COMP 301 - PROJECT 5

Team members:

Muhammet Eren Ozogul
Ada Yıldız
Aynur Ceren Cepni

The code passes all the test cases and bonus part c is also done.

PartA:

Here we have a vector that has 2 fields. The 1st one is a reference to a built in scheme vector and the second one is the length of the vector. When we are reading, updating, or swapping an index in the vector, we go to the reference scheme vector and do reaching or modifying it in there. When copying a vector, we create a new vector and take the scheme vector of the first vector and copy it in the second vector's reference scheme vector.

Part B:

The way that our queue works is that we place our queue to a vector. It has a 4 fields; vector(defined in part a), start, size, and load. Where size is the max size, load is the number of elements in the array. Start is the beginning of the queue. When enqueuing we update the vector element in the index ((load +start)%) and incrementing load by one which gives us a circle-like array. Then we implemented dequeue by pushing our start to one element right since what dequeue does is to pop the first element in our array and decrease the load because now we have 1 less element. We used modula for it to keep going even if it gets to the end of the array so that it can continue adding elements from the beginning when the end is reached. When an element is deleted it becomes a garbage value and won't be an element in the queue. When printing the queue we started from the start point and continued to print the next elements for a number of loads. When we print the queue it starts from the beginning and adds a comma in between the elements.

```
> (run "let x = newqueue(6) in begin enqueue(x, 10); enqueue(x, 11); enqueue(x, 12); dequeue(x); enqueue(x, 40); enqueue(x, 20); dequeue(x); print-queue(x) end")
12, 40, 20
```

Part C (Bonus):

In this part, we simply used a third vector and initialized it with 0 values as the same length as one of the vectors. We also checked if the two vectors to be multiplied have the same length or not and give an error message when needed. We started from the beginning and multiplied each

value pairs by first changing them to number and then casting the result to a num-val. With that num-val we updated the third vector in the corresponding index. We did it for all elements in the vector and returned the third vector.

Workload Distribution:

Part A: We focused on this question together.
Part B: We focused on this question together.
Part C: We focused on this question together.