MA305 – Lab #0
Finite Precision Arithmetic Woes
Due on: 09/15/2023

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:

```
#!/usr/bin/env python3
"""
================================================================================
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/Labs/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 Labs`
`$ cd Labs`
`$ 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 Terminal for running it. Make sure you are in the Lab0 directory again!

3. Run the Python code "lab0.py':

```
$ python3 lab0.py
```

You can also run the Python code "lab0.py" like a shell script (see the first line of the code!). Set the execution permission to the file and run it.

```
$ chmod u+x lab0.py
$ ./lab0.py
$ ./lab0.py > out.txt                    (this will redirect screen output to the file "out.txt")
$ cat out.txt                                        (see the results in the file "out.txt")
```

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. Download the file "lab0-files.zip" from Canvas Course Documents page, save it in the appropriate working directory and unzip it.

```
$ mv ~/Downloads/lab0-files.zip ~/MA305/Labs/Lab0/
$ cd ~/MA305/Labs/Lab0                   (make sure you are in the working directory for Lab0)
$ ls                                        (list the files to make sure "lab0-files.zip" is here)
$ unzip lab0-files.zip                                                        (unzip the file)
$ ls                                          (make sure "lab0.c" and "lab0.f" are here)
```

Now let's compile and run the C and Fortran codes "lab0.c" and "lab0.f". To run these codes, you need to login to `wxsession` server, first. In an `xterm`, type:

```
$ ssh you@wxsession.db.erau.edu
$ cd ~/MA305/Labs/Lab0                   (make sure you are in the working directory for Lab0)
$ ls                                          (make sure "lab0.c" and "lab0.f" are here)
```

Compile and run C code:

```
$ gcc lab0.c -o c.x (the executable will be named c.x instead of the default a.out)
$ ./c.x
```

Compile and run Fortran code:

```
$ 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. Python code by default computes in double precision arithmetic (taking care of 15 decimal digits). In C and Fortran, we should declare the variable types (float/double, real/double precision etc.) in advance. The fortran code is working in single precision (considering only 7 decimal digits). Now let's try it in double precision.

```
$ cp lab0.f lab0_dp.f                          (copy the file "lab0.f" to " lab0_dp.f")
$ vi lab0_dp.f                                  (open the file "lab0_dp.f" for editing)
```

remove ! in line 10, and type ! at the beginning of line 9 (comment/uncomment)

replace `1.0, 2.0, 3.0` in line 12-14 → `1.d0, 2.d0, 3.d0` (double precision constants)

`:wq` (save it and quit)

```
$ gfortran lab0_dp.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?

9. Now rename (`mv`) the file "out.txt" produced in **3** to "lab0_answers.txt", open it with `vi`

```
$ mv out.txt lab0_answers.txt
$ vi lab0_answers.txt
```

type short answers to the questions in **4**, **6**, and **8** at the end of the file, save it and quit.

```
:wq
```

10. Send us the file "lab0_answers.txt" using the `mail` command as shown below and submit your Python code 'lab0.py' thru Canvas.

```
$ ssh you@wxsession.db.erau.edu          (you don't need to do this if you are in wxsession)
$ cd ~/MA305/Labs/Lab0                   (make sure you are in the working directory for Lab0)
$ ls                                     (make sure the file "lab0_answers.txt" is here)
$ mail -s "305-lab0" 305 < lab0_answers.txt                                   (email the file)
```

Note: Please check your email. You should get a copy of the email with the file "lab0_answers.txt" and Subject: "305-lab0". If you did not get the email, please let me/TA know, we will help you fix the `mail alias for 305`.