Balance State is defined as follows:

- Based on the initial load of the processors, we find the average load value.
- We say that the system has achieved a balanced state, if the load on each processor is equal to the absolute value of initial average load - EPSILON.

Algorithm Used to find the balanced state for “k” processors:

1. Initialize the “initial load” for k processors in array processor_load[k];
2. Initialize the “timer vector” for k processors, which is the time at which a processor should do the load balancing activity.
3. Find the total initial load of k processors, which is the summation of loads on k processors.
4. Find the smallest timer value in the timer vector of k processors and find it’s index.
5. Do the load balancing activity for the processor on the above index.
6. Load balancing activity is defined as follows:
    - Find the left neighbors which is processor[index-1] if index is greater than 0. It will be the last processor if the index is 0.
    - Find the right neighbor which is the processor[index +1] if the index is less than k-1. It will be the first processor if the index is k-1.
    - Find the average load of the 3 processors.
    - If the load on the current processor i.e. processor[k-1] is greater than the average then we do the load balancing activity. Otherwise we move out of load balancing activity.
    - If the load of processor[index-1] is less than the average load, then we are allocating the load from processor[index]. Load value given by the processor[index] dependant on whether the load requirement of left neighbor is less or more than the excess load at the current processor.
    - After allocating the load to the left neighbor, we modify the load at left neighbor and current neighbor.
    - Then we allocate the load from processor[index] to right neighbor in the same way as we did for the left neighbor.
    - Eventually, we get the modified load values for processor[index-1], processor[index] and processor[index+1].
7. Processor[index] generates the new time value for itself using the uniform random number generator and updates the timer vector.
8. Increment the cycle count by 1.
9. Compare the load on each processor with the initial average load and see if the balanced state has been reached. If the balanced state is reached, stop the execution. Else, loop to the step4.

Test Results:

EPSILON = 2
No. of processors = 5

```
yagyasen@Yagyawalcyas-MacBook-Pro HW7 % ./yagya_code
The initial load is:
364     150     617     331     383

 cyles: 1823
The Final load is:
367     368     371     371     368
yagyasen@Yagyawalcyas-MacBook-Pro HW7 %
```

EPSILON = 2
No. of processors = 10

```
yagyasen@Yagyawalcyas-MacBook-Pro HW7 % ./yagya_code
The initial load is:
188     352     851     519     905     49      145     109     433     846

 cyles: 465
The Final load is:
440     440     440     440     440     440     439     440     439     439
yagyasen@Yagyawalcyas-MacBook-Pro HW7 %
```

EPSILON = 2
No. of processors = 100

```
yagyasen@Yagyawalcyas-MacBook-Pro HW7 % ./yagya_code
The initial load is:
44      657     246     788     597     523     286     509     606     387     617     15      658
539     476     532     917     578     481     998     477     156     359     434     528     958
423     531     27      970     712     443     173     908     782     81      84      771     396
704     369     433     46      637     420     180     172     467     333     684     740     81
643     112     208     279     47      175     70      18      69      120     470     23      154
239     95      436     984     997     384     633     889     680     472     843     466     914
```

771	760	58	214	109	540	676	886	188	861	977	759	159
793	426	272	178	22	339	269	147	90				

cyles: 266011
The Final load is:

449	449	449	449	449	449	449	449	448	449	449	449	448
449	448	449	449	449	448	449	448	449	449	449	449	448
449	449	449	448	448	449	448	449	448	448	448	449	448
449	449	448	448	449	448	449	449	448	449	448	448	448
448	448	448	448	448	448	448	448	448	448	448	448	448
448	448	448	448	448	448	448	448	448	448	448	448	448
448	448	448	448	448	449	449	449	449	449	449	449	449
449	449	448	448	449	449	449	449	449				

yagyasen@Yagyawalcyas-MacBook-Pro HW7 %