Sabancı University

Faculty of Engineering and Natural Sciences

CS204 Advanced Programming

Fall 2023

Homework #5

Due: 06/12/2023 - 23:55

PLEASE NOTE:

Your program should be robust and handle all relevant programmer mistakes and extreme cases.
You MUST write the code on your own.
You CANNOT HELP any friend while coding.
Plagiarism will not be tolerated!!

1 Introduction

In this assignment, you are required to design and implement a templated vector class, **myVector<T1, T2>**, along with its corresponding iterator class, **myVector<T1, T2>::Iterator**, in C++. The **myVector** class should enable a variety of operations, utilizing a pair of template parameters: **T1** for the value type and **T2** for the unique key type. These operations will be driven by commands read from an input file.

```
vecIntString["Sabanci_University"] = 1999;
vecStringInt[204] = "Advanced_Programming";
```

Your class implementations will be used in a prearranged **main** function so your implementations should strictly follow the input/output format used in main function.

```
myVector<int, std::string> vecIntString;
myVector<int, std::string>::Iterator itrIntString(vecIntString.begin());

myVector<std::string, int> vecStringInt;
myVector<std::string, int>::Iterator itrStringInt(vecStringInt.begin());

/*main function continues with other vector declarations and use cases of commands*/
```

The following sections will detail the requirements for the **myVector** and **Iterator** classes, the expected commands and their functionalities, and the format for the input file. It is imperative that your implementation aligns closely with these guidelines to ensure compatibility with the main function and to meet the homework criteria.

```
template <typename T1, typename T2>
class myVector {
public:
     class Iterator {
     public:
     private:
     };
private:
};
```

2 Classes

Your classes are required to implement the following functionalities:

- A templated class **myVector<T1, T2>** which modifies **std::vector** and supports:
  - Default and copy constructors.
  - Adding and removing elements.
  - Accessing and changing values of elements via subscript operator.
  - Copying a vector into another vector.
  - Checking whether two vectors are equal or not.
  - Processing the data according to the value type of the vector:
    - If the value is string: calculate most frequent word and character for all values of the vector.
    - If the value is arithmetic: calculate the mean, median, standard deviation, max and mode values for all values of the vector.
- An inner **Iterator** class for **myVector** which modifies **std::vector::iterator** and supports:
  - A parametric constructor.
  - Functions to print individual elements and the entire vector.
  - Find any element of a myVector object given the key.
  - Replace the current key of a given element in a myVector object with a new key.

Notice that these are the main functionalities that are **explicitly** asked you to implement so the input/output formats of these functions must exactly match with the use of these functions in main function. In order to provide the full functionality, you might need to implement other functions or operators where deducing those is your job to do.

3 Commands and Input Format

Main function already implements reading and processing the input file. You can follow the below convention to read (understand) the input examples provided to you:

- **push**: Adds an element to a specified vector.
  - Input format: **push**(vector_name, key, value)
- **remove**: Removes an element from a specified vector.
  - Input format: **remove**(vector_name, key)
- **find**: Finds an element in a specified vector.
  - Input format: **find**(vector_name, key)
- **replace_key**: Replaces the key of an element in a specified vector.
  - Input format: **replace_key**(vector_name, old_key, new_key)
- **replace_value**: Replaces the value of an element in a specified vector.
  - Input format: **replace_value**(vector_name, key, new_value)
- **print**: Prints an element or an error message if the element is not found.
  - Input format: **print**(vector_name, key)
- **print_vector**: Prints all elements in a specified vector.
  - Input format: **print_vector**(vector_name)

- **copy**: Copies one vector to another.
  - Input format: **copy**(original_vector, target_vector)
- **check_equal**: Checks if two vectors are equal.
  - Input format: **check_equal**(vector_one, vector_two)
- **process_data**: Processes the data in given vector according to its value type.
  - Input format: **process_data**(vector_name)

  Above format does not represent the corresponding function invocations in main.cpp, you should deduce the function invocations from main.cpp by yourself. Also you can assume that keys, values and vector names are one word only.

4 Some Important Remarks

- **You cannot change anything on main function**. In case you change, you going to receive 0 points from this homework.
- As mentioned on prior sections, **you need to modify** the std::vector and std::vector::iterator classes. You **cannot** use built-in classes like std::map, std::unordered_map, std::algorithm, std::math and functions like std::sort.
- Your classes will be evaluated using various main function structures, including scenarios with multiple and diverse vector types. To ensure compatibility, it is essential that your classes are designed for seamless integration, adaptable for immediate use in different contexts.
- Since type specifications of the myVector is assigned during **compile-time**, type-specific functionalities of the class must be decided also during **compile-time**. To check the specified type during the compilation, you can use expressions like std::is_same or std::is_arithmetic along with constexpr specifier. You can use Google to refer how they can be used.
- Refer to course slides for including template classes into your other files.
- You should use at least C++17 to compile the main.cpp . You can compile your homework like this to make sure it works properly: "g++ *.cpp -std=c++17 -o exec.out"
- Template classes and iterators are fundamental concepts that you will frequently encounter in your career, particularly during interviews. Note that throughout your career you spend most of your time to analyse and modify existing code pieces, so this homework is a good exercise for you to develop these skills. Additionally, they are important topics that are likely to appear in your exams :)

5 Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homeworks we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases.**

What and where to submit (PLEASE READ, IMPORTANT): You should prepare (or at least test) your program using MS Visual Studio 2012 or 2019 C++. We will use the standard C++ compiler and libraries of the above mentioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course). Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.You are asked to submit at least two files this time: Header file that contains your class definitions (you can use the structure in myVector.h) and the cpp file that includes your myVector implementation and main function. Name your files program as follows:

SUCourseUserName_YourLastname_YourName_HWnumber_filename.cpp

Your SUCourse user name is actually your SUNet username that is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the folder name must be:

cago_Ozbugsizkodyazaroglu_Caglayan_hw1_main.cpp

Do not add any other character or phrase to the folder name. Make sure that it contains the last version of your homework program. Compress this folder using WINZIP or WINRAR program. Please use "zip" compression. **"rar" or another compression mechanism is NOT allowed.**. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains your cpp file.
You will receive no credits if your compressed folder does not expand or it does not contain the correct files. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names. **Submit via SUCourse ONLY!** You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!

CS204 Team (Muhammed Orhun Gale, Kamer Kaya)