SUPERMARKET SELF CHECKOUT

This topic illustrates *How to Invent an Algorithm* and *Help for Beginners*.

**Example**
Consider a self-checkout kiosk found in most supermarkets with which the shopper tallies his or her own purchases. If the shopper pays in cash then the machine dispenses any change back to the shopper in bills and coins. For example, if the shopper is owed $3.68 in change, the machine dispenses 3 $1-dollar bills, 2 quarters, 1 dime, 1 nickel and 3 pennies.

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
</table>
| 1    | Understand the problem by solving it | • Use pencil and paper to solve the problem by hand.  
• Use George Pólya’s problem solving techniques.  
• Write a list of any questions & or ambiguities you encounter and seek answers. |

**Dispense $3.68 in bills and coins**

<table>
<thead>
<tr>
<th>Step</th>
<th>Dispense</th>
<th>Value</th>
<th>Amount Remaining</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 $1 bills</td>
<td>$3.00</td>
<td>$3.68</td>
</tr>
<tr>
<td>1</td>
<td>2 quarters</td>
<td>0.50</td>
<td>0.68</td>
</tr>
<tr>
<td>2</td>
<td>1 dime</td>
<td>0.10</td>
<td>0.08</td>
</tr>
<tr>
<td>3</td>
<td>1 nickel</td>
<td>0.05</td>
<td>0.03</td>
</tr>
<tr>
<td>4</td>
<td>3 pennies</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>Total dispensed</td>
<td>3.68</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Questions and Answers**

Q: What bills and coins can be dispensed?
A: For simplicity, let’s keep it to $1 bills, quarters, dimes, nickels and pennies.
2. Determine the program’s requirements

- Identify the program’s input and output.
- Determine what computations transform the input into the output.

### Data for the Self Checkout Algorithm

1. The amount of change to dispense (e.g. $3.68)
2. The value of a $1 bill (i.e. $1.00)
3. The value of a quarter (i.e. 25¢)
4. The value of a dime (i.e. 10¢)
5. The value of a nickel (i.e. 5¢)
6. The value of a penny (i.e. 1¢)
7. The number of $1 bills making up the change
8. The number of quarters making up the change
9. The number of dimes making up the change
10. The number of nickels making up the change
11. The number of pennies making up the change
12. The amount of change remaining to be dispensed after each bill or coin is dispensed
<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Use top-down design to develop a pseudo-code algorithm that solves the problem</td>
<td>• Write down a first-draft algorithm as follows:&lt;br&gt;  o Write in English the steps you employed when solving the problem by hand.&lt;br&gt;  o Don’t worry about language or details; just get your ideas on paper in outline form.&lt;br&gt;  o Be sure to include the input and output steps.</td>
</tr>
</tbody>
</table>

**Self Checkout Algorithm (1st Draft)**

1. input the change to be dispensed
2. determine the number of $1 bills in the change
3. calculate the amount remaining after dispensing $1 bills
4. determine the number quarters in the amount remaining
5. update the amount remaining after dispensing quarters
6. determine the number of dimes in the amount remaining
7. update the amount remaining after dispensing dimes
8. determine the number of nickels in the amount remaining
9. update the amount remaining after dispensing nickels
10. determine the number of pennies in the amount remaining
11. output the number of $1 bills, quarters, dimes, nickels and pennies
**How to Invent an Algorithm – Help for Beginners**

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
</table>
| 3    | Use top-down design to develop a pseudo-code algorithm that solves the problem | • Introduce program variables to hold data you’ve identified.  
• Rewrite your algorithm accordingly. |

**Variables for the Self Checkout Algorithm**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Datum</th>
</tr>
</thead>
<tbody>
<tr>
<td>change</td>
<td>amount of change to be dispensed</td>
</tr>
<tr>
<td>dollars</td>
<td>number of $1 bills</td>
</tr>
<tr>
<td>remainder</td>
<td>amount remaining after dispensing bills or coins</td>
</tr>
<tr>
<td>quarters</td>
<td>number of quarters</td>
</tr>
<tr>
<td>dimes</td>
<td>number of dimes</td>
</tr>
<tr>
<td>nickels</td>
<td>number of nickels</td>
</tr>
<tr>
<td>pennies</td>
<td>number of pennies</td>
</tr>
</tbody>
</table>

**Self Checkout Algorithm (2nd Draft)**

1. input change  
2. determine dollars from change  
3. calculate remainder by deducting value of dispensed dollars from change  
4. determine quarters from remainder  
5. update remainder by deducting value of dispensed quarters  
6. determine dimes from remainder  
7. update remainder by deducting value of dispensed dimes  
8. determine nickels from remainder  
9. update remainder by deducting value of dispensed nickels  
10. determine pennies from remainder  
11. output quarters, dimes and nickels
## How to Invent an Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Check that your algorithm is correct and complete</td>
<td>Desk check the algorithm after each refinement step.</td>
</tr>
</tbody>
</table>

The following table shows the action performed by each step along with the resulting values of the variables. The algorithm’s computations match those of the pencil and paper solution, so it appears to be correct.

### Desk Check of Self Checkout Algorithm 2nd Draft

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>input change</td>
<td>change</td>
</tr>
<tr>
<td>2.</td>
<td>determine dollars</td>
<td>dollars</td>
</tr>
<tr>
<td>3.</td>
<td>calculate remainder</td>
<td>remainder</td>
</tr>
<tr>
<td>4.</td>
<td>determine quarters</td>
<td>quarters</td>
</tr>
<tr>
<td>5.</td>
<td>update remainder</td>
<td>remainder</td>
</tr>
<tr>
<td>6.</td>
<td>determine dimes</td>
<td>dimes</td>
</tr>
<tr>
<td>7.</td>
<td>update remainder</td>
<td>remainder</td>
</tr>
<tr>
<td>8.</td>
<td>determine nickels</td>
<td>nickels</td>
</tr>
<tr>
<td>9.</td>
<td>update remainder</td>
<td>remainder</td>
</tr>
<tr>
<td>10.</td>
<td>determine pennies</td>
<td>pennies</td>
</tr>
<tr>
<td>11.</td>
<td>output dollars, quarters, dimes, nickels and pennies</td>
<td></td>
</tr>
</tbody>
</table>

Supermarket Self Checkout
## How to Invent an Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Use top-down design to develop a pseudo-code algorithm that solves the problem</td>
<td>To “refine some part of the algorithm” use <strong>what-to-how-conversion</strong> or <strong>decomposition</strong>.</td>
</tr>
</tbody>
</table>

### Step 2: Determine Dollars from Change

#### What?

2. determine dollars from change

#### How?

Given any amount of change, the number of $1 bills is the whole part of the number.

- 3 $1 bills make up $3.68
- 2 $1 bills make up $2.50

We need to strip the fractional part of the change: $3.68 ➔ 3$

This is done with the mathematical operation “floor,” written \([\ ]\).

#### Pseudo-code

2. dollars = \([\text{change}]\)

### Step 3: Calculate Remainder by Deducting Value of Dispensed Dollars from Change

#### What?

3. calculate remainder by deducting value of dispensed dollars from change

#### How?

Once bills have been dispensed you must subtract their total value from the amount of money to be given back to the customer.

The total value of the bills is the number of bills times the value of each, which is $1.00.

$\$3.68 \text{ owing and 3 $1 bills dispensed leaves } 3.68 - 3.00 = 0.68 \text{ owing}$

#### Pseudo-code

3. remainder = change - (dollars × 1.00)
### What? 4. determine quarters from remainder

### How?
The number of quarters that “goes into” a certain amount of money is the same as the number of times the 25 “goes into” the amount. This is division. Thus, dividing the amount by 0.25 gives the number of quarters.

- 30¢ contains 1 quarter and \(0.30 \div 0.25 = 1.2\)
- 60¢ contains 2 quarters and \(0.60 \div 0.25 = 2.4\)
- 95¢ contains 3 quarters and \(0.95 \div 0.25 = 3.8\)

We still need to strip the fractional part (3.8 → 3) using the mathematical “floor” method \([\ ]\).

### Pseudo-code
4. quarters = \(\lfloor \text{remainder}/0.25 \rfloor\)

### What? 5. update remainder by deducting value of dispensed quarters

### How?
This is similar to step 3 except that (1) it is updating the variable remainder instead of calculating a new variable and (2) the value of quarters dispensed is the number of quarters times $0.25.

### Pseudo-code
5. remainder = remainder − (quarters × 0.25)

Steps 6 and 8 are similar to step 4; steps 7 and 9 are similar to step 5.

### What? 10. determine pennies from remainder

### How?
At this point in the processing, the remainder is some value from 0.01 to 0.04 because all coins worth $0.05 and above have been removed. To convert this to the number of pennies, multiply it by 100.

### Pseudo-code
10. pennies = remainder × 100
Self Checkout Algorithm (3rd Draft)

1. input change
2. dollars = \lfloor change \rfloor
3. remainder = change – (dollars \times 1.00)
4. quarters = \lfloor remainder/0.25 \rfloor
5. remainder = remainder – (quarters \times 0.25)
6. dimes = \lfloor remainder/0.10 \rfloor
7. remainder = remainder – (dimes \times 0.10)
8. nickels = \lfloor remainder/0.05 \rfloor
9. remainder = remainder – (nickels \times 0.05)
10. pennies = remainder \times 100
11. output dollars, quarters, dimes, nickels and pennies
Another desk check will verify that these refinements are correct.

<table>
<thead>
<tr>
<th>Step</th>
<th>What?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Check that your algorithm is correct and complete</td>
<td>• Desk check the algorithm after each refinement step.</td>
</tr>
</tbody>
</table>

The algorithm’s computations match those of the pencil and paper solution, so it appears to be correct.