Metin Arda Oral
Batuhan Arat
Hüseyin Şenol

COMP301 Project 3 Written Questions

Part A
We choose x = 10, y = 12, and z = 16, randomly.
The environment at each step:

- [z = 16]p
- [y=12,z=16]p
- [x=10,y=12,z=16]p

Part B

Expressed Values→Int + Bool + Queue + String

Denoted Values→Int + Bool +Queue + String

Part D

Note: It is theoretically possible to implement all of this below by just using procedures, instead of changing the myProc's nature. However, we chose to think like C implementation just because we have more knowledge and experience with it.

In order to implement myProc language natively, we have to address our functions at the processor level. Our myProc implementation delegates itself to the Scheme. That's the reason we can create custom expected and denoted values such as queue, list etc. Those data structures and/or the fundamentals needed to implement those structures are defined on Scheme, so we can easily extract them from there. However, at the processor level, the CPU doesn't have a data structure for custom data like queue and list. But, it has the knowledge of number, boolean variables and memory addresses at a bitwise level. Therefore, if we want to construct a minimal set of operations, our expressed values should consist of numbers, booleans and memory addresses.

First, we need to be able to do access, read, write operations on stack, and allocate memory on heap. We have no idea how to do this on machine level (how to implement pointers), however we can use a similar implementation like assembly or C. Assume we have implemented functions for memory allocation, like malloc, calloc, realloc, and free.

Second, we can define data types from scratch using numbers, booleans, and memory addresses, just like Assembly. For simplicity, suppose we have defined the data types from our expressed values. Then, we need to implement bitwise operations to our language. Further, we need to change our existing

operations that work on EOPL delegation, if they use any Scheme based code. Also we should implement a parser code to construct its abstract syntax tree.

Third, we need to be able to define and execute functions, luckily myProc already have this implemented. However, we don't have all the functions we need, for example, we don't have while loop yet. We should be able to execute basic logical expressions and loops to proceed. We also need to be able to return specific variables.

Fourth, we need to define arithmetic operators since we use the Scheme's arithmetic operators. Instead, we can define them by logic gates. If we assume we have implemented the bitwise operations, (AND, NOT, OR, XOR, SHIFTL, <, >, == etc.) below is an example of pseudocode of sum operator,

```
(define sum (a b) (
      while ((< b 0) OR (> b 0)) (
            (carry = a AND b
             a = a XOR b
             b = carry SHIFTL 1

)
return a))
```

After all of these steps above, we believe that we got all we needed. Finally, we can implement the queue structure with memory allocation. (we don't have arrays or lists yet) We can implement a queue struct that takes two inputs, queue data type (int) and maximum number of elements in queue (n), and allocates memory for (n+2)*byte-needed-for(int), where 2 integers are needed to store head and tail pointers.

Then, we can add the enqueue function that takes a queue, element and adds the element to the tail of the queue, before incrementing the tail. We can also implement a check to see if a queue is full, and we can also allocate more memory if needed. Similarly, dequeue function would take a queue and decreases the tail by one, if queue is not already empty. We can also implement isEmpty, which would check whether `tail == head`, and size which would return `tail - head`. If we successfully implement all the requirements above, we would have a basic queue structure in our language.

Workload Breakdown

Throughout the project, each of us contributed vitally to the project. Arda started the work and built the fundamentals of the queue structure, Batuhan & Hüseyin continued with more complex of the project. We constantly communicated through our WhatsApp group to ensure what we were doing was correct, and to exchange ideas. We also opened a Google docs document to do this document together. Overall, we are quite satisfied with our work distribution and group performance.