Lab 2 - Recursion, Series, Irrational Numbers

1. **Introduction**: My lab four simple. There is more code that I removed than is left, which I will go over in my conclusions. For what I did include, there is a main function that loads sets and then prints unions generated by combining these sets. And there is a function to load sets from a csv file. My custom datatype is used but does not add any functionality beyond a typical string.
2. **Process**: My program starts by populating three Hash Sets from three files, A, B, and C. These files are read into a string and then iterated by line, and each line is split along the comma. With the content before the comma becoming the Hash Set’s key and the remainder becoming a value, see below.

    for line in content.split('\n') {

        if let Some(pos) = line.find(',') {

            // Remove comma from string a

            let (a, b) = line.split\_at(pos + 1);

            let a = (&a[0..pos]).trim();

            let b = b.trim();

            // Adds line to

            output.insert((ReadCountString::new(a.to\_string()),b.to\_string()));

        }

    }

Once the Hash Sets are populated from the files, they are operated with each other and printed to console. The following operations are performed: A and B are unioned, printing all elements in either set. A and B are intersected, printing elements that are in both sets. A and C are differenced, printing all elements in A but not C. And B and C are symmetric differenced, printing elements that are exclusively in one of the sets. See below.

    println!("    A ∪ B: ");

    for (s, v) in a.union(&b) {

        println!("{:5} : {}", s.\_value, v)

    }

    println!("    A ∩ B: ");

    for (s, v) in a.intersection(&b) {

        println!("{:5} : {}", s.\_value, v)

    }

    println!("    A \\ C: ");

    for (s, v) in a.difference(&c) {

        println!("{:5} : {}", s.\_value, v)

    }

    println!("    B △ C: ");

    for (s, v) in a.symmetric\_difference(&c) {

        println!("{:5} : {}", s.\_value, v)

    }

1. **Testing**: The program only takes input from a static file, so testing was easy, just to make sure the file is formatted correctly. However, incorrectly formatted files will cause problems. There must only be one comma per line and there may not be any blank lines, or the program will fail without catching. With valid files the program runs as expected and each operation is run correctly and prints exactly those elements that meet the criteria of each operation.

Below is a valid file. Any string can be used as key or value, so the left column can support any Unicode characters, not just numbers. And the right column can also support full Unicode, so Chinese characters, emojis, and all other characters can be used.

A screenshot of a computer

Description automatically generated

1. **Results**: To save space, I will go over the second and last operations and skip the other two, although all operations are successful.

**A intersect B:**

Highlighted are all the elements that appear in both files. And the rightmost column is the console’s correct output.

|  |  |  |
| --- | --- | --- |
| File A | File B | A ∩ B from console |
| A screenshot of a computer  Description automatically generated | A screenshot of a computer  Description automatically generated |  |

**B symmetric difference C:**

Highlighted are all that appear in both sets. All non-highlighted elements appear in the console output. This is the correct behavior for symmetric difference, which works much like XOR.

|  |  |  |
| --- | --- | --- |
| File B | File C | B △ C: from console |
|  |  |  |

1. **Conclusion**: I think I got all the extra credit done, but I’m still not happy with a my program’s lack of functionality. I wanted my ReadCountString to keep track of how often it was being read. Particularly I wanted to know if the value was read when calling operations like “.union()”. However, I hit a roadblock. Rust doesn’t let me mutate values of an object during certain function calls. The commented-out code below will not compile.

impl PartialEq for ReadCountString {

    fn eq(&self, other: &ReadCountString) -> bool {

        // self.read\_count += 1;

        self.\_value == other.\_value

    }

}

The second roadblock I ran into was not being able to call operations on the result of Hash Set operations. I wanted to take say the union of A and B and then union that result with C. This does not work though because the result of the Hash Set operations is a Union type, not a Hash Set. Operations can only be performed on Hash Sets, and due to Rust’s borrow checker I was not able to create a working function to turn Union types into Hash Sets. The below code is an example of what I want to do but it does not compile.

a.union(&(b.union(&c)))

1. **References and Acknowledgements:** I did all my own programming. I used Rust’s official documentation and the link given in the lab instructions (link below). The guide had some code examples that did not compile. After fixing the issues I shared my solution with classmates. I also used Chatgpt extensively to try and fix my union\_to\_hashset function’s borrowing issues. I was unsuccessful and all code relevant to the Chatgpt wasn’t very helpful in this situation.

<https://www.tutorialspoint.com/hashset-in-rust-programming>

1. **Extra credit process**:

**Implement the program to include full tables of data rather than just the keys:** ✓

My tables consist of a key and a single value.

**Implement the program to accept other data types as keys:** ✓

I don’t know what “other” means here, so I created my own object as a datatype. Functionally my datatype is very similar to a String. However, it isn’t technically a String, and if the counter code worked it would be functionally distinct as well.

**Allow for the inclusion of a 3rd set and implement at least four set operations. Be sure to output the operations performed:** ✓

My program has three sets and four operations. The three sets are a, b, cand c. The three operations are union, intersections, difference, and symmetric difference. The instructions don’t ask for this, but I tried to get my program to be able to run operations on all three sets, say the union of a, b, and c. I could not get this working, as I talked about in my conclusion section.