Assignment 3

C++ has several primitive types (`int`, `double`, etc) and several core language objects such as `string` and `vector`. However, programs often require new objects / types to achieve elegant designs. One such object may be a `TimeCode` type, which represents an amount of time in hours, minutes, and seconds.

`4:15:32`

The above is an instance of a `TimeCode` object representing 4 hours, 15 minutes, and 32 seconds. An elegant way of implementing this object is to store only one instance variable: `t`. `t` is a number representing only the total number of seconds. 4 hours, 15 minutes, and 32 seconds is `t = 15,332` seconds.

Your task is to implement a `TimeCode` object in `TimeCode.cpp` following this elegant design. There is an intentional layer of abstraction here. From the user's point of view the TimeCode always looks like `4:15:32` but from the programmers point of view it looks like `t = 15332`

Getting Started
Provided to you is a *screenshot* of the `TimeCode.h` file from the solution. It is a screenshot intentionally so that you have to mentally process its contents (by re-typing it). You probably are thinking to yourself, "Oh great! More work! I believe that the minor tedium of re-typing this code will have untold benefits to my academic and professional development for years to come." and you're right. It will.

You should implement all of the methods in that file. You may optionally implement more methods (public or private) if it seems appropriate to do so. DO NOT STORE HR, MIN, or SEC. Your only instance variable should be `t`.

Testing
In addition to the `TimeCode.h` screenshot, `TimeCodeTests.cpp` is provided which contains a small sampling of *some of the tests* you should perform. When grading this assignment I will use a version of `TimeCodeTests.cpp` that contains thorough tests for *every method of the class*. You should significantly expand `TimeCodeTests.cpp` by coming up with and writing many tests for yourself. Your goal is to test as many cases as you can come up with to ensure that your `TimeCode.cpp` code can pass all of *my* tests.

CodeGrade is not used for this assignment. So, you should test your code thoroughly. As a reminder, at my convenience, I can run your code through the tests and give a preliminary check if ask me to.

Details

- `TimeCode`s cannot be negative (for the purposes of this assignment). Most (all?) of your variables and method parameters should therefore by `unsigned`. If a negative value is identified at any point you can throw an exception, which might look like this:

  ```
  throw invalid_argument("Negative arguments not allowed: " + to_string(min));
  ```

  We haven't learned fully about exceptions yet. You are encouraged to skip ahead to chapter 12 in the zyBook to explore them further if you're interested.

- `TimeCode`s also cannot have invalid states. For example 00:61:00 should not exist. There are two possible ways to handle this sort of issue depending on the circumstances. I will leave it to you to figure out when to choose them.
  - 1) Convert the invalid state (0:61:0) to a valid state (1:1:0).
  - 2) Throw an exception.

- Do not import the `math` module / library.

- The core logic of `TimeCode` should be `GetComponents()`, and `ComponentsToSeconds()`, using these two methods, along with `GetTimeCodeAsSeconds()` it should be relatively easy to implement all other methods (even the constructors).
  - `ComponentsToSeconds()` should contain the code to convert a given hr, min, and sec to the corresponding `t` value. It is static.
  - `GetComponents()` should contain the code to convert an existing `t` into hr, min, and sec. It is not static.
  - All other methods that need to do such conversions should use these methods instead of re-writing the conversion code.

- Addition and subtraction are intuitive and straightforward conceptually (1:15:22 + 2:9:5 = 3:24:27). They can be confusing when considering roll-over (1:15:55 + 0:1:25). Be careful implementing them!

- Although minutes and seconds are limited to 60, there is no such limit on the number of hours.

- `SetHours()`, `SetMinutes()`, and `SetSeconds()` are unique functions to be used for user convenience. They should change the number of hours / minutes / seconds respectively **ONLY** (no roll-over). See the sample tests for more details.

- The constructor allows for very large seconds allowing for roll-over. It should be possible to input something like hr = 2, min = 71, sec = 234719572143
  `TimeCode tc = TimeCode(2, 71, 234719572143); //` *valid code*
  The program should compute the corresponding (valid) `TimeCode` after performing all roll-over as necessary.

- While minutes and seconds are limited to at most 59, the hours has no such limit.

- You should not store hours or minutes or seconds directly. You should only store the total number of seconds: `t`.

- 187:53:27 is a valid timecode. 53:187:27 is not.

- Your program should have a `Makefile` containing an "all" directive that compiles the program and outputs an executable named `tct` (time code tests).

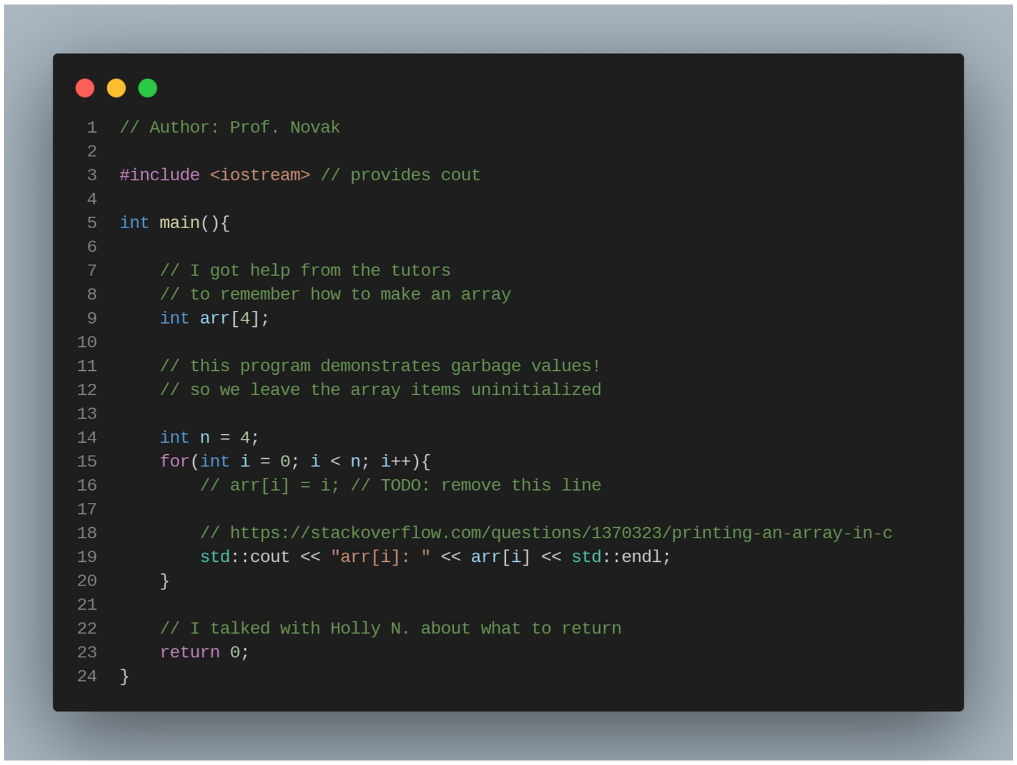
- Multiplication and division do not make sense between two `TimeCode` objects (1:0:0 * 0:15:0 = ?). So, they are implemented using a `TimeCode` and a **`double`** (1:0:0 * **0.5** = 0:30:0).

  Actually, if you think about it this is normal. What is 1 apple divided by ¼ of an apple? Is it "4 chunks of apple" ? That is confusing semantically. But, 1 apple divided by 4 (people, or maybe just unit-less) is ¼ of an apple.

There is no sample output for this program. You are done when your `TimeCode.cpp` implements all the methods in `TimeCode.h` and passes all of your tests in `TimeCodeTests.cpp` The `main()` for this program simply calls your tests and does whatever else you want it to for your own convenience.

If you are reading this far into the assignment you're probably a very diligent and thorough student. Good job! I know you will go far in life.

Proper Comments and Citations
Your code should have comments that explain "why" the code is written the way it is. You should also use comments to cite sources you used such as websites, tutors, and classmates.



```cpp
// Author: Prof. Novak

#include <iostream> // provides cout

int main(){

    // I got help from the tutors
    // to remember how to make an array
    int arr[4];

    // this program demonstrates garbage values!
    // so we leave the array items uninitialized

    int n = 4;
    for(int i = 0; i < n; i++){
        // arr[i] = i; // TODO: remove this line

        // https://stackoverflow.com/questions/1370323/printing-an-array-in-c
        std::cout << "arr[i]: " << arr[i] << std::endl;
    }

    // I talked with Holly N. about what to return
    return 0;
}
```

Submission: Your program will be submitted using GitHub classroom.

After following the instructions on canvas you should have a repository cloned to your computer which contains these assignment instructions. That same repository will be used for you to submit your solutions.

1) Edit and write code in the repo on your computer. Use `git add`, `git commit`, and `git push` to put that code onto github. I have access to the repo. You can make as many commits as you like for this assignment with no penalty.

2) I will grade the final commit made before the due-date.

3) You don't have to make any submission on canvas.