. The first loop checks if the province average is greater than the national average, and if it is, the province and its average will be printed out. The province and national average was stored in an array which is seen in question 1a. The other for loop works similarly by checking if the province average is less than the average.

Figure 6: Output from Question 3

```
--------------------------------------------------------------------------------

3. The following provinces have average percentages above the national diabetic percentage, 10.87 %:
   Ontario: 11.70 %

   The following provinces have average percentages below the national diabetic percentage, 10.87 %:
   British Columbia: 9.67 %
   Quebec: 10.45 %
   Alberta: 10.86 %

--------------------------------------------------------------------------------
```

Question 4: *Highest and Lowest Percentage of Diabetes*

This question was done using nested for loops and if statements. First, a couple of variables were defined. The permax (highest percentage of diabetes) and permin (lowest percentage of diabetes) were set equal to -DBL_MAX and DBL_MAX. These predefined constants are an exclusive to the <float.h> library and serve the purpose of defining the largest negative value and largest positive value. In the nested for loop, the “provavg_peryear” is called to iterate through each year and province. In the if statements, the program will check is the current province and year will check and keep replacing the max and min averages and store them in maxprov or minprov.

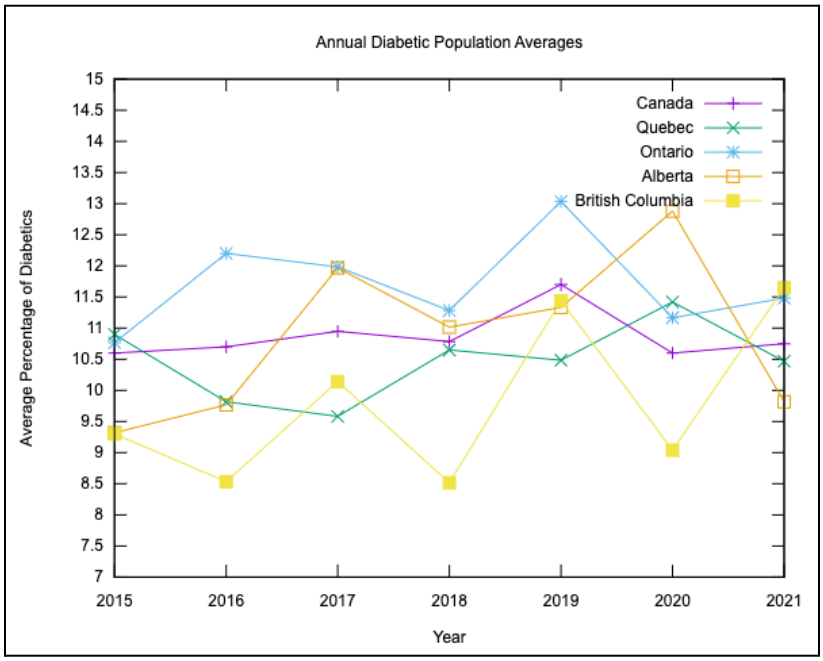
Figure 7: Output from Question 4

```
--------------------------------------------------------------------------------

4. Year and province with the highest and lowest percentage of diabetes:

  The highest percentage of diabetes was in Ontario in the year 2019 with 13.03 %

  The lowest percentage of diabetes was in British Columbia in the year 2018 with 8.52 %

------------------------------------------------------
```

Question 5: *Graphing Yearly Averages per Area*

Question 5 asks for the averages in Question 1a) to be represented in a line graph. The text file used in GNUplot for this graph was generated by the C program. The files **`can1`**, **`qu1`**, **`on1`**, **`al1`**, and **`bc1`** were all written to have the year and yearly average diabetic population for the corresponding province. 5 files were made to reduce any complications with GNUplot. These values were obtained by printing them into each file in the same for loop that printed the output for question 1a).

Figure 8: Graph of the data from Part 1a)

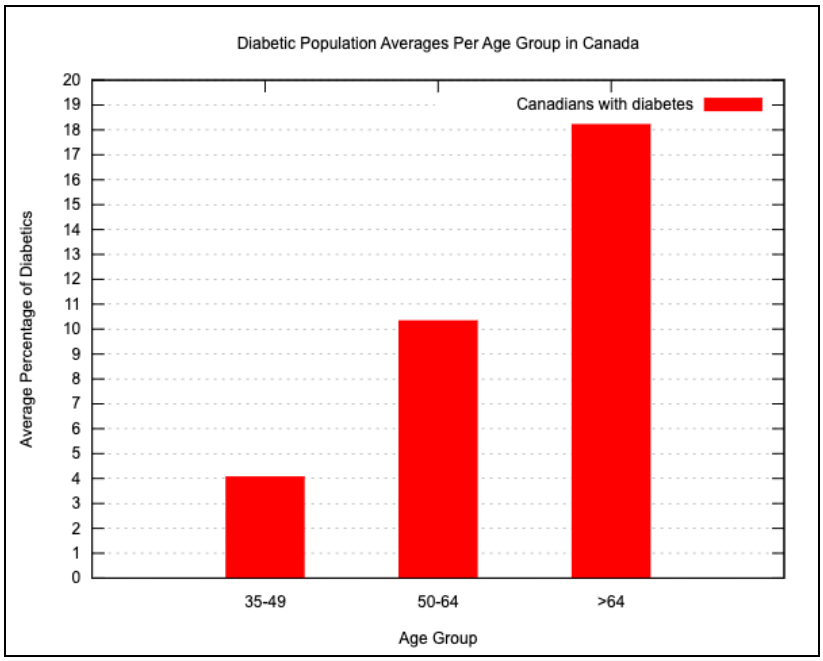


This question was done using various functions. The function ‘set title’ displays the main title of the line graph. The functions ‘set xlabel’ and ‘set ylabel’ display what values the x-axis and y-axis are representing. The functions ‘set xtics’ and ‘set ytics’ are the increments that the values on each axis go by. The function plot ‘ ’ reads the data of the file inputted which contains the values for both the x axis and y axis, and displays those values on the line graph.

Question 6: *Graphing Average of Each Age Group per Area*

Question 6 asks for the averages for Canada in Question 1d) to be represented in a bar graph. The text file used in GNUplot for this graph was generated by the C program. The file **`can2`** was written to have the age group and average diabetic population of each age group in Canada. These values were obtained by printing them into **`out2`** in the same for loop that printed the output for question 1d).

Figure 9: Graph of the data from Part 1d)



Similarly to Question 5, this question used various GNUPlot functions including ‘set title’, ‘set xlabel’, ‘set ylabel’, ‘set xtics’, and ‘set ytics’. However this question required a bar graph instead, which is done with the function ‘set style data histogram’, setting the data in a bar format. The function ‘set yrange[]’ established the range of numbers that were displayed on the y-axis. The function ‘set boxwidth’ determined the width of each bar on the graph. The function ‘set grid’ showed the graph on a grid. The plot function read the data of the given file and displayed it on the bar graph. The function ‘line color’ displays the bars with the color inputted.

Conclusion:

To sum everything up, it should be noted that the incidence of diabetes in Canada is a serious public health issue, particularly in the four most populous provinces. Hence, the information gathered by Statistics Canada can be used to inform public health policies. Such as activities aimed at reducing the prevalence of diabetes in Canada. When doing this project, it was seen

that diabetes is more common in older age groups; thus, these results can guide targeted programmes that aim to lower diabetes risk factors and enhance healthcare access for those who need it most.

When doing this project, it was determined that coding can be an extremely efficient and effective tool to perform calculations and other representations on the small and large scale. Over the course of this project, a better understanding of the c programming language was developed. If this project were to be done again, a better approach would be to do some research on other programming methods, known and unknown, to make the final program as efficient and neat as possible.

Appendix:

Complete C Program:
*Code can also be accessed with this link:
https://replit.com/@AananthKuganen1/cps188projectw23#main.c

```
/*
CPS188 Winter 2023 Term Project
Toronto Metropolitan University
Prof: Dr. Hamelin
Names: Subhan Sajadi, Aananth Kuganendra, Joshua Savunthararajah,  Sami
Matar
Date Created: Sunday, March 26, 2023
Last Modified: Sunday, April 7, 2023
*/

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

#define maxrows 500
#define maxlen 1000

// Defining a structure to store each relevant column of data into an array
//Name of column on data file is commented beside each struct array
typedef struct {
    int year;//REF_DATE
    char* area;//GEO
    char* age_group;//Age Group
    char* sex;//Sex
    double perpopulation;//VALUE
} Column;

//Function to provide the average for question 1a and 1b
double provavg(Column* column_info, int rows, const char* location) {
    double total = 0.0;
    int num_entries = 0;

    for (Column* curr_row = column_info; curr_row < column_info + (rows-2);
curr_row++) {
        if (strcmp(curr_row->area, location) == 0 &&
curr_row->perpopulation != 0) {
```

```
            total += curr_row->perpopulation;
            num_entries++;
        }
    }

    if (num_entries > 0) {
         double AVG = total / num_entries;
        return AVG;
    } else {
        return 0;
    }
}

//Function to provide averages for question 1c
double provavg_peryear(Column* column_info, Column* column_data, int rows,
const char* location, int yr) {
    double total = 0.0;
    int num_entries = 0;
    Column* curr_col;

    for (curr_col = column_info; curr_col < column_info + rows - 2;
curr_col++) {
        if (strcmp(curr_col->area, location) == 0 && curr_col->year - yr ==
0 && curr_col->perpopulation != 0) {
            total += curr_col->perpopulation;
            num_entries++;
        }
    }

    if (num_entries > 0) {
         double AVG = total / num_entries;
        return AVG;
    }
    else {
        return 0;
    }
}

//Function to provide averages for question 1d
double provavg_perage(Column* column_info, Column* column_data, int rows,
const char* location, const char* target_age) {
    double total = 0.0;
    int num_entries = 0;
```

```
    for (Column* info_ptr = column_info; info_ptr < column_info + rows - 2;
info_ptr++, column_data++) {
        if (strcmp(info_ptr->area, location) == 0 &&
strcmp(column_data->age_group, target_age) == 0 && info_ptr->perpopulation
!= 0) {
            total += info_ptr->perpopulation;
            num_entries++;
        }
    }

    if (num_entries > 0) {
         double AVG = total / num_entries;
        return AVG;
    }
    else {
        return 0;
    }
}

//Function to eliminate quotation marks that are present in strings
char* clean(char* str) {
    int len = strlen(str);
    if (len < 2 || str[0] != '"' || str[len-1] != '"') {
        return strdup(str);
    }
      char* new_string = (char*)malloc(len - 1);
      strncpy(new_string, str + 1, len - 2);
      new_string[len - 2] = '\0';

  return new_string;
}

// Function to read the statscan_diabetes file in CSV and store the
relevant columns of data into the C program
int readfile(const char* nof, Column* column_info) {
    FILE* fp = fopen(nof, "r");
    if (fp == NULL) {
        perror("\n\nfile error\n\n");
        return -1;
    }

    char line[maxlen];
```

```
    int row = 0;
    while (fgets(line, maxlen, fp)) {
        // This will skip the title of each row
        if (row == 0) {
            row++;
            continue;
        }

        // Cuts the line
char* token = strtok(line, ",");
Column curr_row = {0};

for (int col = 0; token != NULL; col++, token = strtok(NULL, ",")) {
    switch(col) {
        case 0: curr_row.year = atoi(clean(token)); break;
        case 1: curr_row.area = clean(strdup(token)); break;
        case 3: curr_row.age_group = clean(strdup(token)); break;
        case 4: curr_row.sex = clean(strdup(token)); break;
        case 13: curr_row.perpopulation = atof(clean(token)); break;
        default: break;
    }
}

column_info[row-1] = curr_row;
row++;
    }
    fclose(fp);
    return row-1;
}

//Questions will be answered and answers will be outputted in the main
function
int main() {
    Column column_info[maxrows];
    int rows = readfile("statscan_diabetes.csv", column_info);
    //These are data files created by the C program to be used later for
plotting graphs on GNUplot

      //Files for question 5
      FILE* out = fopen ("can1", "w");
      FILE* out2 = fopen ("qu1", "w");
      FILE* out3 = fopen ("on1", "w");
      FILE* out4 = fopen ("al1", "w");
```

```
      FILE* out5 = fopen ("bc1", "w");

      //File for question 6
      FILE* outb = fopen ("can2", "w");

  printf("Welcome! \nThis C program was designed to extract information
from a data file, statscan_diabetes.csv, which contained information on the
prevalence of diabetes in Ontario, Quebec, Alberta, British Columbia and
Canada. The program outputs percentage averages in varying conditions,
providing a better insight into the nation's struggle with diabetes. \n\n
");

  //Question 1a
    const char* areas[] = {"Quebec", "Ontario", "Alberta", "British
Columbia"};
    double average1[4];


printf("----------------------------------------------------------------------
-------------");
    printf("\n\n");
    printf("1.a) Here are the average percentages of the diabetic
population in each province (all years, sexes, and ages): \n\n");
    for (int i = 0; i < sizeof(areas) / sizeof(areas[0]); i++) {
        average1[i] = provavg(column_info, rows, areas[i]);
        printf("      Average population in %s: %.2lf %%\n", areas[i],
average1[i]);
    }
    printf("\n");

printf("----------------------------------------------------------------------
-------------");
    printf("\n");

  //Question 1b

    printf("\n1.b) Here is the national average percentage of the diabetic
population (all years, sexes, and ages): \n\n");
        double averagen = provavg(column_info, rows, "Canada (excluding
territories)");
        printf("      Average population in Canada (excluding territories):
%.2lf %%\n", averagen);
```

```
    printf("\n");

printf("------------------------------------------------------------------------
-------------");
    printf("\n");

  //Question 1c
    //Printing headers for files used in question 5
    fprintf (out, "# year area avg \n");
    fprintf (out2, "# year area avg \n");
    fprintf (out3, "# year area avg \n");
    fprintf (out4, "# year area avg \n");
    fprintf (out5, "# year area avg \n");

    int yr[] = {2015, 2016, 2017, 2018, 2019, 2020, 2021};
    printf("\n1.c) Here are the yearly average percentages of the diabetic
population in each province (all sexes and ages): \n\n");
    const char* areas2[] = {"Canada (excluding territories)", "Quebec",
"Ontario", "Alberta", "British Columbia"};
    for (int j = 0; j < sizeof(yr) / sizeof(yr[0]); j++){
        printf("      In the year %d\n", yr[j]);
        for (int i = 0; i < sizeof(areas2) / sizeof(areas2[0]); i++) {
            double average = provavg_peryear(column_info, column_info,
rows, areas2[i], yr[j]);
            printf("       Average population in %s: %.2lf %%\n",
areas2[i], average);
            if (strcmp(areas2[i], "Canada (excluding territories)") == 0){
                fprintf (out, "%-15d %-15s %-15lf\n", yr[j], areas2[i],
average);
            }
            if (strcmp(areas2[i], "Quebec") == 0){
                fprintf (out2, "%-15d %-15s %-15lf\n", yr[j], areas2[i],
average);
            }
            if (strcmp(areas2[i], "Ontario") == 0){
                fprintf (out3, "%-15d %-15s %-15lf\n", yr[j], areas2[i],
average);
            }
            if (strcmp(areas2[i], "Alberta") == 0){
                fprintf (out4, "%-15d %-15s %-15lf\n", yr[j], areas2[i],
average);
            }
            if (strcmp(areas2[i], "British Columbia") == 0){
```

```
                fprintf (out5, "%-15d %-15s %-15lf\n", yr[j], areas2[i],
average);
            }
        }

        printf("\n");
    }

printf("------------------------------------------------------------------
-------------");
    printf("\n");

  //Question 1d
    fprintf (outb, "# age area avg \n");
    const char* ages[] = {"35 to 49 years", "50 to 64 years", "65 years and
over"};
    printf("\n1.d) Here is are the average percentages of the diabetic
population of each age group in each province (all years and sexes):
\n\n");
    for (int j = 0; j < sizeof(ages) / sizeof(ages[0]); j++){
        printf("      For people aged %s\n", ages[j]);
        for (int i = 0; i < sizeof(areas2) / sizeof(areas2[0]); i++) {
            double average = provavg_perage(column_info, column_info, rows,
areas2[i], ages[j]);
            printf("       Average population in %s: %.2lf %%\n",
areas2[i], average);
            if (strcmp(areas2[i], "Canada (excluding territories)") == 0){
                if (strcmp(ages[j], "35 to 49 years") == 0){
                    fprintf (outb, "%-15d 35-49 %-15lf\n", j, average);
                }
                if (strcmp(ages[j], "50 to 64 years") == 0){
                    fprintf (outb, "%-15d 50-64 %-15lf\n", j, average);
                }
                if (strcmp(ages[j], "65 years and over") == 0){
                    fprintf (outb, "%-15d >64 %-15lf\n", j, average);
                }

            }
        }
    printf("\n");
    }

printf("------------------------------------------------------------------
```

```
-------------");
    printf("\n");

  //Question 2
    for(int j = 0; j < sizeof(areas) / sizeof(areas[0]); j++){
        for (int k = 0; k < sizeof(areas) / sizeof(areas[0]); k++){
            double temp;
            const char* temp2;
            if (average1[k] > average1[k+1]){
                temp = average1[k];
                temp2 = areas[k];
                average1[k] = average1[k+1];
                areas[k] = areas[k+1];
                average1[k+1] = temp;
                areas[k+1] = temp2;
            }

        }
    }
    printf("\n2. The province with the lowest diabetic percentage average
for all years and age groups is %s with %.2lf %%\n\n",         areas[0],
average1[0]);
    printf("   The province with the highest diabetic percentage average
for all years and age groups is %s with %.2lf %%\n", areas[3],
average1[3]);

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

printf("-------------------------------------------------------------------
-------------");
    printf("\n");
    printf("\n3. The following provinces have average percentages above the
national diabetic percentage, %.2lf %%: \n", averagen);
    for(int i = 0; i < sizeof(areas) / sizeof(areas[0]); i++){
        if (average1[i] > averagen){
            printf("   %s: %.2lf %%\n", areas[i], average1[i]);
        }
    }
    printf("\n   The following provinces have average percentages below the
national diabetic percentage, %.2lf %%: \n", averagen);
    for(int i = 0; i < sizeof(areas) / sizeof(areas[0]); i++){
        if (average1[i] < averagen){
```

```
            printf("   %s: %.2lf %%\n", areas[i], average1[i]);
        }
    }
//Question 4
  printf("\n");

printf("------------------------------------------------------------------
-------------");
  printf("\n");
  printf("\n4. Year and province with the highest and lowest percentage of
diabetes:\n\n");

  // Variables to store highest and lowest percentages and their
corresponding years and provinces
  double permax = -DBL_MAX;
  double permin = DBL_MAX;
  int maxyr = -1;
  int minyr = -1;
  const char* maxprov;
  const char* minprov;
  int i,j;

  for (j = 0; j < sizeof(yr) / sizeof(yr[0]); j++) {
      for (i = 0; i < sizeof(areas) / sizeof(areas[0]); i++) {
          double AVG = provavg_peryear(column_info, column_info, rows,
areas[i], yr[j]);

          if (AVG > permax) {
              permax = AVG;
              maxyr = yr[j];
              maxprov = areas[i];
          }

          if (AVG < permin) {
              permin = AVG;
              minyr = yr[j];
              minprov = areas[i];
          }
    }
  }
  printf("  The highest percentage of diabetes was in %s in the year %d
with %.2lf %%\n\n", maxprov, maxyr, permax);
  printf("  The lowest percentage of diabetes was in %s in the year %d with
```

```
%.2lf %%\n", minprov, minyr, permin);
  printf("\n");

printf("-------------------------------------------------------------------
------------");
  return 0;

}
```

GNUplot Question 5

```
set title 'Annual Diabetic Population Averages'
set xlabel 'Year'
set ylabel 'Average Percentage of Diabetics'
set xtics 1
set ytics 0.5
set yrange[7:15]
plot 'can1.txt' using 1:5 title "Canada" with lp, \
     'qu1.txt' using 1:3 title "Quebec" with lp, \
     'on1.txt' using 1:3 title "Ontario" with lp, \
     'al1.txt' using 1:3 title "Alberta" with lp, \
     'bc1.txt' using 1:4 title "British Columbia" with lp
```

GNUplot Question 6

```
set style data histogram
set style histogram cluster gap 1

set yrange[0:20]
set style fill solid
set boxwidth 0.9
set xtics format ""
set grid ytics

set title 'Diabetic Population Averages Per Age Group in Canada'
set xlabel 'Age Group'
set ylabel 'Average Percentage of Diabetics'
set ytics 1
plot 'can2.txt' using 3:xtic(2) title "Canadians with diabetes"
linecolor "red" \
```