Function Templates

Workshop 10 (out of 10 marks - 3% of your final grade)

In this workshop, you are to define a function in type-generic form.

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated your abilities

- to code a function template
- to specify the type to implement in a call to a templated function
- to describe the purpose of a constrained cast and its syntax
- to describe to your instructor what you have learned in completing this workshop

SUBMISSION POLICY

The "in-lab" section is to be completed during your assigned lab section. It is to be completed and submitted by the end of the workshop period. If you attend the lab period and cannot complete the in-lab portion of the workshop during that period, ask your instructor for permission to complete the in-lab portion after the period. If you do not attend the workshop, you can submit the "in-lab" section along with your "at-home" section (with a penalty; see below). The "at-home" portion of the lab is due on the day of your next scheduled workshop (23:59).

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible to regularly back up your work.

Late submission penalties:

- In-lab submitted late, with at-home: Maximum of 20/50 for in-lab and Maximum of 50/50 for at home
- Workshop late for one week: in-lab, at-home and reflection must all be submitted for maximum of 50 / 100
- Workshop late for more than one week: in-lab, at-home and reflection must all be submitted for maximum of 30 / 100
- If any of in-lab, at-home or reflection is missing the mark will be zero.

TEMPLATED FUNCTIONS

As a secret agent you have discovered the secret headquarters of an international supervillain, a mastermind of crime. In one room you come across an old USB flash drive containing some FBI data about crime statistics in the USA. The data is in the file crimedata_lab.csv and crimedata_home.csv.

There is some code to analyze it in the files ws10_lab.cpp, and data.h with data.cpp. The program reads the data but does not produce any output.

You suspect the data is fake but you need to prove it. You cannot transmit the data back to your headquarters, so they tell you to finish the program and ask 5 questions about it. They will use the answers to authenticate the data. While looking at the data you discover that some of the rows contain integer data, and some contain double precision floating point.

INPUT FILE FORMAT

```
5
Year,2000,2001,2002,2003,2004
Population,281421906,285317559,287973924,290788976,293656842
ViolentCrime,1425486,1439480,1423677,1383676,1360088
ViolentCrime_Rate,506.5,504.5,494.4,475.8,463.2
GrandTheftAuto,1160002,1228391,1246646,1261226,1237851
GrandTheftAuto_Rate,412.2,430.5,432.9,433.7,421.5
```

The file **crimedata.csv** is shown below. You can open it in excel to show the columns better. Your program will prettyprint it out too.

The first line has a single integer with the number of columns in the file, N. Each row after that contains a string (with no spaces) that tells the name of the data in the row. Then there will be N numbers separated by commas. The numbers can be integer or double. The largest number allowed in this file is 1,000,000,000. (1 billion, or 1 followed by 9 zeros). The smallest number allowed in this file is 0.

The following rows have **integer** data: **Year, Population, ViolentCrime, GrandTheftAuto**

The following rows have **double** data: **ViolentCrime_Rate, GrandTheftAuto_Rate**

The file data is put into variables declared in main. The function **read** was already provided by the criminal organization, and it reads one row of the file. Note that you have to read the rows in the same order they are found in the file, one at a time.

FUNCTIONS

The criminal programmer did the minimum amount of work to analyze the data, and by that he/she only code the function **min** (int *array, int N) to find the minimum value of an array. He/she was also a showoff, so they coded a version of **display** that works on integers. You have to modify these functions

so that they work on both integers and doubles. To save time you will use templates so they can work with any data type at all.

Your assignment, is to implement the following functions.

Function Prototypes

Return type	name	Template parameter	Input	Notes
T	***min***	*T*	*const T* array*, *int N*	Returns the smallest element in the array N is the number of elements in the array
T	***max***	*T*	*const T* array*, *int N*	Returns the largest element in the array
T	***sum***	*T*	*const T* array*, *int N*	Returns the sum of all the elements in the array
double	***average***	*T*	*const T* array*, *int N*	Average is usually computed as a double in statistics because sometimes the average is a fraction between 2 integers. Note: the average of an array of N items is the sum of all the items / N

The functions **display**, **read** and **readRow** have already been provided to you.

QUESTIONS

The code in main() computes answers to the following questions. Question 1 has already been solved for you.

1. Print the total population growth from the beginning to end of the data. (Hint: the beginning of the data is at population[0], and the end of the data is at population[N-1].
2. Between start and finish, did violent crime go up or down?
3. Print the average number of Grand Theft Auto incidents over all the years. Format it to show millions.
4. Print the minimum and maximum number of Violent Crime incidents.

Notes:

1. Note that you can convert a double to an integer using **static_cast<int>( ).**

2. The definition of average of an array of numbers is the sum of the array divided by the count of items inside it.

OUTPUT

Your output should look like this

```
               Year          2000          2001          2002          2003          2004
         Population     281421906     285317559     287973924     290788976     293656842
      ViolentCrime        1425486       1439480       1423677       1383676       1360088
  ViolentCrimeRate         506.50        504.50        494.40        475.80        463.20
    GrandTheftAuto        1160002       1228391       1246646       1261226       1237851
GrandTheftAutoRate         412.20        430.50        432.90        433.70        421.50

Population change from 2000 to 2004 is 12.23 million
Violent Crime trend is down
There are 1.23 million Grand Theft Auto incidents on average a year
The Minimum Violent Crime rate was 463
The Maximum Violent Crime rate was 506
```

IN-LAB SUBMISSION

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload **data.h, data.cpp** and **w10_lab.cpp** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit 200_w10_lab <ENTER>
```

and follow the instructions.

Please note that a successful submission does not guarantee full credit for this workshop.

If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

AT HOME

Submit the reflections below.

AT-HOME SUBMISSION

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload **reflect.txt** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit 200_w10_home <ENTER>
```

and follow the instructions.

Please note that a successful submission does not guarantee full credit for this workshop.

If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

REFLECTIONS

1. What happened when you tried to put your templated functions inside data.cpp? Does it work when you move the entire function to data.h?
2. Move one templated function into ws10_lab.cpp. Does it work now? Does it have to be defined ***before*** it is used, above main(), or can be it defined below main()?
3. In the function average, we used a static_cast to convert an integer to a double. If you convert a double to an integer and print it out, how many digits after the decimal place will it have?
4. Could you have done this lab without templates, by just overloading your functions in data.h to accept both integer and double arguments? What advantage do we get by using templates?
5. Are the following equivalent?

   template<class T> and *template<typename T>* ?