Assignment 1 – Final

I started to implement the final design by using the shell structure provided in the milestone. I used this shell because it had already implemented many of the functions that I needed, and it appeared to be a great starting point for the final design.

Part 1:

I figured that the initialisation of all "token" values for each process would be easiest to implement using a for loop (master) and an if statement (slave). Since we want all "tokens" to be assigned before proccing has begun, we must check that the current process is 0 (if the current rank = 0). If the condition is met, then each "token" is initialised to the value of their process (e.g. process 0 is assigned token 0, process 1 is assigned token 1, etc). These values are then sent to each process using the MPI_SEND function.

Part 2:

I decided it would be best to illustrate the neighbour relationship using a table. This table checks the left and right neighbour for each process

	Process 0	Process 1	Process 2	Process 3
Left	?	0	1	2
Right	1	2	3	?

Looking at the shift table, I notice that there are special cases around the shifting with process 0 and process 3. This will be used to make conditions for my if statements because in all other instances the shifting should be the same. After this implementation, the shift table will appear as:

	Process 0	Process 1	Process 2	Process 3
Left	3	0	1	2
Right	1	2	3	0

Part 3:

I started working through this section by writing out all the shifts we want to occur:

First shift:

Process 0 sends token 0 to process 1
Process 0 received token 3 from process 3
Process 1 sends token 1 to process 2
Process 1 received token 0 from process 0
Process 2 sends token 2 to process 3
Process 2 received token 1 from process 1
Process 3 sends token 3 to process 0
Process 3 received token 2 from process 2

Second shift:

Process 0 sends token 3 to process 1
Process 0 received token 2 from process 3
Process 1 sends token 0 to process 2
Process 1 received token 3 from process 0
Process 2 sends token 1 to process 3
Process 2 received token 0 from process 1
Process 3 sends token 2 to process 0

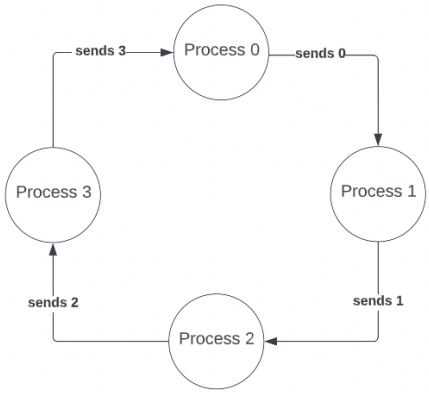
Process 3 received token 1 from process 2

Third shift:

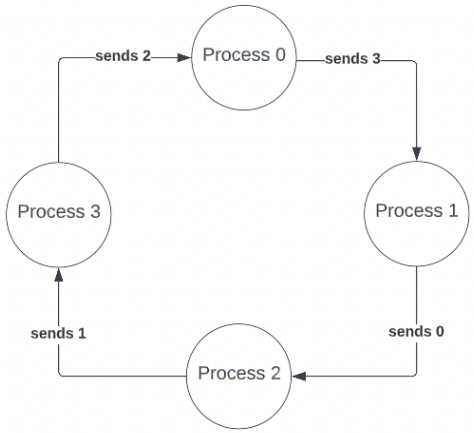
Process 0 sends token 2 to process 1
Process 0 received token 1 from process 3
Process 1 sends token 3 to process 2
Process 1 received token 2 from process 0
Process 2 sends token 0 to process 3
Process 2 received token 3 from process 1
Process 3 sends token 1 to process 0
Process 3 received token 0 from process 2

I decided to simplify this information into an illustration of the ring topology for this task:

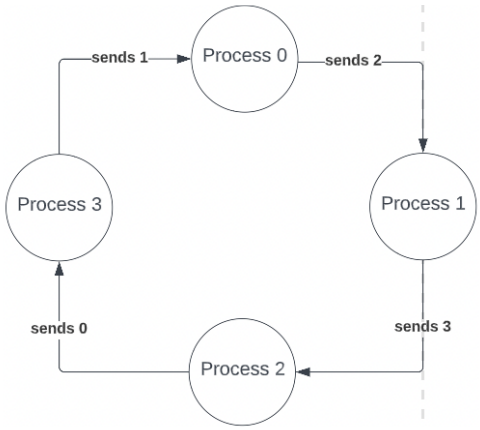
First shift:



Second shift:



Third shift:



Once again, we will be using a for loop (master) and several if statements (slaves) to complete 3 left shifts on the data. The for loop will determine the number of shifts that occur (3) and which shift is currently being undertaken. The if will perform each shift depending on which shift is currently being undertaken.

Output:

Code is not functioning correctly, for some reason it is not ending and some on the receives are incorrect, however, this is the output I am receiving

```
Process 1 sending token 0 to process 2
Process 1 received token 1 from process 0
Process 1 sending token 1 to process 2
Process 1 received token 3 from process 0
Process 1 sending token 3 to process 2
Process 1 received token 0 from process 0

Process 0 sending token 3 to process 1
Process 0 received token 0 from process 3
Process 0 sending token 0 to process 1
Process 0 received token 3 from process 3
Process 0 sending token 3 to process 1

Process 2 sending token 0 to process 3
Process 2 received token 2 from process 1
Process 2 sending token 2 to process 3

Process 3 sending token 0 to process 0
Process 3 received token 3 from process 2
Process 3 sending token 3 to process 0
```

Below are expected outputs

Process 2 received token 1 from process 1
Process 3 received token 2 from process 2
Process 0 received token 2 from process 3
Process 2 sends token 1 to process 3
Process 2 received token 0 from process 1
Process 3 sends token 2 to process 0
Process 3 received token 1 from process 2
Process 0 sends token 2 to process 1
Process 0 received token 1 from process 3
Process 1 received token 2 from process 0
Process 2 received token 3 from process 1
Process 3 sends token 1 to process 0
Process 3 received token 0 from process 2

Part 4:
Each step of communication will be outputted in the user's terminal. This will be implemented in the code using printf statements. These statements will occur before each MPI_Send function and after each MPI_Recv function. The placement of the printf functions is vital in ensuring that the code is correctly outputted.