Function Templates

Workshop 9 (out of 10 marks – 3.75% of your final grade)

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

LEARNING OUTCOMES

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

- code a function template
- implement a call to a templated function
- describe the syntax of a constrained cast and its purpose
- describe 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 (see penalties below). The *at-home* portion of the lab is due on the day that is four days after your scheduled in-lab period (23:59:59) (even if that period is on a holiday).

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* portion submitted late, with *at-home* portion: **0** for *in-lab*. Maximum of 7/10 for the entire workshop.

- If any of *in-lab*, *at-home* or *reflection* portions is missing, the mark for the workshop will be **0**/10.

IN-LAB - TEMPLATED FUNCTIONS (60%)

As a secret agent, you have discovered the headquarters of an international supervillain, a mastermind of crime. In one room you come across an old USB flash drive containing FBI data on crime statistics in the USA. The data is in a file named **crimedata.csv**. You have also found some code to analyze this data in files **ws9_lab.cpp**, **Data.h** and **Data.cpp**. The program that you have found needs to be completed.

You suspect that the data is fake, but you need to prove it. However, you cannot transmit the data back to your headquarters. Instead, they have instructed you to complete the program and to answer 4 questions. They will use your answers to authenticate the data. While reading the file records, you discover that some records contain integer data, and some records contain floating-point data.

INPUT FILE FORMAT:

The contents of file **crimedata.csv** are shown below. You can open the file in Excel to show the columns in spreadsheet form.

```
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 first record has a single integer that holds the number of columns, **n**, of data in each record of the file. Each record, other than this first one, contains a string (with no spaces) that describes the data in that record. Each record has **n** comma-separated data fields. The numbers in these data fields can be integers or floating-point numbers.

The largest data value allowed in this file is 1,000,000,000. (1 billion, that is, 1 followed by 9 zeros). The smallest number allowed in this file is 0. The output field width for each record's description is 20 characters. The output field width for each data field in each record is 15.

The records with integer data are marked:
- Year
- Population
- ViolentCrime
- GrandTheftAuto

The records with floating-point data are marked:
- ViolentCrime_Rate
- GrandTheftAuto_Rate

The `main()` program reads the records in the file sequentially one record at a time. The `readRecord()` function reads a single record. The program, after reading all the data, displays the data and ask the 4 questions about it.

READROW FUNCTION:

The Data module includes two functions named `readRow()` with different signatures. Since each function contains the same logic, you can replace the two functions by a single templated function of the same name, store the function template in **Data.h**, and remove the original function definitions from **Data.cpp**.

ANSWERS FUNCTION:

You also need to complete the `answers()` function in **Data.cpp**. Your code in this function answers the following 4 questions.

1. What is the total population growth from the beginning to the 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 your number to show millions.

4. Print the minimum and maximum number of Violent Crime incidents.

Notes:

1. You can convert a double to an integer using `static_cast<int>( )`.

2. The average of an array of numbers is their sum divided by the number of items in the array.

OTHER FUNCTIONS:

Code the following functions as templated functions and store them in **Data.h**, along with your `readRecord()` templated function.

Function Prototypes

Return type	Function name	Template parameter	Function Parameters	Functionality
T	**min**	T	const T* data, int n	Returns the smallest element in data n is the number of elements in data
T	**max**	T	const T* data, int n	Returns the largest element in data
T	**sum**	T	const T* data, int n	Returns the sum of n elements in data
double	**average**	T	const T* data, 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 the items / n
bool	**read**	T	std::istream& input, T* data, int n	Reads n comma-separated data elements from input. Returns true if successful, false otherwise.
void	**display**	T	const char* name, const T* data, int n	Display name right-justified in a field of 20 and each data element in a field of 15

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      506.5        504.5        494.4        475.8        463.2
 GrandTheftAuto     1160002      1228391      1246646      1261226      1237851
GrandTheftAutoRate    412.2        430.5        432.9        433.7        421.5

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.

Upload **Data.h, Data.cpp** and **w9_lab.cpp** to your matrix account. Compile and run your code and make sure everything works properly.

Then, run the following command from your account (use your professor's Seneca userid to replace `profname.proflastname`, and your section ID to replace XXX, i.e., SAA, SBB, etc.):

~profname.proflastname/submit 244XXX_w9_lab<ENTER>

and follow the instructions generated by the command and your program.

> **IMPORTANT**: 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: REFLECTION (40%)

The at-home portion includes only the reflection.

Study your final solution, reread the related parts of the course notes, and make sure that you have understood the concepts covered by this workshop. **This should take no less than 30 minutes of your time.**

Create a file named `reflect.txt` that contains your **detailed description of the topics that you have learned** in completing this workshop and mention any issues that caused you difficulty. Include in your explanation—**but do not limit it to**—the following points:

1. What happens if you try to put your templated functions in **Data.cpp**? Does your source code compile if you move all functions to **Data.h**? Explain.

2. Move one templated function into **ws9_lab.cpp**. Does it work now? Do you need to define the template function above `main()`, **before** it is used, or can you define it below `main()`?

3. Could you have done this lab without templates, by just overloading your functions in **Data.h** to accept integer and double arguments? What is the advantage of using templates?

4. Are the following equivalent?
   ```
   template<class T>
   ```
   ```
   template<typename T>
   ```

AT-HOME SUBMISSION

To submit, run the following command from your account (use your professor's Seneca userid to replace `profname.proflastname`, and your section ID to replace XXX, i.e., SAA, SBB, etc.):

~profname.proflastname/submit 244XXX_w9_home<ENTER>

and follow the instructions generated by the command and your program.

> **IMPORTANT**: 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.