Comp 301 Project 4
Burak Yıldırım 72849 – Alkan Akısu 71455

Workload Breakdown:
Alkan was responsible for PartB and PartC.
Burak was responsible for PartA and PartC.
Each team member has successfully completed their responsibilities on time.

Parts that work properly, and that do not work properly:
All requirements of Part A and B are accomplished and they are working without any errors. In Part C, first three tests are running succesfully but only the last test gives incorrect results.

Approach to implementations:
Grammars for the methods are declared in the lang.scm file with the related parameters. Array-val which checks the given parameters are array and a number is added to the data-structures.scm file. Expval->array and expval->arrayLength extractors are added to the data-structures.scm file. Array? is for checking if an array is a reference. createNewArray procedure takes length and value then it fills the array with the related value using newref. arrayGet procedure used for dereference the related value of an index in the array. arraySet procedure used for setting reference with a value to a related index in the array.

Part A:

newarray(length, value): It gets the value-of expressions then it converts them into related data types. Lastly, a new array is created using createNewArray procedure with length and value parameters.

update-array(arr, index, value): It gets the value-of expressions then it converts them into related data types. Lastly, it updates the array with a new value to an index using arraySet procedure.

read-array(arr,index): It gets the value-of expressions then it converts them into related data types. Lastly, it gets the value of an index of the array using arrayGet procedure.

print-array(arr): It gets the value-of expressions then it converts them into related data types. It prints the elements in the array with the printHelper procedure. printHelper procedure takes array, length and index as parameters and iterates through the array and prints the values in each index using arrayGet procedure until its length.

Part B:

newstack(): It creates a new array of 1000 elements initialized with value of 0.

stack-push(stk, val): It gets the value-of expressions then it converts them into related data types. stackPushHelper procedure takes stack, value and index as parameters and is used to push a value to the stack. stackPushHelper iterates through the array by getting the values using arrayGet procedure and when it finds the first 0 value, it puts the value to that index of the array by using arraySet procedure.

stack-pop(stk): It gets the value-of expressions then it converts them into related data types. It uses stackPopHelper procedure which takes stack, index, and remove? Boolean as parameters to pop a value from the stack. It gets the next value using arrayGet procedure and if

the next value equals to 0 and if remove is true, it sets the value of the current index to 0 and returns that value.

stack-size(stk): It gets the value-of expressions then it converts them into related data types. It uses stackSizeHelper procedure to calculate the stack size. stackSizeHelper takes stack and counter as parameters. It increments the counter by 1 until reaching to 0 value in the stack. Lastly, it returns the counter which equals to the size of the stack.

stack-top(stk): It gets the value-of expressions then it converts them into related data types. It uses stackPopHelper procedure which takes stack, index, and remove? Boolean as parameters to get the top of the stack. It gets the next value using arrayGet procedure and if the next value equals to 0 and if remove is false, it returns the current value.

empty-stack?(stk): It gets the value-of expressions then it converts them into related data types. If the first element in the stack equals to 0, it returns true. Otherwise, it returns false.

print-stack(stk): It gets the value-of expressions then it converts them into related data types. It uses printHelper procedure to print the elements in the stack. printHelper procedure takes array, length and index as arguments and it used with stackSizeHelper to find the size of the stack. printHelper iterates through the stack and prints the current values using arrayGet procedure until reaching to its length.

Part C:

Array Comprehension: It gets the value-of expressions then it converts them into related data types. It uses comprehensionHelper procedure that takes body, var, array, size, index, and environment as arguments. In comprehensionHelper procedure, it creates a new environment and it extends this environment with corresponding values. It takes the procedure and applies it to each element in the array by using apply-procedure and arraySet procedures.