CPS 188 Term Project
Winter 2023

Instructor: *Dr. Ufkes*

TA: *Mohammed Emrul Hasan*

Sayeed Ahamad, Qurrat-Ul-Ain, Nourhan Antar, Tre Spencer

Student Number: *501209136, 501169908, 501196794, 501087581*

1 Introduction

This report examines actual data gathered by Statistics Canada on the prevalence of diabetes in the four most populous provinces of Canada (Ontario, Quebec, British Columbia, and Alberta) and the as well as national data, from 2015 to 2021. The report gives a summary of a C programming project that includes computations, the creation of graphs, and conclusions based on the gathered data.

The project requires the use of the C programming language in order to take data from a CSV file, do computations, and provide the required output, including tables and graphs. The data file includes information on the prevalence of diabetes among people aged 35 and older in each of the four provinces as well as across the entire country (excluding territories).

In-depth discussion of the project's essential elements is provided in the report, including computation of annual averages, identification of the provinces with the highest and lowest percentages of diabetics, and computation of the provincial and national averages of the population with diabetes diagnoses. The report also highlights the necessity to identify the provinces above and below the national average as well as the years with the highest and lowest percentages of diabetes.

The project also requires the development of two graphs: a line plot showing diabetes percentages from 2015 to 2021 and a bar graph showing the average percentages of diabetes among the three age groups for the entire country. The study emphasizes the significance of clearly labeling the axes as well as presenting each graph with a title and a legend.

The overall objective of the project is to use C programming and GNUPlot features to investigate the prevalence of diabetes in Canada's four most populous provinces and draw conclusions using data collected by Statistics Canada.

2 Problem Sets

2.1 Problem 1

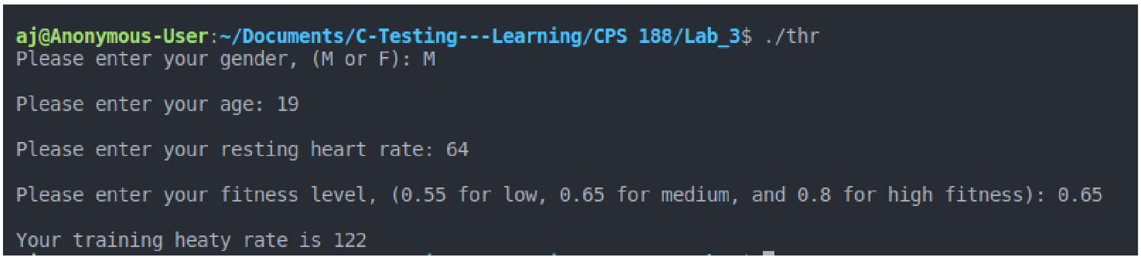
2.1.1 Computer Program

```c
/*  Program to calculate the Training Heart Rate (THR)   */

#include <stdio.h>
#include <math.h>
#include <stdbool.h>

float inputs(void);
bool gender_conditional(char gender);
int male_training_heart_rate(int age, int resting_heart_rate, float fitness_level);
int female_training_heart_rate(int age, int resting_heart_rate, float fitness_level);
int conditional(char gender, int age, int resting_heart_rate, float fitness_level);
void output(int training_heart_rate);

void main(void)
{
    float g, a, rhr, fl = inputs();
    int thr = conditional(g, a, rhr, fl);
    output(thr);
}

float inputs(void)
{
    char gender;
    int age;
    int resting_heart_rate;
    float fitness_level;

    /*  Scanning values for gender selection    */
    printf("Please enter your gender, (M or F): ");
    do
    {
        scanf("%c", &gender);
    } while (gender == 'M' || gender == 'F');


    /*  Scanning values for the age */
    printf("\nPlease enter your age: ");
    scanf("%i", &age);

    /*  Scanning values for the resting heart rate  */
    printf("\nPlease enter your resting heart rate: ");
    scanf("%i", &resting_heart_rate);

    /*  Scanning values for fitness level   */
    printf("\nPlease enter your fitness level, (0.55 for low, 0.65 for medium, and 0.8 for high fitness): ");
    scanf("%f", &fitness_level);

    return gender, age, resting_heart_rate, fitness_level;
}

int conditional(char gender, int age, int resting_heart_rate,
    float fitness_level)
{
    /*  Conditional to check male or female */
    bool binary = gender_conditional(gender);

    /*  Conditional for check male or female THR     */
    int training_heart_rate;
    if (binary == true)
    {
        training_heart_rate = male_training_heart_rate(age,
   resting_heart_rate, fitness_level);
    }

    else
    {
        training_heart_rate = female_training_heart_rate(age,
    resting_heart_rate, fitness_level);
    }

    return training_heart_rate;
}

void output(int training_heart_rate)
{
    printf("\nYour training heaty rate is %i\n",
   training_heart_rate);
}

bool gender_conditional(char gender)
{
    int binary;
    if (gender == 'M')
    {
        binary = true;
    }

    else
    {
        binary = false;
    }

    return binary;
}
```

```c

int male_training_heart_rate(int age, int resting_heart_rate, float fitness_level)
{
    /*  Calculating the maximum heart rate  */
    float maximum_heart_rate = 203.7 / (1 + exp(0.033 * (age - 104.3)));

    /*  Calculating the training heart rate */
    int training_heart_rate = (maximum_heart_rate - resting_heart_rate) * fitness_level + resting_heart_rate;

    return training_heart_rate;
}

int female_training_heart_rate(int age, int resting_heart_rate, float fitness_level)
{
    /*  Calculating the maximum heart rate  */
    int maximum_heart_rate = 190.2 / (1 + exp(0.0453 * (age - 107.5)));

    /*  Calculating the training heart rate */
    int training_heart_rate = (maximum_heart_rate - resting_heart_rate) * fitness_level + resting_heart_rate;

    return training_heart_rate;
}
```

Listing 2.1: *Hello World Program*

2.1.2 Program Output Screenshot



```
aj@Anonymous-User:~/Documents/C-Testing---Learning/CPS 188/Lab_3$ ./thr
Please enter your gender, (M or F): M

Please enter your age: 19

Please enter your resting heart rate: 64

Please enter your fitness level, (0.55 for low, 0.65 for medium, and 0.8 for high fitness): 0.65

Your training heaty rate is 122
```

2.2 Problem 2

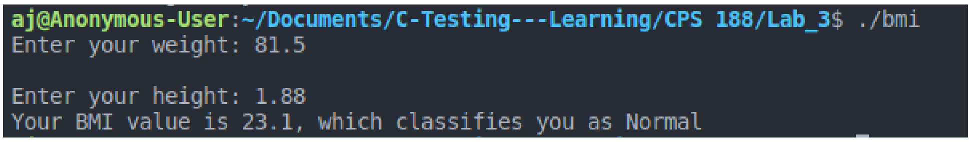
2.2.1 Computer Program

```c
#include <stdio.h>
#include <math.h>
#include <stdbool.h>
```

```c

float weight_input(void);
float height_input(void);
void output(float weight, float height);

void main(void)
{
    float w = weight_input();
    float h = height_input();
    output(w, h);
}

float weight_input(void)
{
    float weight;

    /*  Scanning values for weight  */
    printf("Enter your weight: ");
    scanf("%f", &weight);

    return weight;
}

float height_input(void)
{
    float height;

    /*  Scanning values for height  */
    printf("\nEnter your height: ");
    scanf("%f", &height);

    return height;
}

void output(float weight, float height)
{
    /*  Calculating BMI */
    height *= height;
    float body_mass_index = weight / (height);

    /*  Conditional */
    if (body_mass_index < 18.5)
    {
        printf("Your BMI value is %.1f, which classifies you as Underweight\n", body_mass_index);
    }
    else if (body_mass_index <= 24.9)
    {
        printf("Your BMI value is %.1f, which classifies you as Normal\n", body_mass_index);
    }
    else if (body_mass_index <= 29.9)
    {
        printf("Your BMI value is %.1f, which classifies you as Overweight\n", body_mass_index);
    }
    else
    {
        printf("Your BMI value is %.1f, which classifies you as Obese\n", body_mass_index);
    }
}

```

Listing 2.2: *Program to Calculate the Body Mass Index (BMI) of a person*

2.2.2 Program Output Screenshot



```
aj@Anonymous-User:~/Documents/C-Testing---Learning/CPS 188/Lab_3$ ./bmi
Enter your weight: 81.5

Enter your height: 1.88
Your BMI value is 23.1, which classifies you as Normal
```

2.3 Problem 3

2.3.1 Computer Program

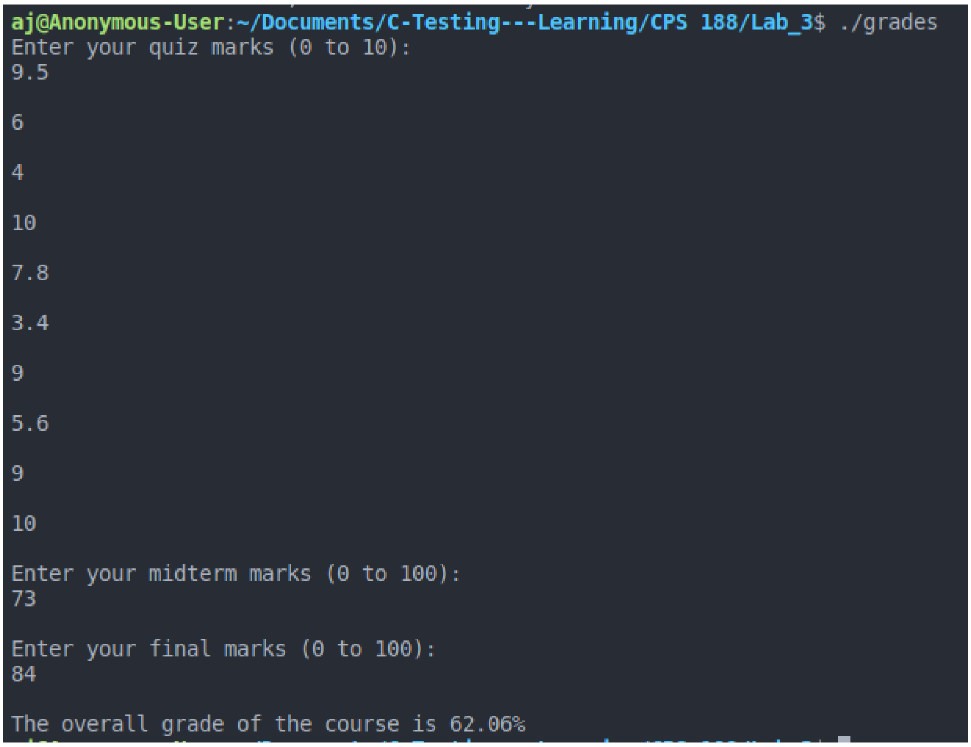
```c
/*  Program to Calculate the Overall grades of a Course */

#include <stdio.h>
#include <math.h>

float quiz(void);
float midterm(void);
float final(void);
float conditional_output(float quiz, float midterm, float
    final);

void main(void)
{
    float q = quiz();
    float m = midterm();
    float f = final();
    conditional_output(q, m, f);
}

```

```
float quiz(void)
{
    float quiz[10];
    float lowest;
    float sum = 0;

    printf("Enter your quiz marks (0 to 10):\n");
    for (int i = 0; i < 10; i++)
    {
        do
        {
            scanf("%f", &quiz[i]);
            printf("\n");
        } while (quiz[i] < 0 || quiz[i] > 10);
    }

    for (int i = 0; i < 10; i++)
    {
        if (quiz[i] < quiz[i+1])
        {
            lowest = quiz[i];
        }
    }

    for (int i = 0; i < 10; i++)
    {
        sum += quiz[i];
    }

    float average = (sum - lowest) / 9;

    return average;
}

float midterm(void)
{
    float marks;

    printf("Enter your midterm marks (0 to 100):\n");
    do
    {
        scanf("%f", &marks);
        printf("\n");
    } while (marks < 0 || marks > 100);

    return marks;
}

float final(void)
```

```c
{
    float marks;

    printf("Enter your final marks (0 to 100):\n");
    do
    {
        scanf("%f", &marks);
        printf("\n");
    } while (marks < 0 || marks > 100);

    return marks;
}

float conditional_output(float quiz, float midterm, float final)
{
    quiz *= 0.25;

    if (midterm >= final)
    {
        midterm *= 0.35;
        final *= 0.4;
    }
    else
    {
        midterm *= 0.25;
        final *= 0.5;
    }

    float grade = quiz + midterm + final;

    printf("The overall grade of the course is %.2f\n", grade);
}


```

Listing 2.3: *Program to Calculate the Overall grades of a Course*

2.3.2 Program Output Screenshot



```
aj@Anonymous-User:~/Documents/C-Testing---Learning/CPS 188/Lab_3$ ./grades
Enter your quiz marks (0 to 10):
9.5

6

4

10

7.8

3.4

9

5.6

9

10

Enter your midterm marks (0 to 100):
73

Enter your final marks (0 to 100):
84

The overall grade of the course is 62.06%
```

Appendices

A C Source Codes

A.1 QQuestions 1, 2, 3 4 Source Code

```c


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

/*  Defining program macros */

#define ARRAY_SIZE 500
#define LINE_SIZE 250
#define STRING_SIZE 50
#define SPACERS 3
#define SUB_SPACERS 2

/*  Initializing UDF's in program use   */

void credits(void);
FILE* file_o(void);
void file_c(FILE* file);
void spacer(void);
void sub_spacer(void);
void avg_province(void);
void avg_year(void);
void year_avg(double f_2015, double f_2016, double f_2017,
    double f_2018, double f_2019, double f_2020, double f_2021
    , double q_2015, double q_2016, double q_2017, double
    q_2018, double q_2019, double q_2020, double q_2021,
    double o_2015, double o_2016, double o_2017, double o_2018
    , double o_2019, double o_2020, double o_2021, double
    a_2015, double a_2016, double a_2017, double a_2018,
    double a_2019, double a_2020, double a_2021, double b_2015
    , double b_2016, double b_2017, double b_2018, double
    b_2019, double b_2020, double b_2021);
void avg_age(void);
void age_avg(double f_35, double f_50, double f_65, double
    q_35, double q_50, double q_65, double o_35, double o_50,
    double o_65, double a_35, double a_50, double a_65, double
     b_35, double b_50, double b_65);

/*  Initializing struct datatypes for CSV data  */
```

```

typedef struct {    /*  Struct to store every Parameter in a line as an array of tokens with respect to their induvidual fields */
  char year[10];
  char province[35];
  char age_group[20];
  char sex[10];
  char values[10];
  char temp_str[10];
} datatypes;    /*  Struct DataType Variable name defined as "datatypes"    */

void main(void)
{
  credits();

    FILE* f = file_o(); /*  Initializing File Operations */

    datatypes data_set[ARRAY_SIZE];
    char line[LINE_SIZE];

    int line_count = 0; /*  Initializing Line Counter Variable to count Lines in the File   */

    while (!feof(f))    /*  Initializing start of CORE program base function    */
    {
        if (line_count == 0)
        {
            fgets(line, LINE_SIZE, f);
            line_count++;
            continue;   /*  Parsing the first line as line 1 encompasses labels and headers which are of no relavance */
        }
        fgets(line, LINE_SIZE, f);
        line_count++;   /*  Line Counter Variable Update */

        int token_count = 0;    /*  Initializing Token Counter Variable to count the tokens after String Tokenization   */
        char* token = strtok(line, ",");    /*  Seperating the line string into subsequent smaller string based on Comma Seperation & Tokenizing a slice of string after "," delimiter as a parameter   */
```

```
     strcpy(data_set[line_count].year, token);   /* Assigning a string value for the Year from this base iteration of Var(token) [NO CONDITION REQUIRED - FIRST FIED ENTRY IN FILE] */
     //token_count++;  /*  Token Counter Variable Update */

     while(token != NULL)
     {
         token = strtok(NULL, ",");
         token_count++;  /*  Token Counter Variable Update */

         /*if (token_count == 1)
         {
             strcpy(data_set[line_count].province, token);   //  Assigning a string value for Province from this iteration of Var(token) if conditional satisfied
         }*/
         if (token_count == 3)
         {
             strcpy(data_set[line_count].age_group, token);  /*  Assiging a string value of Age Goup from this iteration of Var(token) if conditional is satisfied   */
         }
         else if (token_count == 4)
         {
             strcpy(data_set[line_count].sex, token);  /*  Assigning a string value of Sex from this iteration of Var(token) if conditional is satisfied   */
         }
         else if (token_count == 13)
         {
             strcpy(data_set[line_count].values, token);  /*  Assigning a string value of Values (Raw Percentage Floats) from this iterationm of Var(token) if condition is satisfied */
         }
         else if (token_count == 14)
         {
             strcpy(data_set[line_count].temp_str, token);   /*  Assigning a string temp trash value of string literal's after Var(values) to disregard "\"" delimiter from being concatenated into Values string [CONDITION IS ALWAYS SATISFIED - TOTAL TOKEN COUNT IS 19 - temp_str HAS FREE malloc(20) per cycle allocation]  */
         }
     }
 }   /*  End of CORE program base function   */
```

```

  file_c(f);  /*  Terminating File Operations */

printf("Note: All Mathematical values and operations are
 signified and represented as follows in accordance to the
 percent operater parameter. Please refer to Project
 documentation for further information.\n");

  /*          Question 1          */printf("
 |----------------------------------------------------------------------------
 Question
 1----------------------------------------------------------------------------
 n");

spacer();
/*    Province Wise Averages    */avg_province();
spacer();

  /*  Initializing Sum & Iterating Counter Variables for
 Province Averages Calculation  */

  //  float sum_province - Variable to store the sum of the
  Value Data Points subsequently in every iteration
  //  province_iterator_counter - Variable to calculate the
  number of iterations performed in the for loop

  /*  Federal Variables   */

  float sum_federal = 0;
  int federal_iterator_counter = 0;

  /*  Quebec Variables    */

  float sum_quebec = 0;
  int quebec_iterator_counter = 0;

  /*  Ontario Variables   */

  float sum_ontario = 0;
  int ontario_iterator_counter = 0;

  /*  Alberta Variables   */

  float sum_alberta = 0;
  int alberta_iterator_counter = 0;

  /*  British Columbia Variables  */

  float sum_british_columbia = 0;
```

```c
    int british_columbia_iterator_counter = 0;

    /*  Federal Average Calculator  */

    for (int i = 2; i < 44; i++, federal_iterator_counter++)
    {
      char add_val_f[10];   /*  Initializing char variable to copy char pointer to char variable type   */
      char stg_val_f[10];   /*  Initializing char variable to store char to char recieved from char pointer */
      strcpy(add_val_f, data_set[i].values);    /*  Copying to char variable from char pointer variable */
      strcpy(stg_val_f, &add_val_f[1]); /*  Storing the char variable in another char variable to manipulate string literals    */
      double values_federal = atof(stg_val_f);  /*  Converting the stored char variable to a float type varaible data type for mathematical computational maniupulation */
      if (values_federal == 0)
      {
        federal_iterator_counter--; /*  Fail-Safe Mechanism for not counting the iterations in the iterating counter factor if the condition is met [CONDITION IS ONLY SATISFIED IF THE atof FUNCTION RETURN 0, IFF THE Var(char) = NULL]   */
      }
      //printf("%.2lf\n",values_federal);   /*  Fail-Safe in Testing phase to verify succinctity of the values being read from atof function    */
      sum_federal += values_federal;    /*  Calculation of Summa function of all data points being read that are not NULL   */
    } double avg_federal = (sum_federal) / (federal_iterator_counter); printf("Federal Average: %.3lf\n", avg_federal); /*  Calculation of the Average function from the previous Summa function and iterator counter function as inputs    */

    /*  Quebec Average Calculator   */

    for (int i = 44; i < 86; i++, quebec_iterator_counter++)
    {
      char add_val_q[10];   /*  Initializing char variable to copy char pointer to char variable type   */
      char stg_val_q[10];   /*  Initializing char variable to store char to char recieved from char pointer */
      strcpy(add_val_q, data_set[i].values);    /*  Copying to char variable from char pointer variable */
```

```c
  strcpy(stg_val_q, &add_val_q[1]); /*  Storing the char variable in another char variable to manipulate string literals    */
  double values_quebec = atof(stg_val_q);  /*  Converting the stored char variable to a float type varaible data type for mathematical computational maniupulation */
  if (values_quebec == 0)
  {
   quebec_iterator_counter--; /*  Fail-Safe Mechanism for not counting the iterations in the iterating counter factor if the condition is met [CONDITION IS ONLY SATISFIED IF THE atof FUNCTION RETURN 0, IFF THE Var(char) = NULL]   */
  }
  //printf("%.2lf\n",values_quebec);   /*  Fail-Safe in Testing phase to verify succinctity of the values being read from atof function    */
  sum_quebec += values_quebec;    /*  Calculation of Summa function of all data points being read that are not NULL   */
 } double avg_quebec = (sum_quebec) / (quebec_iterator_counter); printf("Quebec Average: %.3lf\n", avg_quebec);

 /*  Ontario Average Calculator  */

 for (int i = 86; i < 128; i++, ontario_iterator_counter++)
 {
  char add_val_o[10];   /*  Initializing char variable to copy char pointer to char variable type   */
  char stg_val_o[10];   /*  Initializing char variable to store char to char recieved from char pointer */
  strcpy(add_val_o, data_set[i].values);    /*  Copying to char variable from char pointer variable */
  strcpy(stg_val_o, &add_val_o[1]); /*  Storing the char variable in another char variable to manipulate string literals    */
  double values_ontario = atof(stg_val_o);  /*  Converting the stored char variable to a float type varaible data type for mathematical computational maniupulation */
  if (sum_ontario == 0)
  {
   ontario_iterator_counter--; /*  Fail-Safe Mechanism for not counting the iterations in the iterating counter factor if the condition is met [CONDITION IS ONLY SATISFIED IF THE atof FUNCTION RETURN 0, IFF THE Var(char) = NULL]   */
  }
  //printf("%.2lf\n",values_ontario);   /*  Fail-Safe in Testing phase to verify succinctity of the values being read from atof function    */
  sum_ontario += values_ontario;    /*  Calculation of Summa function of all data points being read that are not NULL   */
 } double avg_ontario = (sum_ontario) / (ontario_iterator_counter); printf("Ontario Average: %.3lf\n", avg_ontario);

 /*  Alberta Average Calculator  */

 for (int i = 128; i < 170; i++, alberta_iterator_counter++)
 {
  char add_val_a[10];   /*  Initializing char variable to copy char pointer to char variable type   */
  char stg_val_a[10];   /*  Initializing char variable to store char to char recieved from char pointer */
  strcpy(add_val_a, data_set[i].values);    /*  Copying to char variable from char pointer variable */
  strcpy(stg_val_a, &add_val_a[1]); /*  Storing the char variable in another char variable to manipulate string literals    */
  double values_alberta = atof(stg_val_a);  /* Converting the stored char variable to a float type varaible data type for mathematical computational maniupulation */
  if (values_alberta == 0)
  {
   alberta_iterator_counter--; /*  Fail-Safe Mechanism for not counting the iterations in the iterating counter factor if the condition is met [CONDITION IS ONLY SATISFIED IF THE atof FUNCTION RETURN 0, IFF THE Var(char) = NULL]   */
  }
  //printf("%.2lf\n",values_alberta);   /*  Fail-Safe in Testing phase to verify succinctity of the values being read from atof function    */
  sum_alberta += values_alberta;    /*  Calculation of Summa function of all data points being read that are not NULL   */
 } double avg_alberta = (sum_alberta) / (alberta_iterator_counter); printf("Alberta Average: %.3lf\n", avg_alberta);

 /*  British Columbia Average Calculator */

```

```
    for (int i = 170; i < 212; i++, british_columbia_iterator_counter++)
    {
      char add_val_b[10];   /*  Initializing char variable to copy char pointer to char variable type   */
      char stg_val_b[10];   /*  Initializing char variable to store char to char recieved from char pointer */
      strcpy(add_val_b, data_set[i].values);    /*  Copying to char variable from char pointer variable */
      strcpy(stg_val_b, &add_val_b[1]); /*  Storing the char variable in another char variable to manipulate string literals    */
      double values_british_columbia = atof(stg_val_b);  /* Converting the stored char variable to a float type varaible data type for mathematical computational maniupulation */
      if (values_british_columbia == 0)
      {
        british_columbia_iterator_counter--; /*  Fail-Safe Mechanism for not counting the iterations in the iterating counter factor if the condition is met [CONDITION IS ONLY SATISFIED IF THE atof FUNCTION RETURN 0, IFF THE Var(char) = NULL]   */
      }
      //printf("%.2lf\n",values_british_columbia);   /*  Fail-Safe in Testing phase to verify succinctity of the values being read from atof function    */
      sum_british_columbia += values_british_columbia;    /* Calculation of Summa function of all data points being read that are not NULL   */
    } double avg_british_columbia = (sum_british_columbia) / (british_columbia_iterator_counter); printf("British Columbia Average: %.3lf\n", avg_british_columbia);

  spacer();

    /*  Year-Wise Average Calculator*/

  /*    Year Wise Averages    */avg_year();
  spacer();

  /*  Initializing Sum & Iterating Counter Variables for Year Averages Calculation   */

    //  double sum_year - Variable to store the sum of the Value Data Points subsequently in every iteration
  //  double avg_year - Variable to store the average of the Value Data Points subsequently after all iterations
    //  iterator_counter_year - Variable to calculate the number of iterations performed in the for loop

  /*  2015 Variables  */

    double sum_2015 = 0;
  double avg_2015 = 0;
  double sum_2015_f = 0;
  double avg_2015_f = 0;
  double sum_2015_q = 0;
  double avg_2015_q = 0;
  double sum_2015_o = 0;
  double avg_2015_o = 0;
  double sum_2015_a = 0;
  double avg_2015_a = 0;
  double sum_2015_b = 0;
  double avg_2015_b = 0;
  int iterator_counter_2015 = 0;
  int iterator_counter_2015_f = 0;
  int iterator_counter_2015_q = 0;
  int iterator_counter_2015_o = 0;
  int iterator_counter_2015_a = 0;
  int iterator_counter_2015_b = 0;

  /*  2016 Variables  */

  double sum_2016 = 0;
  double avg_2016 = 0;
  double sum_2016_f = 0;
  double avg_2016_f = 0;
  double sum_2016_q = 0;
  double avg_2016_q = 0;
  double sum_2016_o = 0;
  double avg_2016_o = 0;
  double sum_2016_a = 0;
  double avg_2016_a = 0;
  double sum_2016_b = 0;
  double avg_2016_b = 0;
  int iterator_counter_2016 = 0;
  int iterator_counter_2016_f = 0;
  int iterator_counter_2016_q = 0;
  int iterator_counter_2016_o = 0;
  int iterator_counter_2016_a = 0;
  int iterator_counter_2016_b = 0;

  /*  2017 Variables  */

  double sum_2017 = 0;
  double avg_2017 = 0;
```

```
  double sum_2017_f = 0;
  double avg_2017_f = 0;
  double sum_2017_q = 0;
  double avg_2017_q = 0;
  double sum_2017_o = 0;
  double avg_2017_o = 0;
  double sum_2017_a = 0;
  double avg_2017_a = 0;
  double sum_2017_b = 0;
  double avg_2017_b = 0;
  int iterator_counter_2017 = 0;
  int iterator_counter_2017_f = 0;
  int iterator_counter_2017_q = 0;
  int iterator_counter_2017_o = 0;
  int iterator_counter_2017_a = 0;
  int iterator_counter_2017_b = 0;

  /* 2018 Variables */

  double sum_2018 = 0;
  double avg_2018 = 0;
  double sum_2018_f = 0;
  double avg_2018_f = 0;
  double sum_2018_q = 0;
  double avg_2018_q = 0;
  double sum_2018_o = 0;
  double avg_2018_o = 0;
  double sum_2018_a = 0;
  double avg_2018_a = 0;
  double sum_2018_b = 0;
  double avg_2018_b = 0;
  int iterator_counter_2018 = 0;
  int iterator_counter_2018_f = 0;
  int iterator_counter_2018_q = 0;
  int iterator_counter_2018_o = 0;
  int iterator_counter_2018_a = 0;
  int iterator_counter_2018_b = 0;

  /* 2019 Variables */

  double sum_2019 = 0;
  double avg_2019 = 0;
  double sum_2019_f = 0;
  double avg_2019_f = 0;
  double sum_2019_q = 0;
  double avg_2019_q = 0;
  double sum_2019_o = 0;
  double avg_2019_o = 0;
  double sum_2019_a = 0;
  double avg_2019_a = 0;
  double sum_2019_b = 0;
  double avg_2019_b = 0;
  int iterator_counter_2019 = 0;
  int iterator_counter_2019_f = 0;
  int iterator_counter_2019_q = 0;
  int iterator_counter_2019_o = 0;
  int iterator_counter_2019_a = 0;
  int iterator_counter_2019_b = 0;

  /*  2020 Variables  */

  double sum_2020 = 0;
  double avg_2020 = 0;
  double sum_2020_f = 0;
  double avg_2020_f = 0;
  double sum_2020_q = 0;
  double avg_2020_q = 0;
  double sum_2020_o = 0;
  double avg_2020_o = 0;
  double sum_2020_a = 0;
  double avg_2020_a = 0;
  double sum_2020_b = 0;
  double avg_2020_b = 0;
  int iterator_counter_2020 = 0;
  int iterator_counter_2020_f = 0;
  int iterator_counter_2020_q = 0;
  int iterator_counter_2020_o = 0;
  int iterator_counter_2020_a = 0;
  int iterator_counter_2020_b = 0;

  /*  2021 Variables  */

  double sum_2021 = 0;
  double avg_2021 = 0;
  double sum_2021_f = 0;
  double avg_2021_f = 0;
  double sum_2021_q = 0;
  double avg_2021_q = 0;
  double sum_2021_o = 0;
  double avg_2021_o = 0;
  double sum_2021_a = 0;
  double avg_2021_a = 0;
  double sum_2021_b = 0;
  double avg_2021_b = 0;
  int iterator_counter_2021 = 0;
  int iterator_counter_2021_f = 0;
  int iterator_counter_2021_q = 0;
  int iterator_counter_2021_o = 0;
  int iterator_counter_2021_a = 0;
  int iterator_counter_2021_b = 0;

    for (int i = 0; i < ARRAY_SIZE; i++)
    {
    for (int j = 2; j < 44; j++)  /*  Federal Year-Wise Average Calculator  */
    {
      char add_val_y[10];
      char stg_val_y[10];
      strcpy(add_val_y, data_set[j].year);
      strcpy(stg_val_y, &add_val_y[1]);
      double year = atof(stg_val_y);

      if (year == 2015)
      {
        iterator_counter_2015_f++;
        char add_val_2015[10];
        char stg_val_2015[10];
        strcpy(add_val_2015, data_set[j].values);
        strcpy(stg_val_2015, &add_val_2015[1]);
        double values_2015 = atof(stg_val_2015);
        if (values_2015 == 0)
        {
          iterator_counter_2015_f--;
        }

        sum_2015_f += values_2015;
      } avg_2015_f = sum_2015_f / iterator_counter_2015_f;
      if (year == 2016)
      {
        iterator_counter_2016_f++;
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_2016_f--;
        }

        sum_2016_f += values_2016;
      } avg_2016_f = sum_2016_f / iterator_counter_2016_f;
      if (year == 2017)
      {
        iterator_counter_2017_f++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_2017_f--;
        }

        sum_2017_f += values_2017;
      } avg_2017_f = sum_2017_f / iterator_counter_2017_f;
      if (year == 2018)
      {
        iterator_counter_2018_f++;
        char add_val_2018[10];
        char stg_val_2018[10];
        strcpy(add_val_2018, data_set[j].values);
        strcpy(stg_val_2018, &add_val_2018[1]);
        double values_2018 = atof(stg_val_2018);
        if (values_2018 == 0)
        {
          iterator_counter_2018_f--;
        }

        sum_2018_f += values_2018;
      } avg_2018_f = sum_2018_f / iterator_counter_2018_f;
      if (year == 2019)
      {
        iterator_counter_2019_f++;
        char add_val_2019[10];
        char stg_val_2019[10];
        strcpy(add_val_2019, data_set[j].values);
        strcpy(stg_val_2019, &add_val_2019[1]);
        double values_2019 = atof(stg_val_2019);
        if (values_2019 == 0)
        {
          iterator_counter_2019_f--;
        }

        sum_2019_f += values_2019;
      } avg_2019_f = sum_2019_f / iterator_counter_2019_f;
      if (year == 2020)
      {
        iterator_counter_2020_f++;
        char add_val_2020[10];
        char stg_val_2020[10];
        strcpy(add_val_2020, data_set[j].values);
        strcpy(stg_val_2020, &add_val_2020[1]);
        double values_2020 = atof(stg_val_2020);
        if (values_2020 == 0)
```

```c
      {
        iterator_counter_2020_f--;
      }

      sum_2020_f += values_2020;
    } avg_2020_f = sum_2020_f / iterator_counter_2020_f;
    if (year == 2021)
    {
      iterator_counter_2021_f++;
      char add_val_2021[10];
      char stg_val_2021[10];
      strcpy(add_val_2021, data_set[j].values);
      strcpy(stg_val_2021, &add_val_2021[1]);
      double values_2021 = atof(stg_val_2021);
      if (values_2021 == 0)
      {
        iterator_counter_2021_f--;
      }

      sum_2021_f += values_2021;
    } avg_2021_f = sum_2021_f / iterator_counter_2021_f;
  }
  for (int j = 44; j < 86; j++) /*  Quebec Year-Wise
Average Calculator */
  {
    char add_val_y[10];
    char stg_val_y[10];
    strcpy(add_val_y, data_set[j].year);
    strcpy(stg_val_y, &add_val_y[1]);
    double year = atof(stg_val_y);

    if (year == 2015)
    {
      iterator_counter_2015_q++;
      char add_val_2015[10];
      char stg_val_2015[10];
      strcpy(add_val_2015, data_set[j].values);
      strcpy(stg_val_2015, &add_val_2015[1]);
      double values_2015 = atof(stg_val_2015);
      if (values_2015 == 0)
      {
        iterator_counter_2015_q--;
      }

      sum_2015_q += values_2015;
    } avg_2015_q = sum_2015_q / iterator_counter_2015_q;
    if (year == 2016)
    {
      iterator_counter_2016_q++;
```

```
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_2016_q--;
        }

        sum_2016_q += values_2016;
      } avg_2016_q = sum_2016_q / iterator_counter_2016_q;
      if (year == 2017)
      {
        iterator_counter_2017_q++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_2017_q--;
        }

        sum_2017_q += values_2017;
      } avg_2017_q = sum_2017_q / iterator_counter_2017_q;
      if (year == 2018)
      {
        iterator_counter_2018_q++;
        char add_val_2018[10];
        char stg_val_2018[10];
        strcpy(add_val_2018, data_set[j].values);
        strcpy(stg_val_2018, &add_val_2018[1]);
        double values_2018 = atof(stg_val_2018);
        if (values_2018 == 0)
        {
          iterator_counter_2018_q--;
        }

        sum_2018_q += values_2018;
      } avg_2018_q = sum_2018_q / iterator_counter_2018_q;
      if (year == 2019)
      {
        iterator_counter_2019_q++;
        char add_val_2019[10];
        char stg_val_2019[10];
        strcpy(add_val_2019, data_set[j].values);
        strcpy(stg_val_2019, &add_val_2019[1]);
```

```c
      double values_2019 = atof(stg_val_2019);
      if (values_2019 == 0)
      {
        iterator_counter_2019_q--;
      }

      sum_2019 += values_2019;
    } avg_2019_q = sum_2019_q / iterator_counter_2019_q;
    if (year == 2020)
    {
      iterator_counter_2020_q++;
      char add_val_2020[10];
      char stg_val_2020[10];
      strcpy(add_val_2020, data_set[j].values);
      strcpy(stg_val_2020, &add_val_2020[1]);
      double values_2020 = atof(stg_val_2020);
      if (values_2020 == 0)
      {
        iterator_counter_2020_q--;
      }

      sum_2020_q += values_2020;
    } avg_2020_q = sum_2020_q / iterator_counter_2020_q;
    if (year == 2021)
    {
      iterator_counter_2021_q++;
      char add_val_2021[10];
      char stg_val_2021[10];
      strcpy(add_val_2021, data_set[j].values);
      strcpy(stg_val_2021, &add_val_2021[1]);
      double values_2021 = atof(stg_val_2021);
      if (values_2021 == 0)
      {
        iterator_counter_2021_q--;
      }

      sum_2021_q += values_2021;
    } avg_2021_q = sum_2021_q / iterator_counter_2021_q;
  }
  for (int j = 86; j < 128; j++)  /*  Ontario Year-Wise Average Calculator  */
  {
    char add_val_y[10];
    char stg_val_y[10];
    strcpy(add_val_y, data_set[j].year);
    strcpy(stg_val_y, &add_val_y[1]);
    double year = atof(stg_val_y);

    if (year == 2015)
```

```
      {
        iterator_counter_2015_o++;
        char add_val_2015[10];
        char stg_val_2015[10];
        strcpy(add_val_2015, data_set[j].values);
        strcpy(stg_val_2015, &add_val_2015[1]);
        double values_2015 = atof(stg_val_2015);
        if (values_2015 == 0)
        {
          iterator_counter_2015_o--;
        }

        sum_2015_o += values_2015;
      } avg_2015_o = sum_2015_o / iterator_counter_2015_o;
      if (year == 2016)
      {
        iterator_counter_2016_o++;
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_2016_o--;
        }

        sum_2016_o += values_2016;
      } avg_2016_o = sum_2016_o / iterator_counter_2016_o;
      if (year == 2017)
      {
        iterator_counter_2017_o++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_2017_o--;
        }

        sum_2017_o += values_2017;
      } avg_2017_o = sum_2017_o / iterator_counter_2017_o;
      if (year == 2018)
      {
        iterator_counter_2018_o++;
        char add_val_2018[10];
        char stg_val_2018[10];
        strcpy(add_val_2018, data_set[j].values);
        strcpy(stg_val_2018, &add_val_2018[1]);
        double values_2018 = atof(stg_val_2018);
        if (values_2018 == 0)
        {
          iterator_counter_2018_o--;
        }

        sum_2018_o += values_2018;
      } avg_2018_o = sum_2018_o / iterator_counter_2018_o;
      if (year == 2019)
      {
        iterator_counter_2019_o++;
        char add_val_2019[10];
        char stg_val_2019[10];
        strcpy(add_val_2019, data_set[j].values);
        strcpy(stg_val_2019, &add_val_2019[1]);
        double values_2019 = atof(stg_val_2019);
        if (values_2019 == 0)
        {
          iterator_counter_2019_o--;
        }

        sum_2019_o += values_2019;
      } avg_2019_o = sum_2019_o / iterator_counter_2019_o;
      if (year == 2020)
      {
        iterator_counter_2020_o++;
        char add_val_2020[10];
        char stg_val_2020[10];
        strcpy(add_val_2020, data_set[j].values);
        strcpy(stg_val_2020, &add_val_2020[1]);
        double values_2020 = atof(stg_val_2020);
        if (values_2020 == 0)
        {
          iterator_counter_2020_o--;
        }

        sum_2020_o += values_2020;
      } avg_2020_o = sum_2020_o / iterator_counter_2020_o;
      if (year == 2021)
      {
        iterator_counter_2021_o++;
        char add_val_2021[10];
        char stg_val_2021[10];
        strcpy(add_val_2021, data_set[j].values);
        strcpy(stg_val_2021, &add_val_2021[1]);
        double values_2021 = atof(stg_val_2021);
        if (values_2021 == 0)
        {
          iterator_counter_2021_o--;
        }

        sum_2021_o += values_2021;
      } avg_2021_o = sum_2021_o / iterator_counter_2021_o;
    }
    for (int j = 128; j < 170; j++) /*  Alberta Year-Wise Average Calculator  */
    {
      char add_val_y[10];
      char stg_val_y[10];
      strcpy(add_val_y, data_set[j].year);
      strcpy(stg_val_y, &add_val_y[1]);
      double year = atof(stg_val_y);

      if (year == 2015)
      {
        iterator_counter_2015_a++;
        char add_val_2015[10];
        char stg_val_2015[10];
        strcpy(add_val_2015, data_set[j].values);
        strcpy(stg_val_2015, &add_val_2015[1]);
        double values_2015 = atof(stg_val_2015);
        if (values_2015 == 0)
        {
          iterator_counter_2015_a--;
        }

        sum_2015_a += values_2015;
      } avg_2015_a = sum_2015_a / iterator_counter_2015_a;
      if (year == 2016)
      {
        iterator_counter_2016_a++;
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_2016_a--;
        }

        sum_2016_a += values_2016;
      } avg_2016_a = sum_2016_a / iterator_counter_2016_a;
      if (year == 2017)
      {
        iterator_counter_2017_a++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_2017_a--;
        }

        sum_2017_a += values_2017;
      } avg_2017_a = sum_2017_a / iterator_counter_2017_a;
      if (year == 2018)
      {
        iterator_counter_2018_a++;
        char add_val_2018[10];
        char stg_val_2018[10];
        strcpy(add_val_2018, data_set[j].values);
        strcpy(stg_val_2018, &add_val_2018[1]);
        double values_2018 = atof(stg_val_2018);
        if (values_2018 == 0)
        {
          iterator_counter_2018_a--;
        }

        sum_2018_a += values_2018;
      } avg_2018_a = sum_2018_a / iterator_counter_2018_a;
      if (year == 2019)
      {
        iterator_counter_2019_a++;
        char add_val_2019[10];
        char stg_val_2019[10];
        strcpy(add_val_2019, data_set[j].values);
        strcpy(stg_val_2019, &add_val_2019[1]);
        double values_2019 = atof(stg_val_2019);
        if (values_2019 == 0)
        {
          iterator_counter_2019_a--;
        }

        sum_2019_a += values_2019;
      } avg_2019_a = sum_2019_a / iterator_counter_2019_a;
      if (year == 2020)
      {
        iterator_counter_2020_a++;
        char add_val_2020[10];
        char stg_val_2020[10];
        strcpy(add_val_2020, data_set[j].values);
        strcpy(stg_val_2020, &add_val_2020[1]);
        double values_2020 = atof(stg_val_2020);
        if (values_2020 == 0)
        {
          iterator_counter_2020_a--;
        }

        sum_2020_a += values_2020;
      } avg_2020_a = sum_2020_a / iterator_counter_2020_a;
      if (year == 2021)
      {
        iterator_counter_2021_a++;
        char add_val_2021[10];
        char stg_val_2021[10];
        strcpy(add_val_2021, data_set[j].values);
        strcpy(stg_val_2021, &add_val_2021[1]);
        double values_2021 = atof(stg_val_2021);
        if (values_2021 == 0)
        {
          iterator_counter_2021_a--;
        }

        sum_2021_a += values_2021;
      } avg_2021_a = sum_2021_a / iterator_counter_2021_a;
    }
    for (int j = 170; j < 212; j++) /*  British Columbia Year-Wise Average Calculator */
    {
      char add_val_y[10];
      char stg_val_y[10];
      strcpy(add_val_y, data_set[j].year);
      strcpy(stg_val_y, &add_val_y[1]);
      double year = atof(stg_val_y);

      if (year == 2015)
      {
        iterator_counter_2015_b++;
        char add_val_2015[10];
        char stg_val_2015[10];
        strcpy(add_val_2015, data_set[j].values);
        strcpy(stg_val_2015, &add_val_2015[1]);
        double values_2015 = atof(stg_val_2015);
        if (values_2015 == 0)
        {
          iterator_counter_2015_b--;
        }

        sum_2015_b += values_2015;
      } avg_2015_b = sum_2015_b / iterator_counter_2015_b;
      if (year == 2016)
      {
        iterator_counter_2016_b++;
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_2016_b--;
        }

        sum_2016_b += values_2016;
      } avg_2016_b = sum_2016_b / iterator_counter_2016_b;
      if (year == 2017)
      {
        iterator_counter_2017_b++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_2017_b--;
        }

        sum_2017_b += values_2017;
      } avg_2017_b = sum_2017_b / iterator_counter_2017_b;
      if (year == 2018)
      {
        iterator_counter_2018_b++;
        char add_val_2018[10];
        char stg_val_2018[10];
        strcpy(add_val_2018, data_set[j].values);
        strcpy(stg_val_2018, &add_val_2018[1]);
        double values_2018 = atof(stg_val_2018);
        if (values_2018 == 0)
        {
          iterator_counter_2018_b--;
        }

        sum_2018_b += values_2018;
      } avg_2018_b = sum_2018_b / iterator_counter_2018_b;
      if (year == 2019)
      {
        iterator_counter_2019_b++;
        char add_val_2019[10];
        char stg_val_2019[10];
        strcpy(add_val_2019, data_set[j].values);
        strcpy(stg_val_2019, &add_val_2019[1]);
        double values_2019 = atof(stg_val_2019);
        if (values_2019 == 0)
        {
          iterator_counter_2019_b--;
        }

        sum_2019_b += values_2019;
      } avg_2019_b = sum_2019_b / iterator_counter_2019_b;
      if (year == 2020)
      {
        iterator_counter_2020_b++;
        char add_val_2020[10];
        char stg_val_2020[10];
        strcpy(add_val_2020, data_set[j].values);
        strcpy(stg_val_2020, &add_val_2020[1]);
        double values_2020 = atof(stg_val_2020);
        if (values_2020 == 0)
        {
          iterator_counter_2020_b--;
        }

        sum_2020_b += values_2020;
      } avg_2020_b = sum_2020_b / iterator_counter_2020_b;
      if (year == 2021)
      {
        iterator_counter_2021_b++;
        char add_val_2021[10];
        char stg_val_2021[10];
        strcpy(add_val_2021, data_set[j].values);
        strcpy(stg_val_2021, &add_val_2021[1]);
        double values_2021 = atof(stg_val_2021);
        if (values_2021 == 0)
        {
          iterator_counter_2021_b--;
        }

        sum_2021_b += values_2021;
      } avg_2021_b = sum_2021_b / iterator_counter_2021_b;
    }


    } year_avg(avg_2015_f, avg_2016_f, avg_2017_f, avg_2018_f, avg_2019_f, avg_2020_f, avg_2021_f, avg_2015_q, avg_2016_q, avg_2017_q, avg_2018_q, avg_2019_q, avg_2020_q, avg_2021_q, avg_2015_o, avg_2016_o, avg_2017_o, avg_2018_o, avg_2019_o, avg_2020_o, avg_2021_o, avg_2015_a, avg_2016_a, avg_2017_a, avg_2018_a, avg_2019_a,
avg_2020_a, avg_2021_a, avg_2015_b, avg_2016_b, avg_2017_b, avg_2018_b, avg_2019_b, avg_2020_b, avg_2021_b);

spacer();

  /* Age-Group-Wise Average Calculator*/

/*   Age-Group Wise Averages   */avg_age();
spacer();

/* Initializing Sum & Iterating Counter Variables for Age-Group Averages Calculation */

  // double sum_year - Variable to store the sum of the Value Data Points subsequently in every iteration
// double avg_year - Variable to store the average of the Value Data Points subsequently after all iterations
  // iterator_counter_year - Variable to calculate the number of iterations performed in the for loop

/* 35-49 Age Group Variables */

  double sum_35 = 0;
double avg_35 = 0;
double sum_35_f = 0;
double avg_35_f = 0;
double sum_35_q = 0;
double avg_35_q = 0;
double sum_35_o = 0;
double avg_35_o = 0;
double sum_35_a = 0;
double avg_35_a = 0;
double sum_35_b = 0;
double avg_35_b = 0;
int iterator_counter_35 = 0;
int iterator_counter_35_f = 0;
int iterator_counter_35_q = 0;
int iterator_counter_35_o = 0;
int iterator_counter_35_a = 0;
int iterator_counter_35_b = 0;

/* 50-65 Age Group Variables */

double sum_50 = 0;
double avg_50 = 0;
double sum_50_f = 0;
double avg_50_f = 0;
double sum_50_q = 0;
double avg_50_q = 0;
```

```
  double sum_50_o = 0;
  double avg_50_o = 0;
  double sum_50_a = 0;
  double avg_50_a = 0;
  double sum_50_b = 0;
  double avg_50_b = 0;
  int iterator_counter_50 = 0;
  int iterator_counter_50_f = 0;
  int iterator_counter_50_q = 0;
  int iterator_counter_50_o = 0;
  int iterator_counter_50_a = 0;
  int iterator_counter_50_b = 0;

  /*  65+ Age Group Variables */

  double sum_65 = 0;
  double avg_65 = 0;
  double sum_65_f = 0;
  double avg_65_f = 0;
  double sum_65_q = 0;
  double avg_65_q = 0;
  double sum_65_o = 0;
  double avg_65_o = 0;
  double sum_65_a = 0;
  double avg_65_a = 0;
  double sum_65_b = 0;
  double avg_65_b = 0;
  int iterator_counter_65 = 0;
  int iterator_counter_65_f = 0;
  int iterator_counter_65_q = 0;
  int iterator_counter_65_o = 0;
  int iterator_counter_65_a = 0;
  int iterator_counter_65_b = 0;

    for (int i = 0; i < ARRAY_SIZE; i++)
    {
    for (int j = 2; j < 44; j++)  /*  Federal Age-Group-Wise Average Calculator */
    {
      char add_val_y[10];
      char stg_val_y[10];
      strcpy(add_val_y, data_set[j].age_group);
      strcpy(stg_val_y, &add_val_y[1]);
      double age_group = atof(stg_val_y);

      if (age_group == 35)
      {
        iterator_counter_35_f++;
        char add_val_2015[10];
        char stg_val_2015[10];
        strcpy(add_val_2015, data_set[j].values);
        strcpy(stg_val_2015, &add_val_2015[1]);
        double values_2015 = atof(stg_val_2015);
        if (values_2015 == 0)
        {
          iterator_counter_35_f--;
        }

        sum_35_f += values_2015;
      } avg_35_f = sum_35_f / iterator_counter_35_f;
      if (age_group == 50)
      {
        iterator_counter_50_f++;
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_50_f--;
        }

        sum_50_f += values_2016;
      } avg_50_f = sum_50_f / iterator_counter_50_f;
      if (age_group == 65)
      {
        iterator_counter_65_f++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_65_f--;
        }

        sum_65_f += values_2017;
      } avg_65_f = sum_65_f / iterator_counter_65_f;
    }
    for (int j = 44; j < 86; j++) /*  Quebec Age-Group-Wise Average Calculator  */
    {
      char add_val_y[10];
      char stg_val_y[10];
      strcpy(add_val_y, data_set[j].age_group);
      strcpy(stg_val_y, &add_val_y[1]);
      double age_group = atof(stg_val_y);

      if (age_group == 35)
      {
        iterator_counter_35_q++;
        char add_val_2015[10];
        char stg_val_2015[10];
        strcpy(add_val_2015, data_set[j].values);
        strcpy(stg_val_2015, &add_val_2015[1]);
        double values_2015 = atof(stg_val_2015);
        if (values_2015 == 0)
        {
          iterator_counter_35_q--;
        }

        sum_35_q += values_2015;
      } avg_35_q = sum_35_q / iterator_counter_35_q;
      if (age_group == 50)
      {
        iterator_counter_50_q++;
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_50_q--;
        }

        sum_50_q += values_2016;
      } avg_50_q = sum_50_q / iterator_counter_50_q;
      if (age_group == 65)
      {
        iterator_counter_65_q++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_65_q--;
        }

        sum_65_q += values_2017;
      } avg_65_q = sum_65_q / iterator_counter_65_q;
    }
    for (int j = 86; j < 128; j++)  /*  Ontario Age-Group-Wise Average Calculator */
    {
      char add_val_y[10];
      char stg_val_y[10];
      strcpy(add_val_y, data_set[j].age_group);
      strcpy(stg_val_y, &add_val_y[1]);
      double age_group = atof(stg_val_y);

      if (age_group == 35)
      {
        iterator_counter_35_o++;
        char add_val_2015[10];
        char stg_val_2015[10];
        strcpy(add_val_2015, data_set[j].values);
        strcpy(stg_val_2015, &add_val_2015[1]);
        double values_2015 = atof(stg_val_2015);
        if (values_2015 == 0)
        {
          iterator_counter_35_o--;
        }

        sum_35_o += values_2015;
      } avg_35_o = sum_35_o / iterator_counter_35_o;
      if (age_group == 50)
      {
        iterator_counter_50_o++;
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_50_o--;
        }

        sum_50_o += values_2016;
      } avg_50_o = sum_50_o / iterator_counter_50_o;
      if (age_group == 65)
      {
        iterator_counter_65_o++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_65_o--;
        }

        sum_65_o += values_2017;
      } avg_65_o = sum_65_o / iterator_counter_65_o;
    }
    for (int j = 128; j < 170; j++) /*  Alberta Age-Group-Wise Average Calculator */
    {
      char add_val_y[10];
      char stg_val_y[10];
      strcpy(add_val_y, data_set[j].age_group);
      strcpy(stg_val_y, &add_val_y[1]);
      double age_group = atof(stg_val_y);

      if (age_group == 35)
      {
        iterator_counter_35_a++;
        char add_val_2015[10];
        char stg_val_2015[10];
        strcpy(add_val_2015, data_set[j].values);
        strcpy(stg_val_2015, &add_val_2015[1]);
        double values_2015 = atof(stg_val_2015);
        if (values_2015 == 0)
        {
          iterator_counter_35_a--;
        }

        sum_35_a += values_2015;
      } avg_35_a = sum_35_a / iterator_counter_35_a;
      if (age_group == 50)
      {
        iterator_counter_50_a++;
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_50_a--;
        }

        sum_50_a += values_2016;
      } avg_50_a = sum_50_a / iterator_counter_50_a;
      if (age_group == 65)
      {
        iterator_counter_65_a++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_65_a--;
        }

        sum_65_a += values_2017;
      } avg_65_a = sum_65_a / iterator_counter_65_a;
    }
    for (int j = 170; j < 212; j++) /*  British Columbia Age-Group-Wise Average Calculator  */
    {
      char add_val_y[10];
      char stg_val_y[10];
      strcpy(add_val_y, data_set[j].age_group);
      strcpy(stg_val_y, &add_val_y[1]);
      double age_group = atof(stg_val_y);

      if (age_group == 35)
      {
        iterator_counter_35_b++;
        char add_val_2015[10];
        char stg_val_2015[10];
        strcpy(add_val_2015, data_set[j].values);
        strcpy(stg_val_2015, &add_val_2015[1]);
        double values_2015 = atof(stg_val_2015);
        if (values_2015 == 0)
        {
          iterator_counter_35_b--;
        }

        sum_35_b += values_2015;
      } avg_35_b = sum_35_b / iterator_counter_35_b;
      if (age_group == 50)
      {
        iterator_counter_50_b++;
        char add_val_2016[10];
        char stg_val_2016[10];
        strcpy(add_val_2016, data_set[j].values);
        strcpy(stg_val_2016, &add_val_2016[1]);
        double values_2016 = atof(stg_val_2016);
        if (values_2016 == 0)
        {
          iterator_counter_50_b--;
        }

        sum_50_b += values_2016;
      } avg_50_b = sum_50_b / iterator_counter_50_b;
      if (age_group == 65)
      {
        iterator_counter_65_b++;
        char add_val_2017[10];
        char stg_val_2017[10];
        strcpy(add_val_2017, data_set[j].values);
        strcpy(stg_val_2017, &add_val_2017[1]);
        double values_2017 = atof(stg_val_2017);
        if (values_2017 == 0)
        {
          iterator_counter_65_b--;
        }

        sum_65_b += values_2017;
      } avg_65_b = sum_65_b / iterator_counter_65_b;
    }
    } age_avg(avg_35_f, avg_50_f, avg_65_f, avg_35_q,
   avg_50_q, avg_65_q, avg_35_o, avg_50_o, avg_65_o, avg_35_a
   , avg_50_a, avg_65_a, avg_35_b, avg_50_b, avg_65_b);

  spacer();

  /*           Question 2           */printf("
   |----------------------------------------------------------------------------
   Question
   2----------------------------------------------------------------------------
   n");

  spacer();

  double ProvinceData[4] = {avg_quebec, avg_ontario,
   avg_alberta, avg_british_columbia};

  float lowest = 0;
  int l_counter = 0;
  float highest = 0;
  int h_counter = 0;

  for (int i = 0; i < 4; i++)
  {
    if (i == 0)
    {
      lowest = ProvinceData[i];
      l_counter = i;
      highest = ProvinceData[i];
      h_counter = i;
    }
```

```c
    else
    {
      if (ProvinceData[i] < lowest)
      {
        lowest = ProvinceData[i];
        l_counter = i;
      }
      if (ProvinceData[i] > highest)
      {
        highest = ProvinceData[i];
        h_counter = i;
      }
    }
  }

  if (l_counter == 0)
  {
    printf("The Province with the Lowest percentage of Diabetes is Quebec\n");
  }
  if (l_counter == 1)
  {
    printf("The Province with the Lowest percentage of Diabetes is Ontario\n");
  }
  if (l_counter == 2)
  {
    printf("The Province with the Lowest percentage of Diabetes is Alberta\n");
  }
  if (l_counter == 3)
  {
    printf("The Province with the Lowest percentage of Diabetes is British Columbia\n");
  }

  if (h_counter == 0)
  {
    printf("The Province with the Highest percentage of Diabetes is Quebec\n");
  }
  if (h_counter == 1)
  {
    printf("The Province with the Highest percentage of Diabetes is Ontario\n");
  }
  if (h_counter == 2)
  {
```

```
  printf("The Province with the Highest percentage of Diabetes is Alberta\n");
}
if (h_counter == 3)
{
  printf("The Province with the Highest percentage of Diabetes is British Columbia\n");
}

spacer();

/*          Question 3          */printf("|--------------------------------------------------------------------------- Question 3--------------------------------------------------------------------------- n");

spacer();

printf("Provinces with a Diabetes percentage above the National Average are:\n\n");

for (int i = 0; i < 4; i++)
{
  if (ProvinceData[i] > avg_federal)
  {
    if (i == 0)
    {
      printf("Qubec\n");
    }
    if (i == 1)
    {
      printf("Ontario\n");
    }
    if (i == 1)
    {
      printf("Alberta\n");
    }
    if (i == 1)
    {
      printf("British Columbia\n");
    }


  }
}

spacer();

```

```
  /*          Question 4          */printf("
   |-----------------------------------------------------------------------------
   Question
   4-----------------------------------------------------------------------------
   n");

  spacer();

  lowest = 0, highest = 0;  /*  Re-Initializing lowest &
   highest to 0 */
  l_counter = 0, h_counter = 0; /*  Re-Initializing l_counter
    & h_counter to 0  */

  for (int i = 0; i < ARRAY_SIZE; i++)
    {
    char add_val_y[10];
    char stg_val_y[10];
    strcpy(add_val_y, data_set[i].values);
    strcpy(stg_val_y, &add_val_y[1]);
    double values = atof(stg_val_y);

    if (i == 0)
    {
      lowest = values;
      l_counter = i;
      highest = values;
      h_counter = i;
    }
    else
    {
      if (values < lowest)
      {
        lowest = values;
        l_counter = i;
      }
      if (values > highest)
      {
        highest = values;
        h_counter = i;
      }
    }
  } printf("The Province with the highest & lowest percentage
    of diabetes in a year is British Columbia & Ontario in
   the year's %s & %s\n", data_set[h_counter].year, data_set[
   l_counter].year);

  spacer();
}
```

```
/*  Defining all subsequent UDF's utilized in the program
   */

void credits(void)
{
  spacer();
  printf("                      Toronto Metropolitan
   University\n");
  printf("
   *******************************\n");
  printf("
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");
  sub_spacer();
  printf("
   ---------------------------------------------------------------------------
   n");
  printf("                 CPS 188 Term Project | 2023 Source
    Code\n");
  printf("
   ---------------------------------------------------------------------------
   n");
  sub_spacer();
  printf("                               ------------\n");
  printf("-------------------------------|Group -
   40|-------------------------------\n");
  printf("                               ------------\n");
  sub_spacer();
  printf("Copyright (c) 2023 Sayeed Ahamad, Qurrat-Ul-Ain,
   Tre Spencer, Nourhan Antar\n");
  spacer();
  printf("
   /\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\/\\
   n");
  spacer();
}

FILE* file_o(void)   /*  UDF for file opening    */
{
    char temp[100];
    FILE* file; /*  Initializing file storage variable  */
    file = fopen("statscan_diabetes.csv", "r"); /*  Defining
   the address path of file to be opened and read    */
    if (NULL == file)   /*  Fail Safe mechanism for program
   file opening    */
    {
        printf("Error: File Open cannot proceed\n");
        exit;   /*  Program exits and ends if the fail-safe
   breaks  */
    }
```

```
    fscanf(file, "%s", temp);  /*  Scanning the defined file and storing it at address of Var(temp)    */
    return file;    /*  Returning the file type variable */
}

void file_c(FILE* file) /*  UDF for file closing    */
{
    fclose(file);   /*  Closing the already opened file */
}

void spacer(void) /*  UDF for creating NULL newline spaces for better tabular and visual output */
{
    for (int i = 0; i < SPACERS; i++)
    {
        printf("\n");
    }

}

void sub_spacer(void) /*  UDF for creating NULL newline spaces for better tabular and visual output */
{
    for (int i = 0; i < SUB_SPACERS; i++)
    {
        printf("\n");
    }

}

void avg_province(void) /*  UDF for Province-Wise Averages Header */
{
  printf("|--------------------Province-Wise Averages --------------------|\n");
}

void avg_year(void) /*  UDF for Year-Wise Averages Header */
{
  printf("|--------------------Year-Wise Averages --------------------|\n");
}

void year_avg(double f_2015, double f_2016, double f_2017, double f_2018, double f_2019, double f_2020, double f_2021, double q_2015, double q_2016, double q_2017, double q_2018, double q_2019, double q_2020, double q_2021, double o_2015, double o_2016, double o_2017, double o_2018
```

```c
, double o_2019, double o_2020, double o_2021, double a_2015, double a_2016, double a_2017, double a_2018, double a_2019, double a_2020, double a_2021, double b_2015, double b_2016, double b_2017, double b_2018, double b_2019, double b_2020, double b_2021)  /*  UDF to print Year-Wise Averages in a tabular form */
{
  printf("| Year  |   Canada    |   Quebec    |   Ontario   |    Alberta   |   British Columbia    |\n");
  printf("|---------------------------------------------------------------------------------
n");
  printf("| 2015  |   %.3lf   |   %.3lf   |   %.3lf   |   %.3lf   |     %.3lf     |\n", f_2015, q_2015, o_2015, a_2015, b_2015);
  printf("| 2016  |   %.3lf   |   %.3lf   |   %.3lf   |   %.3lf   |     %.3lf     |\n", f_2016, q_2016, o_2016, a_2016, b_2016);
  printf("| 2017  |   %.3lf   |   %.3lf   |   %.3lf   |   %.3lf   |     %.3lf     |\n", f_2017, q_2017, o_2017, a_2017, b_2017);
  printf("| 2018  |   %.3lf   |   %.3lf   |   %.3lf   |   %.3lf   |     %.3lf     |\n", f_2018, q_2018, o_2018, a_2018, b_2018);
  printf("| 2019  |   %.3lf   |   %.3lf   |   %.3lf   |   %.3lf   |     %.3lf     |\n", f_2019, q_2019, o_2019, a_2019, b_2019);
  printf("| 2020  |   %.3lf   |   %.3lf   |   %.3lf   |   %.3lf   |     %.3lf     |\n", f_2020, q_2020, o_2020, a_2020, b_2020);
  printf("| 2021  |   %.3lf   |   %.3lf   |   %.3lf   |   %.3lf   |     %.3lf     |\n", f_2021, q_2021, o_2021, a_2021, b_2021);
}

void avg_age(void)  /*  UDF for Age-Group-Wise Averages Header  */
{
  printf("|--------------------Age-Group-Wise Averages--------------------|\n");
}

void age_avg(double f_35, double f_50, double f_65, double q_35, double q_50, double q_65, double o_35, double o_50, double o_65, double a_35, double a_50, double a_65, double b_35, double b_50, double b_65) /*  UDF to print Age-Group-Wise Averages in a tabular form  */
{
  printf("| Age-Group |   Canada     |   Quebec     |   Ontario
      |   Alberta    |   British Columbia     |\n");
  printf("
   |---------------------------------------------------------------------------
   n");
  printf("| 35-49 |   %.3lf    |   %.3lf    |   %.3lf    |   %.3
   lf    |      %.3lf      |\n", f_35, q_35, o_35, a_35, b_35);
  printf("| 50-64 |   %.3lf    |   %.3lf    |   %.3lf    |   %.3
   lf    |      %.3lf      |\n", f_50, q_50, o_50, a_50, b_50);
  printf("| 65+ |   %.3lf    |   %.3lf    |   %.3lf    |   %.3lf
      |      %.3lf      |\n", f_65, q_65, o_65, a_65, b_65);
}
```

Listing A.1: *CPS 188 Term Project Source Code*

B GNUplot Scripts

B.1 Question 5

```


set title "Diabetes Percentages in Canada (2015-2021)"
set xlabel "Year"
set ylabel "Diabetes Percentage"
set xtics 1
set key outside right

set style line 1 lw 2 pt 7
set style line 2 lw 2 pt 5
set style line 3 lw 2 pt 9
set style line 4 lw 2 pt 13
set style line 5 lw 2 pt 11

plot "data4a.txt" using 1:2 with lines linestyle 1 linecolor
    rgb "#E41A1C" title "Canada ex. territories", \
       "data4a.txt" using 1:3 with lines linestyle 2 linecolor
     rgb "#377EB8" title "British Columbia", \
     "data4a.txt" using 1:4 with lines linestyle 3 linecolor
    rgb "#4DAF4A" title "Alberta", \
     "data4a.txt" using 1:5 with lines linestyle 4 linecolor
    rgb "#984EA3" title "Ontario", \
       "data4a.txt" using 1:6 with lines linestyle 5 linecolor
     rgb "#FF7F00" title "Quebec"
```

Listing B.1: *GNUplot Source Code*

B.2 Question 6

```


set title "Average Percentages of Diabetes Among Age Groups"
set xlabel "Age Groups"
set ylabel "Diabetes Percentage"
set style data histograms
set style fill solid 1.0 border -1
set boxwidth 0.7 relative
set yrange [0:20]
set ytics 0,2,20
plot 'q6.txt' using 2:xtic(1) with histogram title "Diabetes Percentage"
```

Listing B.2: *GNUplot Source Code*