C++ Class Templates

by pratyush

Problem | Submissions | Leaderboard | Discussions | Editorial

A class template provides a specification for generating classes based on parameters. *Class templates* are generally used to implement containers. A class template is instantiated by passing a given set of types to it as template arguments. Here is an example of a class, MyTemplate, that can store one element of any type and that has just one member function *divideBy2*, which divides its value by 2.

```
template <class T>
class MyTemplate {
T element;
public:
MyTemplate (T arg) {element=arg;}
T divideBy2 () {return element/2;}
};
```

It is also possible to define a different implementation of a template for a specific type. This is called *Template Specialization*. For the template given above, we find that a different implementation for type *char* will be more useful, so we write a function *printElement* to print the *char* element:

```
// class template specialization:
template <>
class MyTemplate <char> {
char element;
public:
MyTemplate (char arg) {element=arg;}
char printElement ()
{
return element;
}
};
```

You are given a main() function which takes a set of inputs. The type of input governs the kind of operation to be performed, i.e. concatenation for *strings* and addition for *int* or *float*. You need to write the class template *AddElements* which has a function *add()* for giving the sum of *int* or *float* elements. You also need to write a template specialization for the type *string* with a function *concatenate()* to concatenate the second string to the first string.

Input Format

Input will consist of *N+1* lines where *N* is the number given in the first line of the input followed by *N* lines.

From the second line forward, the type of the following two elements will be provided. The type will be one of *int*, *float* or *string* types only. Out of the following two elements, you have to concatenate or add the second element to the first element.

Constraints

$1 <= N <= 500000$

$1.0 <= value_{float} <= 10.0$, where $value_{float}$ is any float value

$1 <= value_{int} <= 100000$, where $value_{int}$ is any int value

$0 <= len_{string} <= 10$, where len_{string} is the length of any string

The time limit for this challenge is 4 seconds

Output Format

The code provided in the code editor will use your class template to add/append elements and give the output.

Sample Input

```
3
string John Doe
int 1 2
float 4.0 1.5
```

Sample Output

```
JohnDoe
3
5.5
```

Explanation

"Doe" when appended with "John" gives "JohnDoe". 2 added to 1 gives 3, and 1.5 added to 4.0 gives 5.5.

Submissions: 9721
Max Score: 20
Difficulty: Easy

Rate This Challenge:

More

Need Help? Get advice from the discussion forum for this challenge. Or check out the environments page

Current Buffer (saved locally, editable) C++

```
#include ↔
using namespace std;

/*Write the class AddElements here*/
template <class T> class AddElements    {
    public:
        T element;
        AddElements(T i) {
            element = i;
        }
        T add(T i) {
            return element+i;
        }
};
template <> class AddElements <string> {
    public:
        string element;
        AddElements(string i) {
            element = i;
        }
        string concatenate(string element2) {
            return element+element2;
        }
};

int main () {↔}

```

Line: 10 Col: 10

Upload Code as File Test against custom input Run Code Submit Code

Congrats, you solved this challenge!

- ✔ Test Case #0
- ✔ Test Case #1
- ✔ Test Case #2
- ✔ Test Case #3
- ✔ Test Case #4
- ✔ Test Case #5
- ✔ Test Case #6
- ✔ Test Case #7
- ✔ Test Case #8
- ✔ Test Case #9
- ✔ Test Case #10

Next Challenge