Tatyana Vlaskin
Week 4 Report

Problem 1:

Understanding:

I should say that when I looked at this problem, I was so confused. This problem looked identical to the one that we had last week. I had to get clarification on the discussion board. The difference between last week's assignment and this week assignment is that we need to use functions. For this problem we need to write a simple function that accepts two strings and returns whether they have the same contents. We are allowed to reuse code from last week.

Design:

1. I've declared function: void stringCompare (string string1, string string2);
2. I went back to the last weeks code and moved code that involves string comparison to the function definition.
3. Finally, I called function stringCompare (string1, string2) in the main method.

Everything worked as expected. After all this was very straightforward problem.

Problem 2:

Understanding:

We need to write a function that acts like a random number generator after user enters min and max of the range. We are allowed to reuse last weeks code.

Design/Implementation:

1. I already have a code from last week, so my design and implementation will be done simulateously
2. Declare a function: int randInput(int min, int max);
3. Define function:

```
int randInput(int min1, int max1){
    return rand() % (max1 - min1+ 1) + min1; // random will be in the range
of min1-max1
    ////return myRand;
    cout <<endl; // console output of the random number
}
```

4. While looking at my last week's code for this problem, I've realized that I had a mistake in my random number generator. My code was not generating numbers in the range between min-max because I put extra brackets in the rand() % **(**(max1 - min1+ 1) + min1**)**. Because of the extra brackets (in red color), I was generating random numbers between 0 and (max+1). This problem was fixed.
5. Next, I've called int randomNumber = randInput(min1, max1); in the main method. It is possible to generate a random number using 1 line expression as we saw last week. However, it is a good idea to have a function for random number generator in case we need to generate a random number several times in the program and later change the range between which we want to generate random number. When we have a function, we just go to the random number function adjust the range there and the change is automatically applied to all calls of random number function (you do not need to make adjustments to each call one at a time).
6. Looking at my code from last week, I saw that there was still a redundancy. The user is asked to enter 2 numbers, one for max and one for min. At the same time, I have a code for input error handling. Any time user enters a string, or a character, they get an error message. The user needs to enter 2 numbers, the input error test is done x2, and thus there is a redundancy in the code. I've tried to come one with the way to move that part of the program into the function.
7. Using ideas from: http://stackoverflow.com/questions/4604500/use-of-for-in-a-c-sharp-application, the double getNumber() function was created to eliminate redundancy:

```
 {

   for(;;) // I found out that this is equivalent to while (true) {}
   {
    double value;
    cin >> value;
    if(cin.fail())  // if the cin failed
    {
      cin.clear();  // reset scin
      string garbage; // declaration of the string this needs to be discarded
      getline(cin, garbage); // ignores the  rest of line
      cout << "\nINVALID ENTRY. That wasn't a number, try again. ";
    }
    else
{
      return value;
```

```
    break;
  }
```

Problem 3:

Understanding:

We need to write a program that lets the user to input litters of gas consumed and miles that were driven.

Once the information is entered, the program needs to compute the number of miles per gallon.

Calculation need to be done in a separate function.

Conversation from litters to gallons needs to be done using globally defined constant for the number of litters per gallon.

Users need to have an option to repeat the program as often as they want.

Design:

1. Globally define the constant:
    a. const double GALLONS_PER_LITER = 0.264179; // globally defined constant
2. Declare functions that will be used in the program. I am planning to use 3 different functions in the program
    a. First one for the user to make valid input for litters and miles. This function will let the use enter a number, and checks to make sure that the number is positive and not a character/string. The function will be: double getdouble();
    b. Second function will convert litters into gallons using the globally defined constant: GALLONS_PER_LITER = 0.264179. It will return: liters * GALLONS_PER_LITER = gallons
    c. Third function will calculate the number of miles per gallon delivered by the car. This function will return miles_per_gallon = (miles/gallons);
3. Ill also use the following code to control precision. Ill set precision to 2.

```
cout.setf(ios::fixed);
cout.setf(ios::showpoint);
cout.precision(2);
```

Problem 4:

Understanding:

We need to write a program that computes hat, jacket and waist size.

1. Hat size = ((weight/height)*2.9)
2. Jacket size:
    a. If you are under 39, jacket size = ((height*weight)/288);
    b. Staring at age 40, there is an adjustment that needs to be done every 10 years. The adjustment = ((age - 30)/10.0)*(1/8.0);
3. Wait size:
    a. If you are under 28, waist size = weight/5.7
    b. Starting at 28, there is an adjustment that needs to be done every 2 years. The adjustment = ((age - 28)/2.0)*(1/10.0);

Once the computation is done that sizes information is displayed to the user.

Design:

1. For this program, Ill make 5 functions:

I. Function that lets the user to enter a number, and checks to make sure that entered number is positive and not a character/string : double getdouble();
II. Function to compute hat size: double hat_size(double weight, double height);
III. Function to compute jacket size: double jacket_size(double weight, double height, int age);
IV. Function to compute waist size: double waist_in_inches(double weight, int age);
V. Function declaration to set precision. For fat and jacket size, precision will be set to 0. For wait size, precision will be set to 2.

2. Function get double(); was described in previous problems
3. Function to compute hat size: double hat_size(double weight, double height):
    a. return ((weight/height)*2.9);
4. Function to compute jacket size: double jacket_size(double weight, double height, int age);

```
Jacket size=  ((height*weight)/288);
        If (age >= 40 && ((age % 10) == 0))
        {  adjustment = ((age - 30)/10.0)*(1/8.0);
        Jacket size post 40 = jacket size + ((age - 30)/10.0)*(1/8.0);
```

5. Function to compute wait size: double waist_in_inches(double weight, int age);
    Waist size = weight/5.7;
        If (age >= 28 && ((age % 2) == 00))
        { adjustment = ((age - 28)/2.0)*(1/10.0);
        waist size post 28 = waist size + ((age - 28)/2.0)*(1/10.0);
6. Function to set precision will have the following code (taken from the book):
        cout.setf(ios::fixed);
        cout.setf(ios::showpoint);
        cout.precision(precision);
7. All the functions will be called in the main method.

Problem 5

Understanding:

We need to write a program that randomly choses 4 winners out of a pool of 25.

Design:

1. Declare and define function void winner(); that will randomly chose 4 winners
2. Include srand(time(NULL)); . This will guaranty that time the program is run or **loop is executed** a different random numbers will be displayed. (loop is executed is in red because during the implementation stage, it was discovered that that was not the case. See end of the document for more details.)
3. Do something like this:
    for (int consetant = , contestant <=25, contestant ++)
    {
        While (prize >= 1 && prize <= 4)
        { Int contestant = rand ()%25 +1}
    }
    Prize++;
4. Call the void winner function in the main function.

Reflection:

My design has failed. The code for my original void winner (); function is provided below:

```
void winner ()
{
    //srand( (unsigned)time(NULL)); // each time program is run, different random will be displayed
    int prize = 1; // declaration and initialization of prize variable. There are total of 4 prizes.
    // loops thought each contestant one at a time. There are total of 25 contestants.
    srand(time(NULL));
     //srand(time(0));
    for (int contestant = 1; contestant <= 25; contestant ++)
    {
       // srand(time(NULL));
        while(prize >=1 && prize <=4) // loop to distribute prizes, there are total of 4 prizes
        {
        int contestant = (rand()% 25) + 1; // randomly chooses a contestant from a pool of 25
        cout << "Prize number " << prize << " goes to contestant " << contestant << endl;
        cout << endl;//new line
        prize ++; // increments prize number
        }
    }
}
```

The problem was that once in a while, I was getting same random number. It was not making any sense. I had to resort to get assistance on the discussion board. Yulia has helped me a lot. It turns out that srand function does not work the way I used to think it worked. I overestimated the “power of the srand function.” My understanding prior to this assignment was that when I seed srand function, that guaranties that I will not get the same random numbers sequence each time the program is run and I will not get the same random number if I start a program and there is a loop that is executed 100 times to display a random number. It turned out that when you execute the loop, you can get the same number randomly. This was new to me. With Yulias help, I’ve changed a code of the void winner ();. In my current code, I check and makes sure that none of the random number are the same.