Thicccque
October 14, 2018
Dr. Parham-Mocello

Assignment 2 Part 1

Orientation:

1. Ethical dilemmas in the field of computer science:
   - Privacy vs. Security:

     An ethical dilemma in society today, even apart from computer science, is the question of to what extent do we have a right to privacy as citizens. Social and politcal philosiphers have long considered this issue to be a trade-off between security of our persons and the privacy of our lives and data. Especially in a post-9/11 United States where political culture is often characterized by somewhat irrational fear of the 'other' or the amorphous threat of wrong-doers, people hotly debate where this line should be drawn in each sector it effects. For example, support of TSA has been dwindling over time as public opinion has drifted further into the realm that the organization operates more as useless "security theatre" than anything else, but the idea behind the TSA is still very much accepted in the United States. People have agreed over time that it is okay to wave your rights to privacy in select circumstances to promote the general security of *some* locations. When entering public buildings, court-houses, some museums, the airport, and other similar hotspots of activity, people generally don't scoff at securtity measures to ensure weapons aren't finding there way into the facilities. Where the general populace starts to balk at this tradeoff of increased security for waving privacy is when it starts to occur outside these socially acceptable locations to be tested. When people think they are being surveiled out in public or when people feel their personal information might be intercepted, they begin to grumble because the level of inconvenience for them is perceived as too high, because they are innocent. This dilemma of privacy vs security afftects all people living in a country capable of surveilance and especially everyone living in the United States today.

     The national and international issue of this tradeoff is even more prevelant when it comes to the internet, the field of computing, and in hacking communities like those that gather at DEF CON, for example. The internet, from its inception, was open and free in many ways. Although it had a barrier to entry, it quickly became a place where any person could find refuge and commence in activities both legal and illegal. Computer science, as a field, is very closely tied to this issue because the decisions computer scientists make and the programs that programmers and companies put together must adhere to law, try to please a customer base that, depending on how deep into internet communities they are, will either not bat an eye at data extraction and usage or boycott the product and every subsequent product from that company. Tools like Tor have become extraordinarily popular and widely used (and more recently the huge boom in wide adoption of VPN clients too) because people don't want to feel like they're being watched. These companies need to be very aware of what they're doing with user data and that there is a general distrust of larger companies or any companies that associate with the government. Where I personally stand on this issue is pretty mixed. I don't personally have much to hide, but also I do believe that the government and companies I don't give permission to use my data shouldn't have access to my personal files, data, or

history. I believe the best way to adapt the legal systems of the world to this internet age would be to embrace the concept of digital warrants, where people can be held accountable if they retrieve evidence, damning or otherwise, by unjust means. I believe that my data (online or locally stored) should never be touched by anyone for any reason unless there is a reasonable suspicion that I had broken a law in my country or broken a statute of an institution I had signed some agreement to attend. That is my belief and my preferred solution. It is an interesting question though, just because the internet is not owned and should never be owned by any one organization or government (*especially ISPs, thank you very much*) I shouldn't feel like I need to, but I'm often tempted to use a Tor browser or premium VPN to feel like my private life remains that way.

- Responsibility of developer to prevent misuse or bug-fix post launch:

  This is a less complex dilemma, but a dilemma nonetheless. The first half of this dilemma is what the responsibility of a developer is to prevent misuse of their product. If a computer-scientists develops an alternate interface for launching video games, for example, and this tool operates by just ignoring the DRM protecting these files, then the tool could be legally used by users that legally purchased the games they are running, but this tool could also be used to run illegally downloaded games or games the user no longer owns a license for. The question is, what is the responsibility of the developer to prevent people from using their software in this way. Is it enough to make a public statement that the developer does not support usage of their tool for this purpose, or should they implement some catch to limit its usage in this way. Similarly, when it comes to bug-fixing, what is the responsibility of the developer to fix issues that arise after launch of a project? Is the user entitled to a product that work in all cases like they may have expected when purchasing, or is the user entitled to only what they spend for, and future development should necessitate future charges? I think that I have a pretty clear stance on these issues, and that stance is that it depends. When it comes to preventing misuse of a product, I think their is no obligation to put extra work into preventing misuse, but morally and ethically, it is the better thing to do. The developer doens't even need to make it foolproof. Simply offering a slight deterrent or making it slightly harder to use the software for illegal or unintended purposes will go a long way in limiting this misuse. The solution to this dilemma is to keep the industry standard of offering a deterrent, at a minimum, if not an outright blockage of this use. When it comes to post-launch bug-fixing, it really depends on how the service or product is marketed to the user. If the product is clearly marketed as being a tool that will "clean your device of all viruses," for example, but there is a bug that allows a certain major kind of malware through the shield, my standpoint is that the developer has a ethical obligation to plug that hole, because their users spent money on a product they did not receive. Does this mean I think a whole team needs to be dedicated to constantly plugging every minute hole that might appear in a tool indefinitely? No. The solution to this is again very clear. Tell your customers *exactly* what they're spending their good money on. Pretty simple. Be good people and you can still make money. You can even base your business of the idea that you are good people.

- The usage of user data / turning users into products:

  This dilemma is very similar to the first dilemma of privacy vs security, and especially because it concerns every single person who operates in an online space, whether or not they accept that it does. With many of the technological giants of today like Google, Facebook, and YouTube (yes I'm treating as seperate from Google), their product is not necessarily the service they provide the user. Rather, when using the services, the user is supposedly using it knowing that the product that the company offers is not the service, but the user themself (and the profile and history of the user on the platform) that the companies sell to advertisers. The companies are in the trade of advertisement and the trade of users' data to offer advertisers a tailored user for their content, at a marked up price. Google doesn't technically do this, because they both own the users and the system for getting them ads, so their product is more accurately a system that advertisers buy into to be considered for the most visible advertising slots on the internet. Facebook does trade in the data of users in a roundabout way though, and similarly attracts advertisers with their huge web-traffic. These companies aren't like all computing or computer-science based companies, but they started off in very similar spots. This dilemma specifically applies to the field of computer science because every company that operates in the space needs to make the decision of what to do with their users data, whether to store it for internal use, or whether to use it to increase profit externally. Although companies can't do this without user knowledge, there are ways to "inform the user" of this tracking that legally does but in actuality makes only a fraction of users aware (namely the terms and conditions document that users rarely actually read through). I personally believe that the data of a user isn't their own and that the companies have a right to do with it what they will, within reason, but that is probably only true because I'm careful with my information already. For a random United States resident logging on to Google, they might not have even considered that their information might be logged in any way. Ethically, companies should make it clear what is happening and offer users a way to opt out, but in my personal opinion, I don't mind being a product, because I go into each user-service relationship knowing full-well that what I enter could be used. Also, collecting user data for internal use to better systems is a net-positive in my estimation, unless that data could be compromised on a local or cloud level. The solution is always better communication or for users to take it as normal that a free service needs to make money somehow.

Computation:

2. Problem Analysis:
    a. Understanding the Problem:

I believe I understand most if not all of what is required of me to solve this problem. I do not fully understand to what extent we may use or not use any built-in functions. The problems states not to use them for conversion, but can I use them for manipulating lists that are already converted, but in reverse order, for example? The problem says my user can enter any unsigned decimal value and that programmer mode converts decimal to binary, with binary to decimal conversion as an extra-credit opportunity, but what if I want to allow them to do more than just that conversion? What if I want to let them go either direction or convert between bases other than 2 and 10? I think I understand what the problem means by "you must handle any integer value that is not positive," but still, what exactly is meant by "handle?" Does that just mean the program accounts for that possibility? For scientific mode, the problem requires that the user be able to choose from the legal operators and provide two operands. Do I have to get them as individual inputs in that order? Again, I must "handle" illegal operators, but what exactly does that mean? The problem then states that I must ask the user if they want to keep doing calculations "at the end of each iteration," but does that refer to programmer mode, scientific mode, or both? Does asking them if they want to go to a different mode require taking them back to some main menu? That's all I could pick apart for questions. Now for assumptions and requirements. First of all, I am going to need to require my user to choose somehow (with some input style) their mode...

General functional requirements:

- Way to direct user to each mode upon running program / after quitting mode.
- All interaction done in terminal.
- I'll use some starting menu to handle mode selection.
    - Exiting mode will take user back to menu.

User requirements in programmer mode: (if I let them convert any base)

- Number to convert.
- Base of that number.
- Desired base for resulting number.
- After conversion, whether they want to convert a new number.

Functional requirements in programmer mode:

- Take inputs from user.
- Convert inputs to integers.
- Way to repeat process if they want to do it again.
- Way to convert between bases.
- Eliminate leading zeros.
- Handle non-positive inputs (might as well handle everything).
- Give back result of conversion.

Assumptions in programmer mode:

- User will input only integers.
- Base values range from 2-10 so I don't need to go beyond decimal representation.

- User only needs option to leave at end.
- Requirement they start only in decimal and convert only to binary is a suggestion not a limit to functionality.
- I'm gonna make it look shiny.

User requirements in scientific mode:

- First operand.
- Second operand.
- Operator to use.
- Whether they're done after each iteration.

Functional requirements in scientific mode:

- Take inputs from user as single line or seperate inputs.
- Either split single input into operands and operator or treat some inputs as numbers and some as strings.
- Interpret selection of operator (if I allow direct input as string rather than choosing options from a list).
- Allow inputs to be floats or integers.
- Repeat process as many times as user wants.
- Give user option to leave at each iteration.
- Handle selection or input of bad operators (might as well handle everything).
- Give back result of operation.

Assumptions in scientific mode:

- User will input only integers or floats.
- User needs option to leave only after an operation is completed.
- Problem phrasing of "get operator, then operands" is a suggestion rather than a limit to functionality.
- I'm a try-hard and I'm gonna go hard.

b. Inputs, outputs, etc:

Inputs:

- Which mode to use.
- Number to convert.
- Base of that number.
- Desired base for resulting number.
- Whether they want to repeat programmer conversion.
- First operand.
- Second operand.
- Operator to use.
- Whether they want to repeat scientific calculations.

Outputs:

- Result of base conversion.
- Result of each operation.

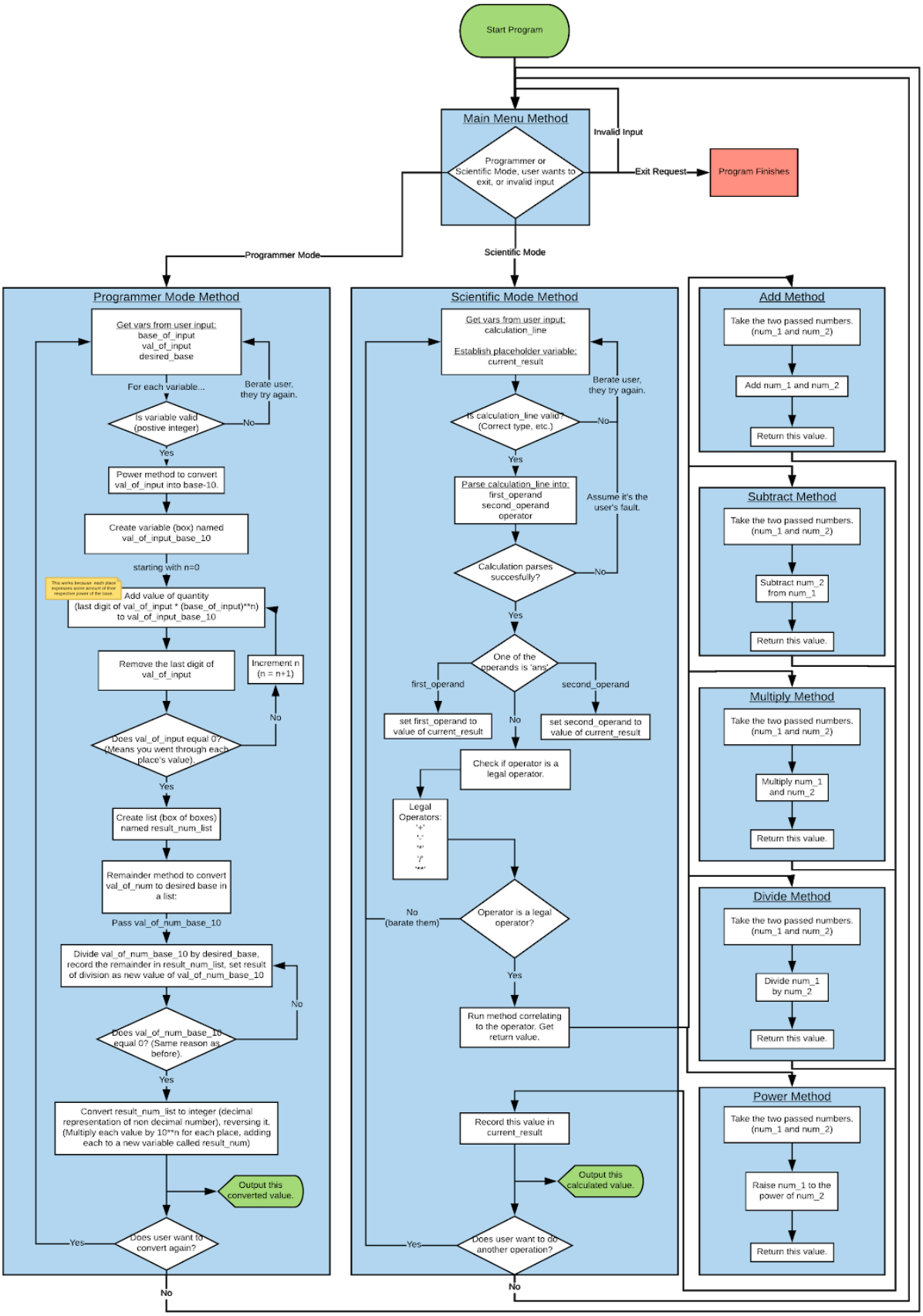
3. Program Design:
    a. Devising Plan: (What are the decisions that need to be made in this program? How are you going to calculate the binary number? Are you going to calculate this starting exponent? How are you going to handle bad input?)

        First of all, when it comes to bad input, I'll use a try/except/else setup, where the code in the 'try block' is attempted, and if an error results, the code in the 'except block' is executed instead. The 'else block' is executed if the 'try block' completes with no errors. I can use this to full effect during type conversions like trying to convert a string to an integer. If something I need to handle would not create an error, I'll put a condition in the 'else block' that checks to see if the user tried to break any rules of my calculator. Basically anytime I get an input I'll put the user in an infinite loop of having to give the required input, where they only escape to progress in the program if they follow the rules I set forth. Using the 'break' command exits the user from the loop they are currently in and continues the code after the loop. The 'continue' command stops the current iteration of the loop and sends the user to the next iteration without completing the rest of the code within the loop following. I'll use these two to navigate through these loops. Because of these harsh requirements, I'll set very clear rules and give them feedback on why they failed if they do. I'll use a lot of infinite loops in this program.

        To calculate the base conversions, because I'm allowing any base to any base conversions, will be a little more difficult. I'll use a combination of the 'multiply each place in the number by its place's value' method (where you chart out a binary number in a table and multiply each place by 2^n depending on its place, adding the results) and the remainder method (where you divide a number by the desired base, recording remainders, then reverse those remainders to get target number in desired base) to accomplish the conversions.

        From my main menu, I need to send the user either to programmer mode or scientific mode. I'll separate each mode into a method, so I can send them there from a main menu method and go back to an infinitely repeating main menu loop after they choose to be done (so they can go from the menu to the other mode). In programmer mode, they give inputs and the program spits out a number, asking them if they want to do it again after. In my preferred solution, in scientific mode, the user gives an input, the computer gives an answer, and asks for another input. Exiting can happen at any step because there is only one input required for each iteration. This way the user doesn't have to say no to leaving every other input. I'll also store and let the user use the last result like in a normal calculator. After they choose to leave, the user will be sent to the menu.

b. Specific Steps: (this took so long, I hope it was worth it). Link if the image fails somehow: https://www.lucidchart.com/invitations/accept/a9574119-239c-4d5d-b7fb-48c4933bb833



4. Program Implementation:
    a. The python implementation is in the .zip file as well, but just in case, link to the file on Drive: https://drive.google.com/open?id=1PL2Xgy3muZ0qadVPB8DalZOvc4-HTNm-

5. Program Testing:
    a. Test Cases for Each Input and Expected Results:
        - Mode selection:
            - Good input: (string of length one, ‘p’ or ‘s’, upper or lowercase)
                - P ⇒ go to programmer mode
                - p ⇒ go to programmer mode
                - S ⇒ go to scientific mode
                - s ⇒ go to scientific mode
            - Bad input:
                - 1 ⇒ ask for re-input
                - 1.5 ⇒ ask for re-input
                - a ⇒ ask for re-input
                - As ⇒ ask for re-input
                - <empty input> ⇒ ask for re-input

        - Number to base-convert:
            - Good input: (positive integer)
                - 1 ⇒ continue with conversion
                - 150270 ⇒ continue with conversion
                - 1.5 ⇒ pass through int(), continue
            - Bad input:
                - Ahashd ⇒ ask for re-input (specifically for int type)
                - -5 ⇒ ask for re-input (specifically for positive)
            - Edge cases:
                - 0 ⇒ handled as expected
                - max_int ⇒ should be handled, although memory might be exhausted.

        - Base of input: (same cases apply to desired base for resulting number)
            - Good input: (integer [or can convert with int()], value is 2-10, inclusive)
                - 3 ⇒ continue with conversion
                - 5.5 ⇒ pass through int(), continue with conversion
                - 9 ⇒ continue with conversion
            - Bad input:
                - Ajdasb ⇒ ask for re-input
                - 0 ⇒ ask for re-input
                - 15 ⇒ ask for re-input
            - Edge cases:
                - 2 ⇒ handled as base-2, continue
                - 10 ⇒ handled as base-10, continue

- Repeat programmer mode?
    - Good input: (‘y’ or ‘n’, upper or lower case)
        - Y ⇒ send user back to beginning of conversion
        - y ⇒ send user back to beginning of conversion
        - N ⇒ send user back to menu
        - n ⇒ send user back to menu
    - Bad input: (any other input)
        - Yes ⇒ ask for re-input
        - No ⇒ ask for re-input
        - Ahhhh ⇒ ask for re-input
        - 135 ⇒ ask for re-input

- First and second operands
    - Good input: (int or float, positive or negative, or string ‘ans’ or ‘’, string ‘exit’)
        - -5 ⇒ use numerical value, continue
        - 8 ⇒ use numerical value, continue
        - 0 ⇒ use numerical value, continue
        - -5.5 ⇒ use numerical value, do not trim to int, continue
        - 10.5 ⇒ use numerical value, do not trim to int, continue
        - ans ⇒ use value of last result of operation, continue
        - <empty string> ⇒ use value of last result of operation, continue
        - exit ⇒ kick user back to main menu
    - Bad input:
        - Asdjias ⇒ ask for re-input
        - True ⇒ ask for re-input
        - int() ⇒ ask for re-input
    - Edge cases:
        - Min or max values of int or float ⇒ fails if operation increases value beyond scope. Not going to account for this.

- Operator to use
    - Good input: (set of legal operators)
        - + ⇒ add first and second operands
        - - ⇒ subtract second operand from first operand
        - * ⇒ multiply first and second operand
        - / ⇒ divide first operand by second operand
        - ** ⇒ raise first operand to the power of second operand
    - Bad input:
        - 1 ⇒ ask for re-input
        - As ⇒ ask for re-input
        - // ⇒ ask for re-input
        - <empty input> ⇒ ask for re-input

b. Actual Results:

In the .zip file is a video that shows me running through every input listed above in the test cases. This is also a Google Drive link to watch the video: https://drive.google.com/file/d/1RZ73NnOmO1GoiaUQvQyldOWIhq1bxmq1/view?usp=sharing

6. Extra Credit:

The program handles not only the extra credit requirement of binary ⇒ decimal conversion, but all conversions from and to numbers of base-2 to base-10.