```
public class Kuba {
        private Integer Currency;
```

Inside Kuba I have two types of Add functions. One gets a new Kuba object and the other gets a Galli object.

```
public void Add(Kuba newKuba){
```

I am converting the new Currency value from integer to string in our Add function, which takes a Kuba object. Then we convert it to a string array (sArray) with the help of the split function in order to navigate over it. In the for loop, I convert the indexes of the sArray to integer and multiply by 7index and throw these values, which I obtained, into the integer sArray2 that I created before. These operations will help me convert to decimal. I apply the same operations for the existing Currency values and assign them to sArray3. Then, in my first for loop, I collect the indexes of sArray2, in my second for loop I collect the indexes of sArray3 and finally with allSum, I add them all and get my total value in decimal. In the next step, I convert the allSum value, which I obtained in the next step, to string again and convert it to seventh base. However, since I get the reverse of the number while doing this conversion, then again I print them in reverse with for loops and find the right number with the logic that the inverse of inverse is itself. Finally, I show my result with ShowCurrency. (I can run it by typing kuba1.Add(kuba2).)

```
}
public void Add(Galli newGalli){
```

Here I am doing the same process as above, except for a few differences, to add the Kuba and Galli. One of my differences is that Galli is in nine base, so I convert the Galli value that I get from the user from nine base to decimal in the for loop. Then I convert the Kuba value from base seven to base decimal. After completing my decimal conversion, I write the same codes as in Add(Kuba newKuba). Since the second difference is $1(\text{Kuba})_{10} = 2(\text{Galli})_{10}$ in our system, I continue my process by dividing the sum1 value I obtained into two. Now the values I have obtained are suitable numbers for Cuba and I do my necessary additions. Since we are working in Kuba object, I keep the final value in seven base and print it to the screen with ShowCurrency. (I can run it by typing kuba1.Add(galli1).)

```
}
public void GetCurrency(){
```

When we call the GetCurrency function over the reference of any object we have created, I get a number from the user. If the length of the number I get is greater than 7, if the number contains the digits 7,8,9, I want the user to enter an appropriate new value by printing an error. If the length of the value entered by the user is less than 7 and does not contain the numbers 7,8,9, that is, if it is a suitable value for Kuba, I keep this value in Currency.

```
}
public void GetCurrency(Integer newKuba){
```

Here, I check the validity of the Integer newKuba value we will get in a similar way. If appropriate, we keep the entered value in Currency, if not, I request a new value from the user.

```
}
public void ShowCurrency(){
```

It allows us to print whatever our current Currency value is to the screen.

```
}
public void Subtract(Kuba newKuba){
```

In this part, I wrote the same code as in `Add(Kuba newKuba)`because the only difference is that I don't do subtraction instead of addition. I summed up my sum1 and sum2 values that I obtained in `Add(Kuba newKuba)`. I opened an if loop for `Subtract(Kuba newKuba)` and subtracted the smaller value from the larger value entered instead of subtracting the second value from the first value to avoid a negative result. Again, I printed it to the screen with ShowCurrency.

```
}
public Galli Convert(){
```

In this step I created a Galli object and then converted it to string. I converted it to string array with the help of split function. I converted the seven base number I got with the for loops to decimal. Again, not forgetting the rule of $1(\textit{Kuba})_{10} = 2(\textit{Galli})_{10}$ I multiplied the sum1 value I obtained by 2 and assigned it to galley. I wrote down my

nine base conversion codes like I do everywhere. I was keeping my numbers in reverse (like 123 as 321) in the array. So I inverted again to keep the correct number and then returned newGalli.
}

I applied all the same steps for Kuba as I did for Galli and I tried all of them in main.