CPS125 - Digital Computation and Programming
Term Project - Winter 2020
Due Date: ***Monday, April 13rd, 2020***

Report Cover Page and Evaluation Form

Professor: Qinmin (Vivian) Hu

Student Name	Student Number	Initials	Section Number
Hoang Vo	500976404	H.V	16
Erik Kozy	500962839	E.K	13
Erio Hoti	500946852	E.H.	15
Nicholas Vieira	500977730	N.V	16

INTRODUCTION

The purpose of the term project is to implement programs to assess real data in regards to average surface water temperature of the Great Lakes that lie on the Canada-U.S border. The program itself uses calculations and conclusions primarily through the use of helper functions and array structures in the C programming language as a method to compare the accuracy of results with calculated average lake temperatures. Calculations were also made within the program to sort the collected data to determine the warmest and coldest lakes during specific time intervals, in addition to calculating the number of days where the water is above a specific temperature suitable for swimming. All of these requirements were satisfied and calculated using modifications to the C program which allowed for greater efficiency and analysis of results since the language is significantly closer in interaction and calculations to assembly languages. As a result, this program can be analyzed in real-time and to more easily present the results of calculations made within the C program. In order to properly display our calculations in an organized manner; we've decided to associate all related data to each lake instead of just listing our calculations by points (as outlined in the rubric); that way it looks more appealing and it makes searching for specific data on a lake much more efficient since you don't have to look around. The C program itself uses helper functions to manually sort the lake temperatures while 2D arrays are primarily used to create matrices to more effectively group the data of individual lakes, which improves the efficiency of calculations made within the program.

CODE OUTPUT

```
Average Lake Temperatures
6.048438  |9.110110  |8.217863  |11.227589  |9.795616  |10.728356  | The average
temperature of all lakes is 9.19 in 2019, and 10 in 2018

Lake Superior is below the average temperature of all the lakes
The coldest lake is lake Superior
Warmest temperature for lake Superior: 16.60 on 17/8/2019
Coldest temperature for lake Superior: 0.20 on 5/3/2019
For 0 days, the water's warm enough to swim in this lake
For 0 days, the water's too cold to swim in this lake

Lake Michigan is below the average temperature of all the lakes
Warmest temperature for lake Michigan: 21.75 on 20/8/2019
Coldest temperature for lake Michigan: 0.89 on 8/3/2019
For 36 days, the water's warm enough to swim in this lake
For 0 days, the water's too cold to swim in this lake
```

Lake Huron is below the average temperature of all the lakes
Warmest temperature for lake Huron: 20.41 on 7/8/2019
Coldest temperature for lake Huron: 0.20 on 1/3/2019
For 9 days, the water's warm enough to swim in this lake
For 0 days, the water's too cold to swim in this lake

Lake Erie is above the average temperature of all the lakes
The warmest lake is lake Erie
Warmest temperature for lake Erie: 24.86 on 6/8/2019
Coldest temperature for lake Erie: 0.20 on 1/2/2019
For 99 days, the water's warm enough to swim in this lake
For 0 days, the water's too cold to swim in this lake

Lake Ontario is above the average temperature of all the lakes
Warmest temperature for lake Ontario: 23.18 on 6/8/2019
Coldest temperature for lake Ontario: 1.12 on 1/3/2019
For 65 days, the water's warm enough to swim in this lake
For 0 days, the water's too cold to swim in this lake

Lake St. Claire is above the average temperature of all the lakes
Warmest temperature for lake St. Claire: 25.04 on 20/7/2019
Coldest temperature for lake St. Claire: 0.20 on 19/1/2019
For 91 days, the water's warm enough to swim in this lake
For 0 days, the water's too cold to swim in this lake

The warmest overall temperature is 25.04 at lake St. Claire on 20/7/2019
The coldest overall temperature is 0.20 on multiple days in multiple lakes

The sorted warmest average temperatures, in degrees, during the summer are 22.86 at Erie, 22.43 at St. Claire, 20.57 at Ontario, 19.05 at Michigan, 17.74 at Huron, 12.59 at Superior,

The sorted warmest average temperatures, in degrees, during the winter are 2.73 at Huron, 2.43 at St. Claire, 1.56 at Ontario, 1.47 at Erie, 1.47 at Michigan, 0.89 at Superior,

Average Temperatures for each lake in 2018 vs 2019
Lake Superior : 5.87 | 6.05
Lake Michigan : 9.87 | 9.11
Lake Huron : 8.78 | 8.22
Lake Erie : 11.56 | 11.23
Lake Ontario : 10.60 | 9.80
Lake St. Claire : 11.20 | 10.73

WORK BREAKDOWN AND SCHEDULE

Name	Roles
Hoang Vo	Programmer, Report Editor
Erik Kozy	Programmer
Erio Hoti	Programmer
Nicholas Vieira	Report Editor

PROGRAM INFORMATION

The program operates using multiple functions with a mixture of 1D and 2D arrays. 2D arrays are used to define the range of values for maximum and minimum values whereas the void sort function was established as an intermediate method for sorting temperature data relative to one city as defined by the char* pointer variable. To increase efficiency, functions and structures could have been created to incorporate pointers more in the code, however the outcome remained the same. The main function also uses file redirection for more efficient debugging and data collection. The aim of using the sorting function was to initialize a method to sort redirected data from the file into the sorting function which enables the rest of the code to function as the function sorts individual combinations of two lakes through the for loops. The main function then uses simple for loops and if-else statements to evaluate for criteria including the average temperature of all lakes, the warmest and coldest lake and the days suitable for swimming or days where the lakes were frozen during separate time intervals.

CODE EVALUATION

The program begins through variable declarations, where all temperature sums are put into 1D arrays that are defined and initialized as ***double*** *variable_name[6] = {0,0,0,0,0,0}*. They were initialized with elements 0 to limit the way different compilers can interpret the variable initializer, which has caused errors for programmers in the past. Additionally, double variables are used for the data pertaining to the warmest and coldest days present in each lake for the given year. Within the variable declarations as well exists the integer arrays that are used for temperature ranges at which the lakes are adequately warm or cold to swim in. The **char*** variable is also used to declare the names of the lakes with specified sizes directly without the need for the ‘0/’ character within separate arrays for each lake. The data sets are then declared as 2D arrays where the rows indicate the lake data being read (0 being Lake Superior, matching the order of the data file), and the columns being the temperatures for each day for a maximum of 364 days, excluding the [0] cell value. The appropriate code is shown below (Lines 3-17)

```
//VARIABLE DECLARATIONS
FILE *laketemps;
FILE *laketemps2018;
double sum[6] = {0, 0, 0, 0, 0, 0};
double sum2018[6] = {0, 0, 0, 0, 0, 0};
double avgtemp[6] = {0, 0, 0, 0, 0, 0};
double avgtemp2018[6] = {0, 0, 0, 0, 0, 0};
double totalavgtemp = 0;
//maxtemps-> row 0 for temp, row 1 for the lake it is, row 2 for the day it is.
double maxwarmtemp[3][1], maxcoldtemp[3][1];
int warmtoswim[6] = { 0,0,0,0,0,0 }, coldtoswim[6] = { 0,0,0,0,0,0 };
double warmestday[2][6], coldestday[2][6]= { {0,0,0,0,0,0}, { 10000, 10000, 10000, 10000, 10000, 10000}};
int initialspot[] = { 0,1,2,3,4,5 };
char* initials[] = {"Superior", "Michigan", "Huron", "Erie", "Ontario", "St. Claire"};
double data[8][365], data2018[8][365];
```

Figure 1: Variable declarations

The void function “sort” is then declared which sorts two lakes based on temperature and relays it to the main function. For loops are used to determine if the temperature value present at a given x[i] is less than or equal to that of x[j]. The if statement indicates that **if(x[i] < x[j])** then the array cell at x[i] is equal to the array cell at x[j] and thus, it is equal to the temperature value of the first lake in the void function. This condition also sets the array cell at y[i] equal to y[j] and thus, equal to the temperature of the second lake. The else-if statement declares that **if(x[i]>x[j])** the same conditions are satisfied. The void function closes by creating a for loop to determine the temperature for cell *z* in *y[]* which becomes the new reference value of *j*. The program then prints a statement for the temperature at the lake specified within *initials* [j]. There is no return statement as the function is built under the void argument. The code used for the void sort function is shown in Figure 2 (below).

```c
void sort(double x[], char order, int y[]){
    double temp1;
    int temp2;
    int n=6;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (order == 'd') {
                if (x[i] < x[j]) {
                    temp1 = x[i];
                    x[i] = x[j];
                    x[j] = temp1;
                    temp2 = y[i];
                    y[i] = y[j];
                    y[j] = temp2;
                }
            } else if(order == 'a'){
                if (x[i] > x[j]) {
                    temp1 = x[i];
                    x[i] = x[j];
                    x[j] = temp1;
                    temp2 = y[i];
                    y[i] = y[j];
                    y[j] = temp2;
                }
            }
        }
    }
    for(int z=0;z<n;z++){
        int j = y[z];
        printf("%.2lf at %s, ",x[z], initials[j]);
    }
}
```

Figure 2: Program for the *void sort* function.

```c
void getDate(int days) {
    int month, day;
    (1 <= days && days <= 31) ? month = 1, day = days : 1;
    (32 <= days && days <= 59) ? month = 2, day = days - 31 : 1;
    (60 <= days && days <= 90) ? month = 3, day = days - 59 : 1;
    (91 <= days && days <= 120) ? month = 4, day = days - 90 : 1;
    (121 <= days && days <= 151) ? month = 5, day = days - 120 : 1;
    (152 <= days && days <= 181) ? month = 6, day = days - 151 : 1;
    (182 <= days && days <= 212) ? month = 7, day = days - 181 : 1;
    (213 <= days && days <= 243) ? month = 8, day = days - 212 : 1;
    (244 <= days && days <= 273) ? month = 9, day = days - 243 : 1;
    (274 <= days && days <= 304) ? month = 10, day = days - 273 : 1;
    (301 <= days && days <= 334) ? month = 11, day = days - 304 : 1;
    (335 <= days && days <= 365) ? month = 12, day = days - 334 : 1;
    printf("%d/%d/2019\n",day,month);
}
```

Figure 3: Function to convert days to dates

The void function **getdate** is an efficient way to convert the day number to date; the logic behind the algorithm is essentially compact if statements. The first option before the colon is executed if the statement is true, if it's false it'll do nothing (a 1 just acts as an empty command). The day is obtained from shaving off the preceding number of days and month is found by brute force.

```
int main() {
    laketemps = fopen("glsea-temps2019_1024.txt","r");
    laketemps2018 = fopen("glsea-temps2018_1024.txt", "r");

    //if the file doesnt exist or isnt in the right place, exit program
    if (laketemps == NULL || laketemps2018 == NULL) {
        printf("Cannot open file, does it exist?\n");
        return -1;
    }

    for (int i = 0; i < 365; i++) {
        //collect data from the file
        for (int j = 0; j < 8; j++) {
            fscanf(laketemps, "%lf", &data[j][i]);
            fscanf(laketemps2018, "%lf", &data2018[j][i]);
        }
        //printf("%lf \n", data[1][i]);
```

Figure 4: File redirection and for loop file scanning program

The main function begins by initializing variables necessary for file redirection. Whereas *lake temps* reads temperature data from the 2019 text file, *laketemps2018* opens the data collected from 2018. An if condition exists if the file does not exist in the directory and contains a return condition of (-1). If the file exists in the appropriate directory, however, it will continuously get a number input from the file until the end; in this case, 365 inputs of numbers. The values are scanned directly from the file and placed into the 2D arrays with the notation [j][i]. A commented out printf statement closes the for loop segments by printing out the current intermediate data.

Once the data is sorted into 365 numerical inputs, the sum is calculated by using a for loop to increment the sum of the *jth* cell in the 1D sum arrays into the two data arrays that contain the inputs from the preceding code. A set of if branches then determines if the incremented data set is either less than the recorded coldest date or greater than the recorded warmest date. The respective 2D array is then updated with the value of the 2D data array with respect to the *jth* cell. Two other if statements determine if the temperature data is either greater than 20 degrees (warm enough to swim) or less than 0 degrees (lakes are frozen)

```
    //calculate sum
    for (int j = 0; j < 6; j++) {
        sum[j] += data[j + 2][i];
        sum2018[j] += data2018[j+2][i];
        if(data[j+2][i]>warmestday[1][j]){
            warmestday[1][j]= data[j+2][i];
            warmestday[0][j] = i;
        }
        if(data[j+2][i]<coldestday[1][j]){
            coldestday[1][j]= data[j+2][i];
            coldestday[0][j] = i;
        }
        if(data[j+2][i]>20){
            warmtoswim[j]++;
        }
        if(data[j+2][i]<0){
            coldtoswim[j]++;
        }
    }
}
```

Figure 5: Coldest and warmest day if branches

As a follow-up, the average temperature of each lake in the given year is determined by first initializing the maxavg and minavg into 1D arrays initialized at [0]. These values are then filled through a for loop which searches for temperature data for each lake. An if statement branch then determines whether or not the average temperature is greater than the maxavg value or less than the minavg value and updates it accordingly. **(Code in Figure 6)**

```
//calculate avg temp and total avg temp
for (int i = 0; i < 6; i++) {
    avgtemp2018[i] = sum2018[i] / 365;
    avgtemp[i] = sum[i] / 365;
    totalavgtemp += avgtemp[i];
    totalavgtemp2018 += avgtemp2018[i];
    printf("%lf  |", avgtemp[i]);
}
double maxavg=avgtemp[0], minavg=avgtemp[0];
for(int i=0;i<6;i++){
    if (avgtemp[i] > maxavg) {
        maxavg = avgtemp[i];
    }
    if (avgtemp[i] < minavg) {
        minavg = avgtemp[i];
    }
}
```

Figure 6: Average and total temperature data

The maximum and minimum average temperatures are also used to determine the total average temperature of all lakes. The first set of if statements determines whether or not the average temperature is greater than or less than the global lake average to determine which lakes are above and below the total average temperature. The maximum and minimum temperatures are then compared to the average temperature of the lakes from the for loop to search for the warmest and coldest lakes found within the file data. The code then iterates on this by evaluating for the coldest and warmest temperature of each lake on the warmest and coldest date. The third set of if statements in this section of code determines if the warmest day is warmer than the initial value of the maximum warm temperature array and if the coldest day is colder than the maximum coldest temperature. These if statements search for each lake's average and maximum temperatures to determine the total number of days where the water is either suitable to swim in or too cold to swim in. **(Code in Figure 7)**

```
totalavgtemp = totalavgtemp / 6;
totalavgtemp2018 = totalavgtemp2018 / 6;
maxwarmtemp[0][0] = warmestday[1][0];
maxcoldtemp[0][0] =  coldestday[1][0];
//Find the warmest lake, and see which are above/below avg temp
printf("The average temperature of all lakes is %.21f in 2019, and %2.1f in 2018\n", totalavgtemp, totalavgtemp2018);
for (int i = 0; i < 6; i++) {
    if (avgtemp[i] > totalavgtemp) {
        printf("Lake %s is above the average temperature of all the lakes \n", initials[i]);
    } else if (avgtemp[i] < totalavgtemp) {
        printf("Lake %s is below the average temperature of all the lakes \n", initials[i]);
    }

    if(maxavg==avgtemp[i]){
        printf("The warmest lake is lake %s\n", initials[i]);
    }
    if(minavg==avgtemp[i]){
        printf("The coldest lake is lake %s\n", initials[i]);
    }
    printf("Warmest temperature for lake %s: %.21f on day %.1f \n", initials[i], warmestday[1][i], warmestday[0][i]+1);
    printf("Coldest temperature for lake %s: %.21f on day %.1f \n", initials[i], coldestday[1][i], coldestday[0][i]+1);

    if(warmestday[1][i]>maxwarmtemp[0][0]){
        maxwarmtemp[0][0] = warmestday[1][i];
        maxwarmtemp[1][0] = i;
        maxwarmtemp[2][0] = warmestday[0][i];
    }
    if(coldestday[1][i]<maxcoldtemp[0][0]) {
        maxcoldtemp[0][0] = coldestday[1][i];
        maxcoldtemp[1][0] = i;
        maxcoldtemp[2][0] = coldestday[0][i];
    }
    printf("For %d days, the water's warm enough to swim in this lake\n", warmtoswim[i]);
    printf("For %d days, the water's too cold to swim in this lake\n", coldtoswim[i]);
    printf("\n");
}
int initial1 = maxwarmtemp[1][0];
printf("The warmest overall temperature is %.21f at lake %s on day %.1f\n",maxwarmtemp[0][0], initials[initial1], maxwarmtemp[2
printf("The coldest overall temperature is %.21f on multiple days in multiple lakes \n",maxcoldtemp[0][0]);
```

Figure 7: Sorting for warmest and coldest lake temperatures

The next segment of the program calls on the void sort function by first initializing two double variables responsible for the sum and average of all lakes within the dates pertaining to summer **(See Figure 8)**. This encompasses days 175 to 265 of the calendar year which are used as initialization and condition checks within the for loop. The for loop enters data from each lake and inputs it into the correct data range and then updates the summer sum array. Once the summer sum array is updated for each lake, it is then divided to produce the average lake temperature within the range of dates and the data is then printed for each lake.

The sort function is then called to provide the order value ‘d’ from the first if condition in the void function **(See Figure 1)** which calls the function to print the values of the summer average of all lakes. A similar process is completed for the winter temperature averages (days 354 to 365). The entire program concludes by comparing the sorted average temperatures of each lake from 2018 to 2019 by searching each file for the called data and then updating it through the for loop . The values are then printed into the same printf statement.

```
double summersum[6] = {0,0,0,0,0,0};
double summeravg[6] = {0,0,0,0,0,0};
for(int i=171; i<265;i++){
    for (int j = 0; j < 6; j++) {
        summersum[j] += data[j + 2][i];
        }
    }
for(int i=0;i<6;i++){
    summeravg[i] = summersum[i]/94;
}
printf("\n");
printf("The sorted warmest average temperatures, in degrees, during the summer are ");
sort(summeravg,'d', initialspot);

printf("\n");

double wintersum[6] = {0,0,0,0,0,0};
double winteravg[6] = {0,0,0,0,0,0};
for(int i=0;i<79;i++){
    for(int j=0; j<6;j++){
        wintersum[j] += data[j+2][i];
    }
}
for(int i=354;i<365;i++){
    for(int j=0; j<6; j++){
        wintersum[j] += data[j+2][i];
    }
}
for(int i=0; i<6; i++){
    winteravg[i] = wintersum[i]/90;
}
printf("The sorted warmest average temperatures, in degrees, during the winter are ");
sort(winteravg,'d',initialspot);
printf("\n\n");
printf("Average Temperatures for each lake in 2018 vs 2019\n");
for(int i=0;i<6;i++){
    printf("Lake %s : %.2lf | %.2lf \n", initials[i], avgtemp2018[i], avgtemp[i]);
}
return 0;
}
```

Figure 8: Summer and winter averages using the sort function

RESULTS AND CONCLUSIONS

Lake Superior

Warmest Temperature	Coldest Temperature	Days Warm Enough To Swim	Days Too Cold To Swim
16.6ºC	0.20ºC	0 days	0 days
Sorted Warmest (Summer)	**Sorted Warmest (Winter)**	**Average Temperature (2018)**	**Average Temperature (2019)**
12.59ºC	0.89ºC	5.87ºC	6.05ºC

Table 1: Data For Lake Superior

Lake Michigan

Warmest Temperature	Coldest Temperature	Days Warm Enough To Swim	Days Too Cold To Swim
21.75ºC	0.89ºC	36 days	0 days
Sorted Warmest (Summer)	**Sorted Warmest (Winter)**	**Average Temperature (2018)**	**Average Temperature (2019)**
19.05ºC	1.47ºC	9.87ºC	9.11ºC

Table 2: Data For Lake Michigan

Lake Huron

Warmest Temperature	Coldest Temperature	Days Warm Enough To Swim	Days Too Cold To Swim
20.41ºC	0.20ºC	9 days	0 days
Sorted Warmest (Summer)	**Sorted Warmest (Winter)**	**Average Temperature (2018)**	**Average Temperature (2019)**
17.74ºC	2.73ºC	8.78ºC	8.22

Table 3: Data For Lake Huron

Lake Erie

Warmest Temperature	Coldest Temperature	Days Warm Enough To Swim	Days Too Cold To Swim
24.86ºC	0.20ºC	99 days	0 days
Sorted Warmest (Summer)	**Sorted Warmest (Winter)**	**Average Temperature (2018)**	**Average Temperature (2019)**
22.86ºC	1.47ºC	11.56ºC	11.23ºC

Table 4: Data For Lake Erie

Lake Ontario

Warmest Temperature	Coldest Temperature	Days Warm Enough To Swim	Days Too Cold To Swim
23.18ºC	1.12ºC	65 days	0 days
Sorted Warmest (Summer)	**Sorted Warmest (Winter)**	**Average Temperature (2018)**	**Average Temperature (2019)**
20.57ºC	1.56ºC	10.60ºC	9.80ºC

Table 5: Data For Lake Ontario

Lake St. Clair

Warmest Temperature	Coldest Temperature	Days Warm Enough To Swim	Days Too Cold To Swim
25.04ºC	0.20ºC	91 days	0 days
Sorted Warmest (Summer)	**Sorted Warmest (Winter)**	**Average Temperature (2018)**	**Average Temperature (2019)**
22.43ºC	2.43ºC	11.20ºC	10.73ºC

Table 6: Data For Lake St. Clair

From the results of the program, the coldest lake is Lake Superior and the warmest overall lake is Lake Erie. Lake Superior, Lake Michigan, and Lake Huron all fall below the average yearly lake temperature whereas Lake Erie, Lake Ontario and Lake St. Clair all have above average yearly lake temperatures. With the exception of Lake Superior, the average lake temperatures have seen a slight decrease from 2018 to 2019. This defies the general trend that the average temperature of every lake has been on the gradual increase since 1865, as noted by the decreased water levels amongst each lake. [1] Climate change may be responsible for influencing local lake air temperatures along the plateaus of each lake [2]. This in turn leads to a feedback effect on the surface water temperatures which explains the general trend. The results obtained from the data, however, may reflect a more extreme trend where the colder months experience colder lake temperatures on average compared to the previous year; a trend demonstrated within the graphing of Lake Superior's average temperature from 1992 to 2018 compared to that of 2019 data. Possible physical reasons for this unusual skew of data could originate from the polar vortex of cold air moving into the proximity of the Great Lakes and surrounding provinces and U.S. territories which are attributed from higher than average water levels within the Great Lakes themselves. [4]

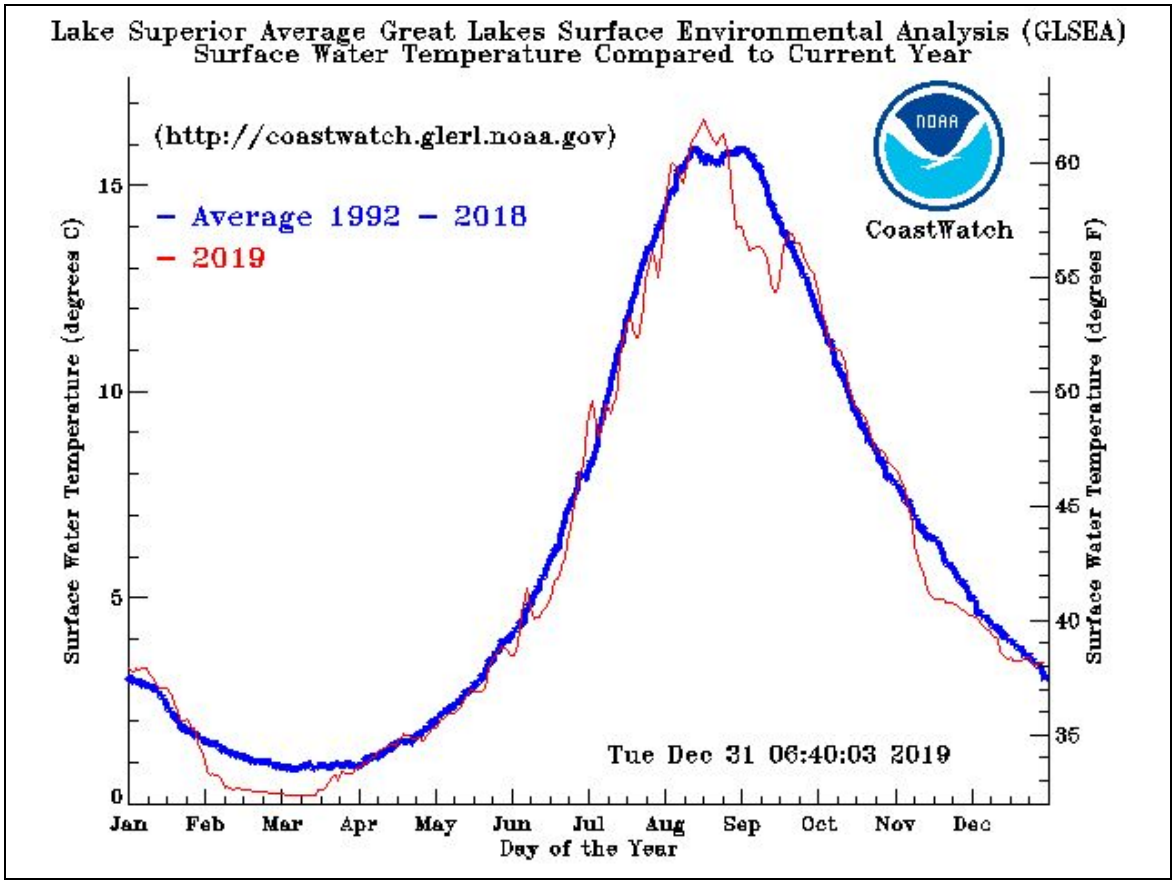

Figure 8: Average Lake Temperature Data of Lake Superior [3]

REFERENCES

[1] United States Environmental Protection Agency “Great Lakes Water Levels and Temperatures” *EPA*, 2016 Avaliable: https://www.epa.gov/sites/production/files/2016-08/documents/print_great-lakes-2016.pdf

[2] J. Shuter, Brain, K. Minns, Charles, Trumpickas, Justin. “Forecasting impacts of climate change on Great Lakes surface water temperatures” *Journal Of Great Lakes Research* 35(3), 454 - 463, 2009, Available: https://journals-scholarsportal-info.ezproxy.lib.ryerson.ca/pdf/03801330/v35i0003/454_fioccoglswt.xml

[3] Great Lakes Statistics, “Physical Characteristics of the Great Lakes”, *NOAA Coast Watch*, 2020. Available: https://coastwatch.glerl.noaa.gov/statistic/

[4] Gillham. Doug, “POLAR VORTEX: Polar vortex puts a ‘harsh’ spin on winter”, *The Weather Network, 2019*. Available: https://www.theweathernetwork.com/ca/news/article/2019-2020-canada-winter-forecast-temperature-precipitation-snow-outlook

APPENDIX

Term Project Program

```
#include <stdio.h>

//VARIABLE DECLARATIONS
FILE *laketemps;
FILE *laketemps2018;
double sum[6] = {0, 0, 0, 0, 0, 0};
double sum2018[6] = {0, 0, 0, 0, 0, 0};
double avgtemp[6] = {0, 0, 0, 0, 0, 0};
double avgtemp2018[6] = {0, 0, 0, 0, 0, 0};
double totalavgtemp = 0, totalavgtemp2018=0;
//maxtemps-> row 0 for temp, row 1 for the lake it is, row 2 for the day it is.
double maxwarmtemp[3][1], maxcoldtemp[3][1];
int warmtoswim[6] = { 0,0,0,0,0,0 }, coldtoswim[6] = { 0,0,0,0,0,0 };
double warmestday[2][6], coldestday[2][6]= { {0,0,0,0,0,0}, { 10000, 10000, 10000, 10000,
10000, 10000}};
int initialspot[] = { 0,1,2,3,4,5 };
char* initials[] = {"Superior", "Michigan", "Huron", "Erie", "Ontario", "St. Claire"};
double data[8][365], data2018[8][365];

void getDate(int days) {
   int month, day;
   (1 <= days && days <= 31) ? month = 1, day = days : 1;
   (32 <= days && days <= 59) ? month = 2, day = days - 31 : 1;
   (60 <= days && days <= 90) ? month = 3, day = days - 59 : 1;
   (91 <= days && days <= 120) ? month = 4, day = days - 90 : 1;
   (121 <= days && days <= 151) ? month = 5, day = days - 120 : 1;
   (152 <= days && days <= 181) ? month = 6, day = days - 151 : 1;
   (182 <= days && days <= 212) ? month = 7, day = days - 181 : 1;
   (213 <= days && days <= 243) ? month = 8, day = days - 212 : 1;
   (244 <= days && days <= 273) ? month = 9, day = days - 243 : 1;
   (274 <= days && days <= 304) ? month = 10, day = days - 273 : 1;
   (301 <= days && days <= 334) ? month = 11, day = days - 304 : 1;
   (335 <= days && days <= 365) ? month = 12, day = days - 334 : 1;
   printf("%d/%d/2019\n", day, month); //the date can be changed to 2018 as well
}

void sort(double x[], char order, int y[]){
   double temp1;
   int temp2;
   int n=6;
```

```
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (order == 'd') {
                if (x[i] < x[j]) {
                    temp1 = x[i];
                    x[i] = x[j];
                    x[j] = temp1;
                    temp2 = y[i];
                    y[i] = y[j];
                    y[j] = temp2;
                }
            } else if(order == 'a'){
                if (x[i] > x[j]) {
                    temp1 = x[i];
                    x[i] = x[j];
                    x[j] = temp1;
                    temp2 = y[i];
                    y[i] = y[j];
                    y[j] = temp2;
                }
            }
        }
    }
    for(int z=0;z<n;z++){
        int j = y[z];
        printf("%.2lf at %s, ",x[z], initials[j]);
    }
}

int main() {
    laketemps = fopen("glsea-temps2019_1024.txt","r");
    laketemps2018 = fopen("glsea-temps2018_1024.txt", "r");

    //if the file doesnt exist or isnt in the right place, exit program
    if (laketemps == NULL || laketemps2018 == NULL) {
        printf("Cannot open file, does it exist?\n");
        return -1;
    }

    for (int i = 0; i < 365; i++) {
        //collect data from the file
        for (int j = 0; j < 8; j++) {
            fscanf(laketemps, "%lf", &data[j][i]);
```

```
        fscanf(laketemps2018, "%lf", &data2018[j][i]);
    }
    //printf("%lf \n", data[1][i]);

    //calculate sum
    for (int j = 0; j < 6; j++) {
        sum[j] += data[j + 2][i];
        sum2018[j] += data2018[j+2][i];
        if(data[j+2][i]>warmestday[1][j]){
            warmestday[1][j]= data[j+2][i];
            warmestday[0][j] = i;
        }
        if(data[j+2][i]<coldestday[1][j]){
            coldestday[1][j]= data[j+2][i];
            coldestday[0][j] = i;
        }
        if(data[j+2][i]>20){
            warmtoswim[j]++;
        }
        if(data[j+2][i]<0){
            coldtoswim[j]++;
        }
    }
}


//calculate avg temp and total avg temp
printf("Average Lake Temperatures\n");
for (int i = 0; i < 6; i++) {
    avgtemp2018[i] = sum2018[i] / 365;
    avgtemp[i] = sum[i] / 365;
    totalavgtemp += avgtemp[i];
    totalavgtemp2018 += avgtemp2018[i];
    printf("%lf  |", avgtemp[i]);
}
double maxavg=avgtemp[0], minavg=avgtemp[0];
for(int i=0;i<6;i++){
    if (avgtemp[i] > maxavg) {
        maxavg = avgtemp[i];
    }
    if (avgtemp[i] < minavg) {
        minavg = avgtemp[i];
    }
```

```
    }
    totalavgtemp = totalavgtemp / 6;
    totalavgtemp2018 = totalavgtemp2018 / 6;
    maxwarmtemp[0][0] = warmestday[1][0];
    maxcoldtemp[0][0] =  coldestday[1][0];
    //Find the warmest lake, and see which are above/below avg temp
    printf(" The average temperature of all lakes is %.2lf in 2019, and %2.lf in 2018\n\n",
totalavgtemp, totalavgtemp2018);
    for (int i = 0; i < 6; i++) {
        if (avgtemp[i] > totalavgtemp) {
            printf("Lake %s is above the average temperature of all the lakes \n", initials[i]);
        } else if (avgtemp[i] < totalavgtemp) {
            printf("Lake %s is below the average temperature of all the lakes \n", initials[i]);
        }

        if(maxavg==avgtemp[i]){
            printf("The warmest lake is lake %s\n", initials[i]);
        }
        if(minavg==avgtemp[i]){
            printf("The coldest lake is lake %s\n", initials[i]);
        }
        printf("Warmest temperature for lake %s: %.2lf on ", initials[i], warmestday[1][i]);
getDate(warmestday[0][i]+1);
        printf("Coldest temperature for lake %s: %.2lf on ", initials[i], coldestday[1][i]);
getDate(coldestday[0][i]+1);

        if(warmestday[1][i]>maxwarmtemp[0][0]){
            maxwarmtemp[0][0] = warmestday[1][i];
            maxwarmtemp[1][0] = i;
            maxwarmtemp[2][0] = warmestday[0][i];
        }
        if(coldestday[1][i]<maxcoldtemp[0][0]) {
            maxcoldtemp[0][0] = coldestday[1][i];
            maxcoldtemp[1][0] = i;
            maxcoldtemp[2][0] = coldestday[0][i];
        }
        printf("For %d days, the water's warm enough to swim in this lake\n", warmtoswim[i]);
        printf("For %d days, the water's too cold to swim in this lake\n", coldtoswim[i]);
        printf("\n");
    }
    int initial1 = maxwarmtemp[1][0];
    printf("The warmest overall temperature is %.2lf at lake %s on ",maxwarmtemp[0][0],
initials[initial1]); getDate(maxwarmtemp[2][0]+1);
```

```
    printf("The coldest overall temperature is %.2lf on multiple days in multiple lakes
\n",maxcoldtemp[0][0]);

    double summersum[6] = {0,0,0,0,0,0};
    double summeravg[6] = {0,0,0,0,0,0};
    for(int i=171; i<265;i++){
        for (int j = 0; j < 6; j++) {
            summersum[j] += data[j + 2][i];
            }
        }
    for(int i=0;i<6;i++){
        summeravg[i] = summersum[i]/94;
    }
    printf("\n");
    printf("The sorted warmest average temperatures, in degrees, during the summer are ");
    sort(summeravg,'d', initialspot);

    printf("\n");

    double wintersum[6] = {0,0,0,0,0,0};
    double winteravg[6] = {0,0,0,0,0,0};
    for(int i=0;i<79;i++){
        for(int j=0; j<6;j++){
            wintersum[j] += data[j+2][i];
        }
    }
    for(int i=354;i<365;i++){
        for(int j=0; j<6; j++){
            wintersum[j] += data[j+2][i];
        }
    }
    for(int i=0; i<6; i++){
        winteravg[i] = wintersum[i]/90;
    }
    printf("The sorted warmest average temperatures, in degrees, during the winter are ");
    sort(winteravg,'d',initialspot);
    printf("\n\n");
    printf("Average Temperatures for each lake in 2018 vs 2019\n");
    for(int i=0;i<6;i++){
        printf("Lake %s : %.2lf | %.2lf \n", initials[i], avgtemp2018[i], avgtemp[i]);
    }
    return 0;
}
```