DSBA Introduction to Programming // HW5 (Problem 1) // Programming Test

General Info

Implement the required functions and send your solution to Yandex.Contest. Write everything in a single file `solution.h` and submit that file.

Function `main()` is not needed, but you will need to check your solution locally. To do that, use the provided file `main.cpp` .

If you have problems with CMake, you may merge the two solution files manually and submit the file without the `main()` function.

All tasks are assumed to be completed in order.

Data structure

The application manages bibliographic citations - entries listing sources of referenced or cited information of a paper.

Information about a single reference is provided to you via a map.

```
    std::map<std::string, std::string> entryData;
```

The contents may look like this:

```
    std::map<std::string, std::string> dataWeb = {
            {"key", "{latex2023}"},
            {"title", "LaTeX"},
            {"year", "2023"},
            {"url", "https://en.wikipedia.org/wiki/LaTeX" }};
```

You will need to turn such data into custom classes.

Classes

Consider the following class `Citation`:

```cpp
class Citation
{
public:
    Citation(const std::string& title, int year)
        : _title(title), _year(year)
    {
    }

    virtual ~Citation() {}
    virtual void printCitation(std::ostream& ostr) const = 0;

protected:
    std::string _title;
    int _year;
};
```

This abstract class is used as a base for other types of citations in this work.

The only objects you will need to create are `WebPage` and `Article`. Other classes are needed either as parts of inheritance hierarchy (`Citation`, `PublishedWork`) or as reference (`Book`).

Task 1: Derived classes [up to 0.1]

Declare and implement two derived classes `PublishedWork` and `Article`.

`PublishedWork` is derived from `Citation` and adds two more fields: `std::string _firstname` and `std::string _lastname`.

`Article` is derived from `PublishedWork` and adds one more field: `std::string _journal`.

Add overridden versions of the function `void printCitation(std::ostream& ostr)` to `Article`. `printCitation()` should output information about the citation in the following format:

```
Title: LaTeX
Year: 2023
Author: Firstname Lastname
Journal: Journal Name Written Here
```

Each line should end with a newline character, including the last one.

Information should be written to the output stream object the functions take as the input argument.

Notes

Class `PublishedWork` may be abstract, tests never try to create objects of this class.

You have an unused class `Book` in code that you may use an example. This class is given only for reference and not used in tests.

Task 2: Add class pointers to a map [up to 0.15]

Implement a function `addCitation` with the following prototype:

```
Citation* addCitation(std::map<std::string, Citation*>& citations
                    , CitationType type
                    , const std::map<std::string, std::string> data
)
```

It should do the following:

1. Determine which class object is being created using `CitationType type` enum which is
    - `CitationType::Article`
    - `CitationType::WebPage`
2. Read data from the map `data`

The contents of the variable `data` may look like this:

```
    std::map<std::string, std::string> dataWeb = {
            {"key", "{latex2023}"},
            {"title", "LaTeX"},
            {"year", "2023"},
            {"url", "https://en.wikipedia.org/wiki/LaTeX" }};
```

or like this
given in `solution.h` at the top. Possible options are:

```
    std::map<std::string, std::string> dataArticle = {
            {"key", "{hdrc}"},
            {"title", "Gradient domain high dynamic range compression"},
{"year", "2002"},
            {"journal", "Computer graphics and interactive techniques"},
            {"lastname", "Fattal"},
            {"firstname", "Raanan"}};
```

Keys always are the same. If the object is a `WebPage`, the keys are *key*, *title*, *year*, *url*. If the object is an `Article`, the keys are *key*, *title*, *year*, *journal*, *lastname*, *firstname*.

Hint: you can access elements of a constant map using method `m.at(key)` instead of operator `m[key]`.

Task 3: Add a virtual function [up to 0.15]

Add a **constant** virtual method `std::string getInlineCitation()` to `Citation` and override it in `WebPage` and `Article` .

This method should return a string that represents how a citation will look in text. One of the common ways to add inline citations is the following:

```
This point was already researched in [Lastname, 2022].
Another point was raised in [Web page, 2023].
```

So, the article appears as the last name of the author and the year of publication.

`WebPage` objects don't contain author names, only year, so for them the function should print `[Web page, 2023]` .

Example

This code:

```
    std::map<std::string, std::string> data = {
            {"key", "{hdrc}"},
            {"title", "Gradient domain high dynamic range compression"},
{"year", "2002"},
            {"journal", "Computer graphics and interactive techniques"},
            {"lastname", "Fattal"},
            {"firstname", "Raanan"}};
    Article art(data.at("title"), std::stoi(data.at("year")),
data.at("firstname"), data.at("lastname"), data.at("journal"));


    std::cout << art.getInlineCitation << std::endl;
```

should print

```
[Fattal, 2002]
```

Task 4: Replace keys with inline citations. [up to 0.25]

Use the previous method to implement function `void insertInlineCitations(std::string& text, const std::map<std::string, Citation*>& citations)` .

This function takes two input variables:

- `text` - a long string of text containing *keys*, the same keys as the ones used in Task 2, formatted as `{keyword123}` with curly braces.
- `citations` - the map of citations from Task 2.

This function should go through the text and replace all occurrences of the *keys* with corresponding *inline citations*.
This documentation page has good examples on how to use `replace` method of strings: https://cplusplus.com/reference/string/string/replace/.

Example

Considering the following citation data

```
    std::map<std::string, std::string> dataWeb = {
            {"key", "{wikiarticle2}"},
            {"title", "Important article"},
            {"year", "2023"},
            {"url", "https://en.wikipedia.org/wiki/Important"}};
     std::map<std::string, std::string> dataArticle = {
            {"key", "{lastname2022a}"},
            {"title", "Gradient domain high dynamic range compression"},
{"year", "2022"},
            {"journal", "Computer graphics and interactive techniques"},
            {"lastname", "Lastname"},
            {"firstname", "Person"}};
```

The text

```
This point was already researched in {lastname2022a}.
Another point was raised in {wikiarticle2}.
```

should become

```
This point was already researched in [Lastname, 2022].
Another point was raised in [Web page, 2023].
```

Task 5: Bibliography in appearance order. [up to 0.2]

Implement a function `void printBibliographyAppearance(std::string& text, const std::map<std::string, Citation*>& citations)` that prints the list of citations in the order in which they appear in the text.

The function takes the same inputs as `insertInlineCitations`:

- `text` - a long string of text containing *keys*, the same keys as the ones used in Task 2, formatted as `{keyword123}` with curly braces.
- `citations` - the map of citations from Task 2.

It should print a list of references - outputs of `printCitation` in the order of the *first appearance* of *keys*.

The list of references should be **enumerated** with numbers going on their own line and formatted as `1.` with a dot.

Example

Considering the following citation data

```
    std::map<std::string, std::string> dataWeb = {
            {"key", "{wikiarticle2}"},
            {"title", "Important article"},
            {"year", "2023"},
            {"url", "https://en.wikipedia.org/wiki/Important"}};
     std::map<std::string, std::string> dataArticle = {
            {"key", "{lastname2022a}"},
            {"title", "Gradient domain high dynamic range compression"},
{"year", "2022"},
            {"journal", "Computer graphics and interactive techniques"},
            {"lastname", "Lastname"},
            {"firstname", "Person"}}; The
```

text

```
This point was already researched in {lastname2022a}.
Another point was raised in {wikiarticle2}.
The point mention in {lastname2022a} is quite important.
```

should result in the following bibliography.

```
1.
Title: Gradient domain high dynamic range compression
Year: 2022
Author: Person Lastname
Journal: Computer graphics and interactive techniques
2.
Title: Important article
Year: 2022
URL: https://en.wikipedia.org/wiki/Important
```

Task 6: Inline citations in last name order. [up to 0.15]

Implement a function `void insertInlineAlphabetical(std::string& text, const std::map<std::string, Citation*>& citations)` .

This function takes the same two input variables:

- `text` - a long string of text containing *keys*, the same keys as the ones used in Task 2, formatted as `{keyword123}` with curly braces. `citations` - the map of citations from
- Task 2.

This function should go through the text and replace all occurrences of the *keys* with corresponding **indices** of citations when ordered the following way.

1. First, all `Article` citations, ordered by last names of their authors. Last names of the authors may be assumed to be unique.
2. Second, all `WebPage` citations, ordered by titles. Titles of web pages may be assumed to be unique.

The first index is **1**.

Don't consider uppercase/lowercase letters, use simple string comparison.

Example

Considering the following citation data

```
    std::map<std::string, std::string> dataWeb = {
            {"key", "{wikiarticle2}"},
            {"title", "Important article"},
            {"year", "2023"},
            {"url", "https://en.wikipedia.org/wiki/Important"}};

    std::map<std::string, std::string> dataWeb2 = {
            {"key", "{wikiarticle3}"},
            {"title", "Very important article"},
            {"year", "2021"},
            {"url", "https://en.wikipedia.org/wiki/MoreImportant"}};

    std::map<std::string, std::string> dataArticle = {
            {"key", "{lastname2022a}"},
            {"title", "Gradient domain high dynamic range compression"},
            {"year", "2022"},
            {"journal", "Computer graphics and interactive techniques"},
            {"lastname", "Lastname"},
            {"firstname", "Person"}};

    std::map<std::string, std::string> dataArticle = {
            {"key", "{article1}"},
            {"title", "Ordering data automatically"},
            {"year", "2023"},
            {"journal", "Journal of Data"},
            {"lastname", "Aaronson"},
            {"firstname", "Person"}};
```

The text

```
This point was already researched in {wikiarticle2}.
Another point was raised in {lastname2022a}. The point
mentioned in {wikiarticle3} is quite important.
It was also brought up in {article1}.
```

should be changed to

```
This point was already researched in [3].
Another point was raised in [2].
The point mentioned in [4] is quite important.
It was also brought up in [1].
```

based on the following order:

1. Article by **Aaronson**
2. Article by **Lastname**
3. Web page **Important article** (starts with **I**)
4. Web page **Very important article** (starts with **V**)