Sonoma State University
Computer Science Department
CS 460 – Fall 2023 – Watts

Exercise 2 Prelim 2

(Due date: Monday, 18 September 2023, 3 pm.)

Designing any type of software requires several phases. Our initial phases were the **request** and **proposal** phases. We are now working on the **design** and **analysis** phases.

We now have a preliminary design. We need to analyze it for completeness and feasibility.

Please carefully review the preliminary version of the class "money" has which has been posted in the file ~tiawatts/cs460pickup/money.h (and at the end of this document.)

Some questions:

- Do we need additional constructors?
- Do we need additional operators?
- How do you envision the actions of each operator?
- Do we need addition mutators (getters) and/or accessors (setters)?
- Do we need additional attributes (variables) and/or methods (functions) in the private section of this class?
- Do you have other ideas regarding the attributes declared in the private section of this class?
- Do you have questions about the syntax of the method prototypes included in this interface?
- And others…

You are not expected to implement these functions at this time. However, writing brief pseudo code is one method of analysis.

Please bring a hard copy of your analysis to our class meeting on Monday. The goal is to complete the interface for money by the end of the day.

Also, please take a picture (or screenshot) of what you consider to be the most important part of your analysis document (questions, observations, additions) and upload it to the course dropbox as lastnameE2.jpg or lastnameE2.png – this is practice for Project 1 Prelim.

```
#ifndef MONEY_H
#define MONEY_H

#include <iostream>
using namespace std;

class money
{
    public:
        // Constructors
        money ();
        money (const money & M);
        // Destructor
        ~money ();
        // Operators
        // Assignment
        money & operator = (const money & M);
        // Extraction (input)
        friend istream & operator >> (istream & input, money & M);
        // Insertion (output)
        friend ostream & operator << (ostream & output, const money & M);
        // Arithmentic operators
        money operator + (const money & M) const;
        money operator += (const money & M);
        money operator - (const money & M) const;
        money operator -= (const money & M);
        money operator * (const double & F) const;
        money operator *= (const double & F);
        money operator / (const double & F) const;
        money operator /= (const double & F);
        // Logical operators
        bool operator == (const money & M) const;
        bool operator != (const money & M) const;
        bool operator < (const money & M) const;
        bool operator <= (const money & M) const;
        bool operator > (const money & M) const;
        bool operator >= (const money & M) const;
        // Accessors and Mutators
        unsigned getDollars () const;
        unsigned getCents () const;
        void setDollars (unsigned D);
        void setCents (unsigned C);
        unsigned * getCurrency () const;
        void setCurrency (unsigned * C) const;
        unsigned & Dollars ();
        unsigned & Cents ();
    private:
        // Possible attributes
        bool positive;
        unsigned dollars, cents;
        unsigned size;       // required
        unsigned * currency; // required
};

#endif
```