CSC 7003 : Basics of Software Engineering

J Paul Gibson, A207

paul.gibson@telecom-sudparis.eu

http://www-public.telecom-sudparis.eu/~gibson/Teaching/CSC7003/

The Balance Problem : Sample Solution

/~gibson/Teaching/CSC7003/L2-TheBalanceProblem-SampleSolution.pdf
The Ternary Weight System

A simple class to weigh items on a balance with 2 cups - a given integer value using a ternary weight set:
1, 3, 9, 27, 81, 243, ...

Input (on the command line) should be a valid integer value.
If there is no valid integer value input on the command line then the default value of 100 will be used.

The output will be a text string on system.out of the form:

To weigh 100 in right cup of balance, one needs to place the ternary weights in the left (L) and right (R) cups as follows:

L : 11
R : 2
L : 1

This is to represent the balance in the state:

\( 81, 27, 1/ \quad \ \ 100, 9/ \)

\( 1 \quad \ \ 1 \)

TO DO: Develop this application (in whatever language you wish) and demonstrate your best software engineering techniques/skills.
The Balanced Ternary System

Donald Knuth:
“Perhaps the prettiest number system of all is balanced ternary.”

Some **problem analysis**
(secondary sources):

Solution C/C++

```c++
#include <iostream>
#include <stdlib.h>
#include <LIMITS.H>
using namespace std;

char flip(char side){
    if (side == 'L') return 'R'; else return 'L';}

void split(int target, char side){
    if (target ==0) return;
    int power3 =1;
    while (power3<target) power3=power3*3;
    if (target == power3) {cout <<side<<": "<<target;return;}
    if (target <= power3/2)
        {cout<<side<<": "<<power3/3<<endl; split(target-power3/3, side);}
    else  {cout<<side<<": "<<power3<<endl; split(power3-target, flip(side));}
}

int main(int argc, char* argv[]){
    int target;
    if (argc <2) target = 100; else target = atoi(argv[1]);
    if (target <1 || target > INT_MAX /2 ) target = 100;

    cout <<"To weigh " << target <<" in right cup of balance,";
    cout <<"one needs to place the ternary weights in the left (L) and right (R) cups as follows:\n";
    split(target, 'L');
}
```

This shows my programming skills but not necessarily my software engineering skills
Solution C/C++

Is this solution acceptable?

• How (easy) to compile/make?
• How (easy) to execute?
• How (easy) to test?
• How (easy) to understand?
• How (easy) to maintain/improve?
• How (easy) to re-use?

```c
$ g++ -o balance.exe balance.cc
gibson@PAT9106 ~/balanceCode
$ ./balance
To weigh 100 in right cup of balance, one needs to place the ternary weights in the left (L) and right (R) cups as follows:
L: 81
R: 9
L: 1

$ ./balance 40
To weigh 40 in right cup of balance, one needs to place the ternary weights in the left (L) and right (R) cups as follows:
L: 27
L: 3
L: 1

$ ./balance 2147483647
To weigh 100 in right cup of balance, one needs to place the ternary weights in the left (L) and right (R) cups as follows:
L: 81
R: 9
L: 1
```
public class Balance
{

static char flip (char side) {
    if (side == 'L') return 'R';
    else return 'L';
}

static void split (int target, char side) {
    if (target == 0) return;

    int power3 = 1;
    while (power3 < target) {power3 = power3 * 3;}

    if (target == power3) {System.out.println(side + " : " + target); return;}
    if (target <= power3 / 2) {System.out.println(side + " : " + power3 / 3);
        split(target - power3 / 3, side); return;}
    else {System.out.println(side + " : " + power3);
        split(power3 - target, flip(side)); return;}
}

Re(verse) engineering
The *same* solution (in Java)

```java
public static void main (String [] args){

    int target = 100; // default test value

    if (args.length > 0)
        try{target = Integer.parseInt(args[0]);}
            catch (NumberFormatException exc){target = 100;}

    if (target > Integer.MAX_VALUE/2) target = 100;

    System.out.print("To weigh "+target+" in right cup of balance, one needs to place the ternary weights in the ");
    System.out.println("left (L) and right (R) cups as follows - ");
    split(target, 'L');

}
```

```
To weigh 100 in right cup of balance, one needs to place the ternary weights in the left (L) and right (R) cups as follows
L : 11
  : 27
R : 9
L : 1
```

```
To weigh 40 in right cup of balance, one needs to place the ternary weights in the left (L) and right (R) cups as follows -
L : 27
  : 9
L : 3
L : 1
```
The *same* solution (in Java)

Is this solution acceptable?

- How (easy) to compile/make?
- How (easy) to execute?
- How (easy) to test?
- How (easy) to understand?
- How (easy) to maintain?
- How (easy) to re-use?

Did changing language make any difference to these issues?
A software engineering solution

Functional correctness is important but there are other issues:

• How to compile/make? … should be as simple as possible (with as few dependencies/requirements as possible)

• How to execute? … should be as simple as possible

• How to test? … should be automated and ‘of high quality’

• How to understand? … should be documented and ‘of high quality’

• How to maintain? … should be documented and well-structured/designated

• How to re-use? … should be correct and documented

Did changing language make any difference to these issues?
A better solution: WeighingAndBalancing.zip

QUESTIONS:
What design decisions did I make?

Is all this extra work worth the effort?

What could be improved?
I followed a **process**, and I used **tools** to help support the process


**IDE (Eclipse + plugins)** –
editor, compiler, debugger, profiler, version control

**Documentation – Javadocs**

**Testing – JUnit**

**Design – OO (UML)**

**Implementation - Java**
Typical Working Screenshot of a Software Engineer

Let’s Experiment Together With The ‘Solution’ (for a few minutes)