**CS 3110**

**Assignment #04**

**Design Document**

**Andisha Safdariyan, as3254**

**Design Document for Assignment 4**

Problem Description:

The goal of Problem 1 is to create a command-line utility in OCaml that identifies the set of lines that two files have in common and prints out those lines in sorted order. The utility should support both case-sensitive and case-insensitive comparisons of lines.

Design Overview:

To achieve this goal, we will design and implement a data abstraction for sets in OCaml. We will use functors, an other feature of the OCaml module system, to create a parameterized data abstraction that allows us to specify the type of elements in the set and the comparison operation on those elements. This will enable us to create sets of any type that supports comparison.

Data Abstraction Design:

1. ElementSet Module Type:

- type t : The type of elements in the set.

- val compare : t -> t -> int: Comparison operation on elements.

- val to\_string : t -> string: Function to convert elements to strings.

2. MySet Module Type:

- type element: The type of elements in the set.

- type t: The type of the set.

- val empty : t : Creates an empty set.

- val add : element -> t -> t : Adds an element to the set.

- val to\_string : t -> string : Converts the set to a string.

- val intersection : t -> t -> t : Computes the intersection of two sets.

Functor Design:

We will define a functor named “Make” that takes a module of type “ElementSet” as input and produces a module of type “MySet”. The “Make” functor will implement the set operations using a list representation for simplicity.

Implementation Steps:

1. Define the “ElementSet” module type with the required functions.

2. Define the “MySet” module type with the required functions.

3. Implement the “Make” functor to create a set module using the input “ElementSet” module.

4. Implement functions to read lines from files and construct sets from the lines.

5. Use the set operations to find the intersection of the two sets.

6. Convert the resulting set to a string and print it.

Testing:

We will write an OUnit test suite for the set abstraction to ensure its correctness. We will also test the utility with different input files and comparison options to verify its functionality.

Conclusion:

By designing and implementing a parameterized data abstraction for sets using functors, we can create a flexible and efficient solution to the problem of finding common lines between two files. This approach demonstrates the power and versatility of OCaml's module system in solving real-world problems.