Doruk Özer 70192

Sarp Göl 72368

Project 4

For this project we worked as a whole. We didn't divide the parts, instead we worked collectively for each part and take benefit of our different perspectives to the tasks and we discussed and choose the best possible approaches for every task.

For Part A we decided to create a new data structure called my-array that is constituted of one or more reference values to expressed values. Then we defined the correct extractor and called it expval->arr and defined the array value to the expressed value data structure. Later we defined the new-array-expression that creates the data structure of my-array that holds the specified amount of references to the given value and returns the array value of that data structure. Later we defined update-array that returns num-val 82. It basically finds the reference at given index in the array and sets the expressed value with given value. Later we defined read-array-expression that finds the reference at given index in the array and returns that expressed value. Lastly we defined print-array-expression that prints the every expressed values that the references in the array points to.

For Part B, we didn't create a new data structure, instead we used the array structure that we defined. New-stack-expression returns a array data structure with length 1000 that holds reference points to -1. The reason why we chose -1 as a redflag. Since for stack the values will be integers in the range 1 to 10000, -1 means empty places in the stack. In addition, the reason why the new-stack-expression creates an array with length 1000 is to limit the number of push and pop operations with 1000 as we saw in the constraints and assumptions. We didn't use any global variables. Stack-push-expression basically first empty index which is our red flag and sets the expressed value of that reference to the given value . Stack-pop-expression finds the last pushed expressed value of the stack and sets the expressed value that it points to as -1. Stack-size-expression returns the length of the stack.Empty-stack-expression checks if the length is 0 or not then returns the bool-val #t if it's length is 0 and boolval #f if it's length is not 0. Stack-top-expression works similar to the stack-pop-expression but it finds index of the lastly pushed expressed value and returns it without setting it's reference to -1. Lastly we defined print-stack that prints the expressed values that the references in the array points,if they are not our redflag in the LIFO order.

For Part C we decided to create a procedure with the first expression and the second identifier. Then we applied that procedure, to every expressed value that the references in the array points to and set the those values calculated by the procedure to the corresponding references points.

Our project passes from every test except array-detailed-test-3 and array-comp-proc-test. The reason why our project fails from those 2 tests is that the return value of read-array-expression. In our implementation, read-array-expression returns the value of the specified index, however in those test cases read-array-expression should return the reference value that symbolizes the rest of the array in order to correctly interpret the nested read-array-expressions. What I mean is array value consists of zero or more references to the expressed values, and expressed values also may be an array value so since read-array-expression should return an expressed value, it can also be an array value but we implemented in a way that the read-array-expression returns the exacat value that the reference is pointed to.