Assignment 6

Aman Kikani, Sean Kenny, Gavin Fors

December 14, 2024

1 Introduction

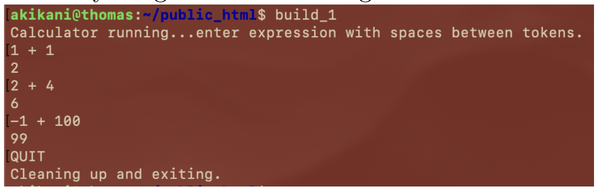
For the course of the assignment, we were tasked to test and asses the functionality of both the build_1 and build_2. Both of these files are soley command line but they vary slightly in their functionality.

2 Usability of File 1

File 1 (build1_1) is a functional file but it lacks most of the QOL features like telling people how to input data, what the supported functions are, how to easily quit the program and many others. Overall, this isnt very usable and doesnt take into account any of the user experience

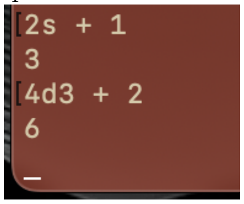
3 Testing File 1

File was (called "build_1") is a command line file which works as a basic and rudimentary calculator. Testing this file consisted of looking at and entering basic/valid data to check the outputs. On first glance everything seems to be alright as seen below



```
[akikani@thomas:~/public_html$ build_1
Calculator running...enter expression with spaces between tokens.
[1 + 1
2
[2 + 4
6
[-1 + 100
99
[QUIT
Cleaning up and exiting.
```

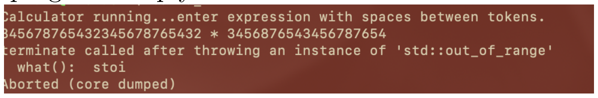
But, when further testing is conducted, the calclator functions reveal their lack of real functionality. When dividing, the output only comes as an integer, all decimals are lost. Another thing thay has presented itself as an issue is that the inputs, if given a number, letter, then number, will output the computation of first and last number.



```
[2s + 1
3
[4d3 + 2
6

_
```

Another flaw which was found in the first build file was that when multiplying very large numbers, the file and build simply collapsed and ended. Instead of expetion handling the issue like divide by 0, the program simply breaks and exits



```
Calculator running...enter expression with spaces between tokens.
345678765432345678765432 * 3456876543456787654
terminate called after throwing an instance of 'std::out_of_range'
  what():  stoi
Aborted (core dumped)
```

Besides those flaws, build one works mostly as expected

4 Usability of File 2

File 2 was a much better and nicer to use program. It had more instructions on what to do and the comptation output is also much more readable. When running, the program will tell you the issue its running into if it has one as well as still trying to compute the computation.

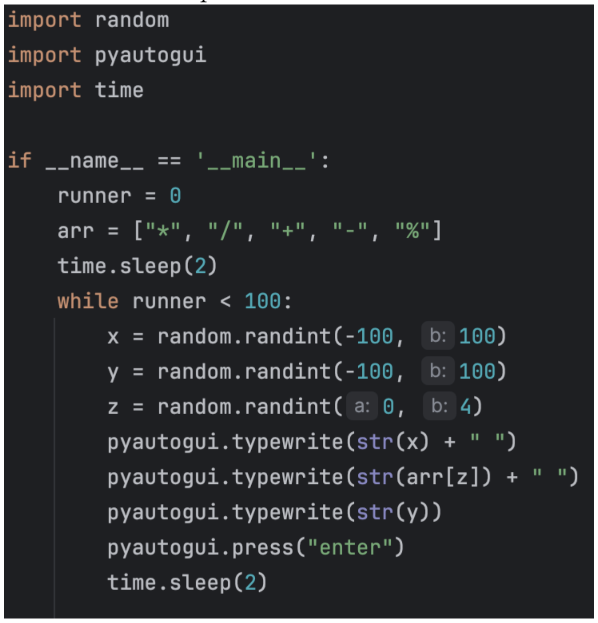
5 Testing of File 2

When testing file 2, all programs ran as expected except for the division once again. Otherwise everything worked as intended. When the same test was run for the large number multiplication, instead of vreaking the program, it simply outputs it into mutliple differnt lines. As well as when you enter in invalid data, it tells you the error and still down notfail

6 Reflections

Sean: System Testing is a type of black box testing that looks at a system from a very high level. System testing explores how different elements of a program interact with each other and validates the system by comparing the program to the program requirements. I was helping to create test cases for the calculator. In order to create the test cases, we looked to the program specifications to see what different areas we needed to ensure complied with the requirements. We created test cases to test the various operators such as addition, subtraction, multiplication, division, and modulus. We also knew that the program could handle both positive and negative integers so we created test cases for those, we also had to make sure that the program would loop until the user typed QUIT. Creating the different test cases reminded me of our BVA assignment but a lot more enjoyable and much less repetitive.

Aman: Here in this assignment I worked on the actual testing and looked for the difference cases which would break the file. When testing we had to find a way to input in tons of variables so what I did was code a python file that generated random numbers and expressions over and over again while waiting slowly. This allowed us to test a multitude of values. I also took the liberty of writing this document in LaTex for the extra credit since I had a little bit of experience from using it in my MA 200 Introduction to proofs class



```
import random
import pyautogui
import time

if __name__ == '__main__':
    runner = 0
    arr = ["*", "/", "+", "-", "%"]
    time.sleep(2)
    while runner < 100:
        x = random.randint(-100,  b: 100)
        y = random.randint(-100,  b: 100)
        z = random.randint( a: 0,  b: 4)
        pyautogui.typewrite(str(x) + " ")
        pyautogui.typewrite(str(arr[z]) + " ")
        pyautogui.typewrite(str(y))
        pyautogui.press("enter")
        time.sleep(2)
```

Gavin: I worked with Sean on the devlopment of the different test cases and how the program shoud perform. One of the most important aspects in this assignment was to make sure that when we were making the test cases, we also had the right answer. This is critical because when we are testing a blackbox system like this, we dont know where errors are going to occur. I also helped a bit with the design and text of the LaTex file but since this is a smaller assignment there is quite a lot less to do.