MA305 – Lab 0.
Finite Precision Arithmetic Woes

Due: 09/06/2018

This is a sobering example (by W. Kahan) to serve as a warning of how inaccurate finite precision arithmetic can be. It is meant to scare you away from trusting blindly anything produced by a computer, and make you aware that strange things can happen easily, that what you get DOES depend on how you write the expressions (and even the constants!), and that machine arithmetic is NOT infinite precision arithmetic! Below is a code that performs some very simple calculations:

```python
#!/usr/bin/env python
"""
=======================================================================
MA305 - Lab 0: your name - date
Purpose: To implement W. Kahan's example of finite precision arithmetic
failings from C. Van Loan, Intro. to Computational Science, 1995; p.xxiv
-----------------------------------------------------------------------
"""
h = 1/2
x = 2/3 - h
y = 3/5 - h
u = (x + x + x) - h
v = (y + y + y + y + y) - h
q = u / v

print()
print('Results from the program:')
print('h :', h)
print('x :', x)
print('y :', y)
print('u :', u)
print('v :', v)
print('q :', q)
print()
```

1. Copy the formulas on paper (by hand). Perform the calculations by hand exactly (as fractions) and record the (exact) results.
2. Now do it on the machine. Move the mouse to an xterm and create a new directory "MA305/Lab0", cd into it and type the code into a file "lab0.py". Do you remember how?
`$ mkdir MA305` (skip this step if you had already created the MA305 directory)
```
$ cd MA305
$ mkdir Lab0
$ cd Lab0
$ vi lab0.py
```
and type it in. Save and exit from vi with `ZZ`, or better yet, save (`:w`) without exiting (since you may need to edit it again), and go to another xterm for running it. Make sure you are in the Lab0 directory again!
3. Run the Python code "lab0.py':
```
$ python lab0.py
```

4. Compare the computed values with the exact values you found by hand. Which ones agree and which ones do not? Any wild guess as to what may have caused the discrepancy?
5. Now let's compile and run the C and Fortran codes "lab0.c" and "lab0.f" posted at your course Canvas.
C:
`$ gcc lab0.c -o c.x` (the executable will be named c.x instead of the default a.out)
`$ ./c.x`
Fortran:
`$ gfortran lab0.f -o f.x` (produces the executable f.x)
`$ ./f.x`
6. Did you get different values now? Which code (C, Fortran or Python) is more accurate?

7. Now let's try double precision arithmetic in the Fortran program "lab0.f".
`$ cp lab0.f lab0dp.f` (what does this do?)
Open the file "lab0dp.f" for editing
`$ vi lab0dp.f`
and make the following modifications:
replace `real` → `double precision`
replace `.0` → `.d0` in all constants (double precision constants)
`:wq` (save it and quit)
Compile and run:
`$ gfortran lab0dp.f -o fdp.x`
`$ ./fdp.x`
8. Did you get different values now? Is the double precision calculations in Fortran more correct than the single precision? Any wild guess as to what's happening?
9. Send us email using the `mail` command with short answers to the questions 4, 6, and 8. In CW0 you already created a mail alias "305". You can email us (and you) simply by typing: "mail 305"
`$ mail 305`

Please ALWAYS enter a Subject: `305:lab0`, hit **Enter** then type the body of the message. Terminate with . (dot) on a new line.
10. Submit your Python code 'lab0.py' thru Canvas.
