CS 278 (Fall 2014) Lab 1

Hing Leung

For the lab session on August 27, 2014.

Submit the programs and the answers through CANVAS.

Note: for the programming tasks, you are asked to write the programs using Java.

Suppose we can buy a chocolate bar from the retail store for $1 each. Inside every chocolate bar, there is a coupon included. After collecting 10 coupons, we can redeem the 10 coupons for one chocolate bar. Again, a coupon is found in the chocolate bar redeemed. In other words, once you have started purchasing chocolate bars, you always have some coupons in your pocket.

Given n dollars, we consider the number of chocolate bars that we can consume with n dollars. For example, for 10 dollars, effectively we could have consumed 11 chocolate bars (after redeeming 10 coupons for one bar). Moreover, we would still have one coupon left in the pocket. Another example is that for 19 dollars, we could have consumed 21 chocolate bars and still have one coupon left.

We define `bars`(n) to be the number of chocolate bars that one can buy (and redeem) for n dollars. Thus, `bars`$(10) = 11$ and `bars`$(19) = 21$.

The function `bars` takes a nonnegative integer as argument and return a nonnegative integer as result. That is, the domains and codomains of `bars` function are the set of natural numbers $\{0, 1, 2, \ldots\}$.

To define a function, one way is give a mathematical formula for `bars`. It turns out that the formula for `bars` is quite challenging to find.

Another way is to define the function in a constructive way, but not aiming for a closed-form formula.

We define `bars` in terms of another function called `barsRedeem`. Suppose we have k coupons, `barsRedeem`(k) returns the number of chocolate bars that one can redeemed with k coupons. Examples: `barsRedeem`$(10) = 1$ and `barsRedeem`$(19) = 2$.

If the function `barsRedeem` has been defined, the function `bars` can easily

be defined as

$$\texttt{bars}(n) = n + \texttt{barsRedeem}(n)$$

as with n dollars, we can get n chocolate bars plus the number of chocolate bars that one can redeem with n coupons.

We define `barsRedeem` constructively by an inductive definition. When the number of coupons is less than 10, it is clear that no chocolate bars can be redeemed. Thus,

$$\texttt{barsRedeem}(k) = 0 \qquad \text{if } 0 \leq \text{k} < 10$$

Next, suppose we have defined `barsRedeem` for arguments $0, 1, 2, \ldots$ up till $k - 1$. We can define `barsRedeem`based on the previously defined values of `barsRedeem`(0), `barsRedeem`(1), `barsRedeem`(2), ..., `barsRedeem`$(k - 1)$ as follows:

$$\texttt{barsRedeem}(k) = 1 + \texttt{barsRedeem}(k - 10 + 1) \qquad \text{if k} \geq 10$$

since we can redeem one chocolate bar by spending 10 coupons and the remaining number of coupons is $k - 10 + 1$, as the chocolate bar redeemed also contains a coupon, with which we can redeem for more chocolate bars.

That is, from `barsRedeem`(1) = 0, we define `barsRedeem`(10) to be

$$1 + \texttt{barsRedeem}(10 - 10 + 1) = 1 + \texttt{barsRedeem}(1) = 1 + 0 = 1$$

Similarly, from `barsRedeem`(2) = 0, we define `barsRedeem`(11) to be

$$1 + \texttt{barsRedeem}(11 - 10 + 1) = 1 + \texttt{barsRedeem}(2) = 1 + 0 = 1$$

Also, from `barsRedeem`(10) = 1, we define `barsRedeem`(19) to be

$$1 + \texttt{barsRedeem}(19 - 10 + 1) = 1 + \texttt{barsRedeem}(10) = 1 + 1 = 2$$

That is, the function `barsRedeem` is defined inductively in a bottom-up manner.

Part 1:

1. Implement the functions `bars` and `barsRedeem` according to the inductive definitions above in Java as recursive programs.
2. What is `bars`(1000)? `bars`(1001)? `bars`(1002)? `bars`(1003)? `bars`(1004)? `bars`(1005)? `bars`(1006)? `bars`(1007)? `bars`(1008)? `bars`(1009)?

Part 2:

1. Implement again the function `bars` using loops (no recursion) in Java.
2. Do you get same answers for `bars`(1000), `bars`(1001), `bars`(1002), `bars`(1003), `bars`(1004), `bars`(1005), `bars`(1006), `bars`(1007), `bars`(1008), `bars`(1009) as in Part 1?

Part 3:

1. Develop a mathematical formula for `bars`(n) using +, -, *, / and the floor and/or ceiling functions.
2. Verify that your formula is correct by writing a program that test the answers given by the formula against the the answers returned by the program from Part 1.

   Note: It is not easy to come up with a correct formula. Give it a good try. If you cannot figure it out, just report your efforts/attempts, and move on to the next part of the lab.

   Note: If it is easy for you to find the correct formula, consider what the formula will be if there are three coupons in each chocolate bar, and 10 coupons are needed to redeem a chocolate bar.
3. Is your formula correct when $n = 0$?

Note: Given a real number x, the floor function $\lfloor x \rfloor$ returns the largest integer that is smaller than or equal to x, whereas the ceiling function $\lceil x \rceil$ returns the smallest integer that is larger than or equal to x. Examples: $\lfloor 3.2 \rfloor = 3$, $\lceil 3.2 \rceil = 4$, $\lfloor -3.2 \rfloor = -4$, $\lceil -3.2 \rceil = -3$, $\lfloor 3 \rfloor = 3$ and $\lceil 3 \rceil = 3$.

Next, we are interested in computing the average cost per chocolate bar. Since we can consume 11 chocolate bars for \$10 dollars, the average cost per chocolate bar is $10/11 = 0.9090909$ dollar. On the other hand, for \$19 dollars, we can consume 21 bars. In this case, the average cost per chocolate bar is $19/21 = 0.9047619$ dollar.

We define a function `AvgCost` as follows:

$$\texttt{AvgCost}(n) = n \ / \ \texttt{bars}(n)$$

Part 4:

1. Is the function `AvgCost` monotonically increasing? monotonically decreasing?
2. Is $\texttt{AvgCost}(n) > \texttt{AvgCost}(n+9)$ always true?
3. In general, what can you say how `AvgCost` changes as n increases?
4. Indeed, `AvgCost` approaches a limit c as n grows unbounded. what is the value of c?
5. Find the smallest value of n_0, such that $|\texttt{AvgCost}(n) - c| < 0.001$ for all $n \geq n_0$.
6. Find the smallest value of n_1, such that $|\texttt{AvgCost}(n) - c| < 0.0001$ for all $n \geq n_1$.
7. It is observed empirically from the calculations returned by a computer program that the average cost approaches the limit c. Can you give an explanation of why the average cost per chocolate bar is c (without referring to the empirical results)?

Suppose there is a kid who does not have \$1 to buy a chocolate bar. Instead, he has 50 cents and 6 coupons. In order to boost sales, a store owner decides selling one chocolate bar to the kid after taking the kid's 50 cents and 6 coupons.

Part 5:

1. Determine a fair dollar value for one coupon.

2. Has the store owner taken advantage of the kid? Did the kid pay more than what the chocolate bar is worth?

Note: Submit both the written answers and the Java programs.