OzoBlockly Challenge: Evo the Coin Change Maker

By Richard Born
Associate Professor Emeritus
Northern Illinois University
rborn2@niu.edu

Introduction

Before the advent of computers, making proper change for a sale at a store was the job of the clerk. If the bill for the purchased items was, for example $1.59, and the customer gave the clerk $2, the clerk would have to figure out (1) how much the change should be, and (2) what denominations of coins to give the customer in return. The clerk would mentally subtract 59 from 100 to determine that the amount of change should be 41 cents. There are, of course, many ways to make the change: four dimes and a penny; or a quarter, dime, and six pennies; or even 41 pennies, which would probably make the customer somewhat unhappy. In one sense the best way to make the change would be to give the customer a quarter, dime, nickel, and penny.

This last way is the generally accepted way to make coin change, assuming that there is a sufficient supply of each of the four coin denominations. Note that we are also assuming that the half-dollar coin does not enter the picture, as its circulation is quite small in this day and age. This algorithm for making change looks at the coins in order from largest to smallest denomination, first determining the number of quarters, then the number of dimes in the remaining change, then the number of nickels in what remains, and finally the number of pennies.

We've all been at a grocery store, fast-food restaurant, or supercenter store, to mention just a few, and grabbed our change from a dispenser attached to a point-of-sale terminal. There is a good chance that the point-of-sale terminal used the above algorithm to determine your coin change. So the question in this day and age is "how is this coin change algorithm programmed." Designing the program for this algorithm is an excellent exercise for any student of computer science. What better way to do this than to program Evo to determine the change amount and then speak the number of quarters, dimes, nickels, and pennies making up the change!

There are two OzoBlockly blocks that are particularly helpful when implementing the coin change making algorithm. They are shown in Figure 1.

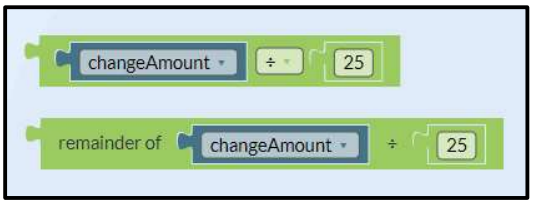


Figure 1

You need to remember that OzoBlockly division is integer division, and as such the division process results in the quotient with the remainder missing. Thus, if the *changeAmount* was 68 cents, the result of the division would give a quotient of 2, the number of quarters required in the coin change! The remainder when *changeAmount* is divided by 25 would be 18. 18 could then, in turn, be divided by 10 to determine the number of dimes, and so on through nickels and pennies.

The Ozomap for this Challenge Lesson

In this program, we are only concerned with the coin change that would be dropped into the coin dispenser, not with any paper money change returned to the customer by the clerk. Figure 2 shows a small copy of the map that we will use to discuss how your program should handle Evo's behavior. A larger copy of the map, suitable for printing and using with Evo and your OzoBlockly program, can be found on the final page of this document.

You will immediately notice that the map consists of two main sections. In the red portion at the top, Evo starts at the location labeled "Start" facing the direction shown by the gray arrow. Evo will stop at the first intersection and speak the billed cents. As an example, the value of the billed cents might be $.32. From this, Evo would calculate the total coin change due the customer as 100 - 32 = 68 cents.

Evo will then move to the next intersection, where he will wait until the user presses Evo's button. This gives the user a chance to determine (on a piece of scratch paper, if necessary) the proper way in which to make the coin change. Evo will then move to the blue portion of the Ozomap where he will speak the number of quarters, dimes, nickels, and pennies that should be given to the customer as proper change.

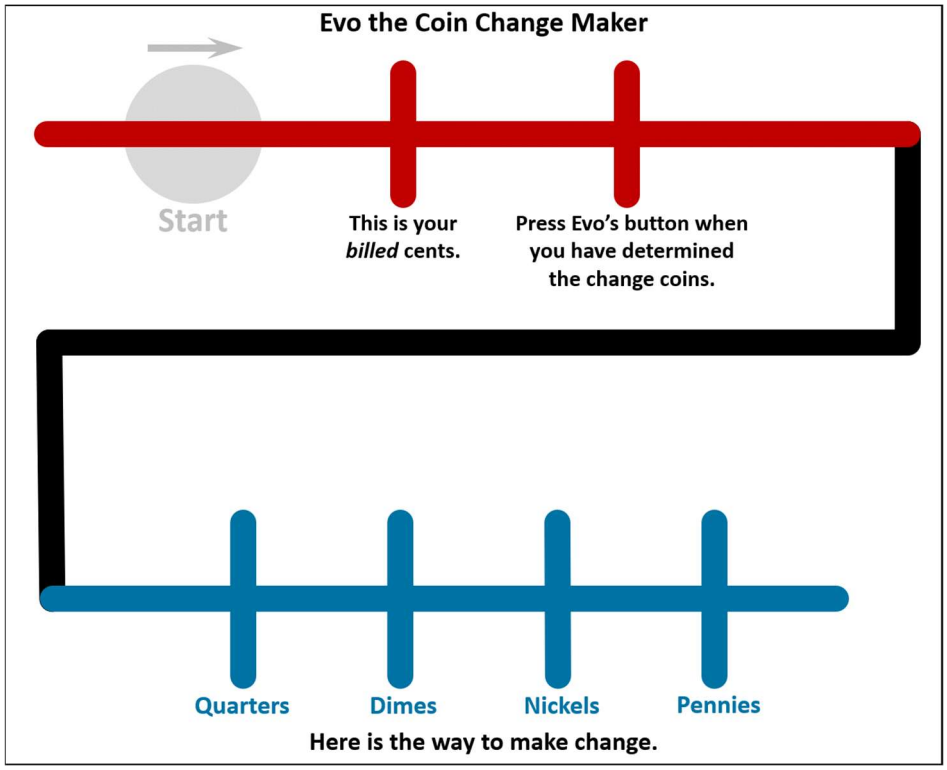


Figure 2

Food for Thought

You can program this challenge without the use of arrays. However, ***judicious*** use of two arrays as shown in Figure 3 and a loop to handle the coin change algorithm, can really challenge your understanding of array concepts and manipulation. It will also shorten the length of your OzoBlockly program. The array *coinDenominations* has the values 25, 20, 5, and 1 in array indices 0 through 3, respectively. The algorithm computes and stores the number of quarters, dimes, nickels, and pennies to be returned to the customer in array indices 0 through 3 of the *coinChange* array. ***Can you come up with the most elegant challenge solution in your class?***

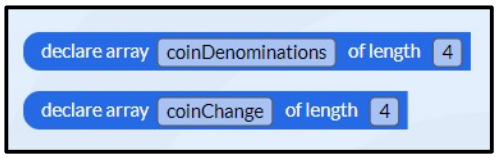
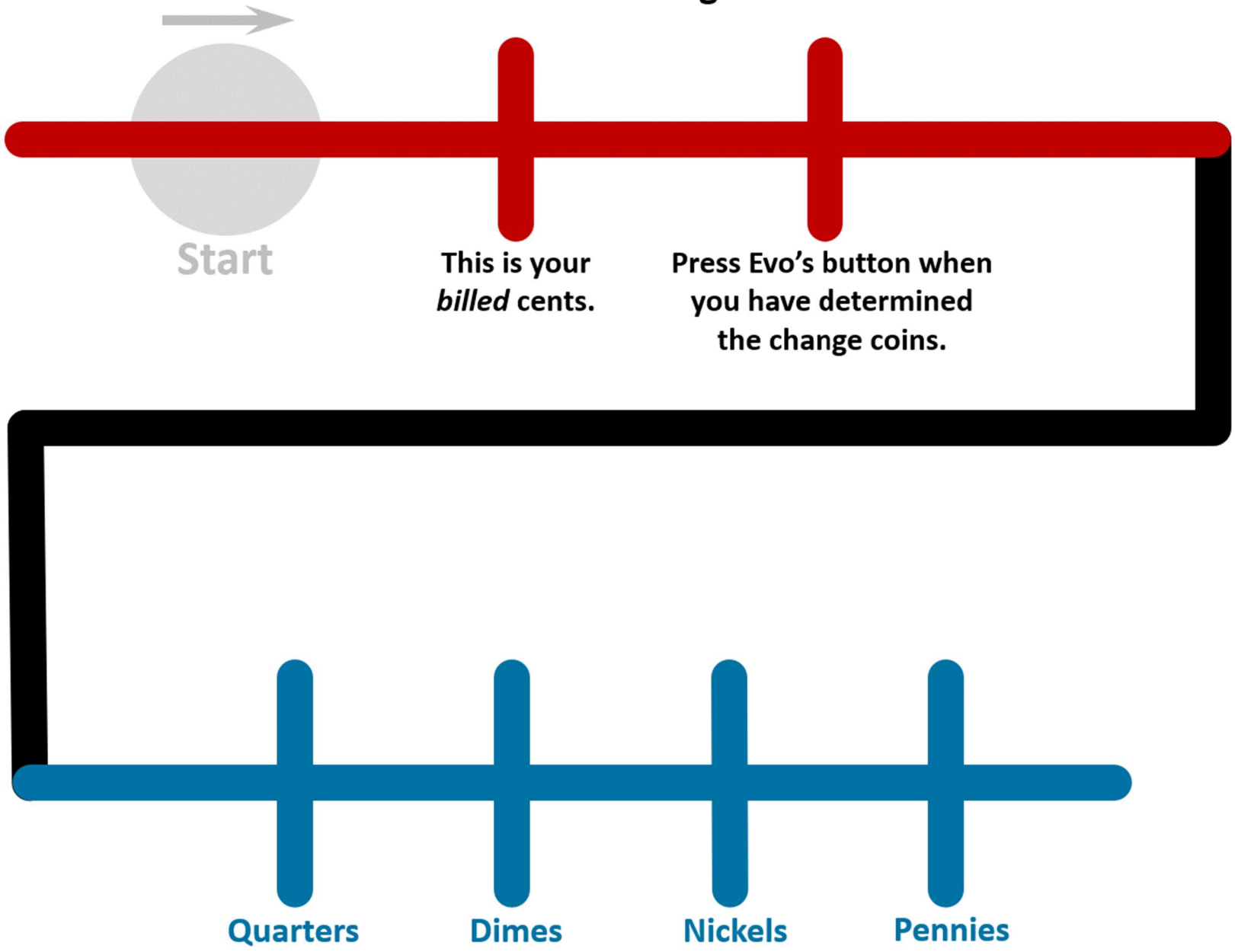


Figure 3

Evo the Coin Change Maker



Here is the way to make change.