Programming Assignment 1 Part 3

The Collection Class

Objectives: This assignment gives you some experience with designing and writing C++ classes using "big five" (see the textbook), operator overloading, exception, and templates (Part 3).

21C Addendum: Please take a moment to update your autograder before *starting this assignment:*

`sudo apt update` *or* `brew update`

`sudo apt upgrade` *or* `brew upgrade`

(10 Points) Create a README based on the standard format

- Submit to Canvas a PDF Version of the README
- Test the Program locally in your Linux Environment
- Starter Code and Autograder is provided for your reference on GitHub
- One complete, call `make deliverable` and upload the output tar-archive and README to Canvas

---

Download the autograder and starter code with (paste all 3 lines at once):

```
REMOTE=updates git-update-agent --init-with \
https://github.tamu.edu/alex-born/csce221-pa1-p3 \
--dir pa1-p3
```

Problem Description

(70 Points) Implement the Templated Class Collection

Please note, templated classes require special consideration. All function definitions **must** be in the same file as the header. There will no longer be a `Collection.cpp` , all the function bodies will now be in a single, large `Collection.h` file.

Convert class `Collection` to accept three template parameters, one for the stored type, one for the color type, one for the size type. Call these three parameters `Obj` , `F1` , `F2` , respectively. These will be analogous to the `Stress_ball` , `Stress_ball_colors` , and `Stress_ball_sizes` , respectively.

Templates frequently get very lengthy. As such you should consider using aliases:
`using CollectionSB = Collection<Stress_ball, Stress_ball_colors, Stress_ball_sizes>;`

The stream extraction (input) and stream insertion (output) operator will be another point of potential complication. Each class `Obj` should implement its own extraction/insertion operator: `istream& operator>>(istream &is, Obj& o);` and `ostream& operator<<(ostream &os, const Obj& o);` . Then a single, generic, templated stream extraction and insertion operator can be made for the class `Collection<Obj, F1, F2>` which look like `istream& operator>>(istream&is, Collection<Obj, F1, F2>& c);`

and `ostream& operator<<(ostream&is, const Collection<Obj, F1, F2>& c);` . As always, these operators live *outside* their respective classes.

(20 Points) Implement the Class Jeans

Complete following functions for the class `Jeans` , similar to how you did in class `Stress_ball` :

- `Jeans();` - *Default Constructor*
- `Jeans(Jeans_colors c, Jeans_sizes s);` - *Parameterized Constructor*
- `Jeans_colors get_color() const;`
- `Jeans_sizes get_size() const;`
- `bool operator==(const Jeans& sb) const;`

Also complete the stream functions *oustide* the class `Jeans` :

- `ostream& operator<<(ostream& os, const Jeans& sb);`
- `istream& operator>>(istream& is, Jeans& sb);`

The definitions for `Jeans_colors` and `Jeans_sizes` are provided for you.

(0 Points) Testing

Test your code thoroughly. Sample code is provided to ensure your collection class works with both `Stress_ball` and `Jeans`

When compiling locally, please use the following compilation command to ensure the warnings are consistent with the autograder:

```
g++ -Wextra -Wall -pedantic -Werror -g -std=c++17 \
  Jeans.cpp  Stress_ball.cpp sample_main.cpp \
  -o test
```

(0 Points) Autograder Addendum

We may (will) test your code with additional classes and/or input files. You may assume that these additional classes will implement their own `operator>>` , `operator<<` , `operator==` , as well as any other functions/methods to establish parity with `Stress_ball` and `Jeans` . You may also assume that any additional input files will be validly formatted. **You may make no other assumptions regarding the structure of new classes**.