Parallel Computing Coursework 2 Report

January 10, 2022

Contents

1 Code **2**

1.1 Design 2

1.2 Usage 3

2 Correctness Testing **3**

2.1 Sequential Tests 3

2.2 Parallel Tests 4

2.3 Compiling 5

3 Scalability Investigation **5**

3.1 Large Input Arrays 5

3.2 Smaller Input Arrays 8

3.3 Scaling with Precision 9

3.4 Splitting Processes Across Nodes 10

4 Appendix **11**

4.1 Table 1 Logs 11

4.2 Table 2 Logs 16

1 Code

1.1 Design

As it gave good results in the previous shared memory version, I decided to once again to distribute the workload evenly across the processes, using a static allocation at the beginning of the program, rather than using dynamic allocation. Similarly to how dynamic allocation required too many synchronization primitives in the shared memory version, it would require a lot of messaging in a distributed version, which would lead to an even higher overhead. Indeed, as there should not be much deviation in the computation time taken by different processes for this problem, static allocation of workload should continue to work well. For an input array of size $n \times n$ and p processes, the integer division $d = \frac{n^2}{p}$ is calculated, and so is the remainder, $r = n^2 - nd$. The first r processes are each given a workload of $d + 1$ elements, and the rest are given d, such that the remainder is evenly distributed..

Rather than one process calculating the workload distribution and sending it to each other process, I decided to have every process calculate the workload distribution. There are two reasons for this - firstly, supposing I instead took the former approach, all other processes would not be able to do any computation until they receive a message telling them which elements to process, and so they would be idle anyway. Having all processes recalculate the workload distribution removes the need for messaging this entirely, which may save time. Secondly, every process must have access to information about how the array is split in order to use the *MPI_Gatherv()* function, which happens to be very useful for this problem.

MPI_Gatherv() builds one array by combining smaller sub-arrays sent by each process in a communicator. Where *MPI_Gather()* requires the sent sub-arrays to all be the same length, *MPI_Gatherv()* allows varied lengths, as long as information about each sub-array length is given as an argument. This is safe to use as it is a blocking function - so the processor with rank 0 will not proceed until it finishes the full result array, and won't try to combine the sub-arrays until each other processor has copied them out of their send buffers. Using the nonblocking version, *MPI_Igatherv()* may result in a bug when processes start the next iteration before they combine the results from the current one, so the blocking version is necessary.

Each process has a *changed* Boolean variable, which is set to false at the start of each iteration and only changed to true if an element is changed by an amount larger than the precision value. All processes should do another iteration if just one of these Booleans are true. The *MPI_Allreduce()* function with the *MPI_LOR* (logical or) operation reduces all the processes' *changed* variables according to the 'or' operation, and broadcasts the result back to each process. Again, this is a blocking function, so all processes will wait to receive the new reduced value before checking whether to proceed to the next iteration.

Initially, I used a different version of the gather function which broadcasted the full result array back to each processor, however this had a dramatic negative effect on speedup for larger arrays. To make this more efficient, I instead opted to have only the processor with

rank 0 form the full result array. At the start of each iteration, it sends only the relevant rows of the array to each process. That is, the rows which contain elements that are going to be used, and the rows directly above and below them. This means that at a given point in the program's runtime, only process 0 has a sensible, fully correct result array, while the rest may only have a few rows of the array that are up to date. Although this made the code slightly more complicated, it caused a massive speedup for larger problems, which I deemed more valuable.

1.2 Usage

To change the precision, array dimension and whether or not the program prints the before after each iteration, change the 3 variables under the 'arguments' comment in the main function in main.c. To change the contents of the input array, edit the input() function at the top of the file to return the desired element in row i, column j. Figure 2 later in this report shows how to compile using mpicc.

2 Correctness Testing

2.1 Sequential Tests

To verify the correctness of the program I ran the program for smaller arrays at various precisions, and had it print the array after each iteration. Firstly, I performed many tests on the program running with 1 process (sequentially), and planned to compare the results to those with higher amounts of processes. I checked by hand that the following conditions were met in each test:

1. Elements on the boundary of the array are not changed,
2. Elements that are not on the boundary of the array become the average of their four neighbours in the previous iteration (I picked two iterations and sampled random elements in each to check, then calculated the expected results by hand and compared them to the actual results).
3. The penultimate iteration changed one element by a value greater than the precision.
4. The final iteration changed all elements by values smaller than the precision.

All tests used arrays with a fixed form: elements in the top row and leftmost column are 1, and the rest are 0, for example as shown in Figure 1.

The results of the tests are given in Table 1, and their respective excerpts from the console are in the Appendix.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 1: An example 6×6 input matrix of the form used in the tests.

Test ID	Array Size	Precision	Conditions				Result
			1	2	3	4	
1.1	8	0.001	y	y	y	y	Positive
1.2	6	0.01	y	y	y	y	Positive
1.3	7	0.001	y	y	y	y	Positive
1.4	9	0.1	y	y	y	y	Positive
1.5	4	0.0001	y	y	y	y	Positive
1.6	6	0.01	y	y	y	y	Positive
1.7	12	0.00001	y	y	y	y	Positive
1.8	3	0.0001	y	y	y	y	Positive

Table 1: Sequential test results

2.2 Parallel Tests

After testing the sequential version, I moved on to trying the same tests with different amounts of processes. Additionally, I had each process print out which elements of the array they were working on, to ensure that there was no overlap. I also coded in random duration sleeps in each iteration, in order to implement an artificial time drift between processes and ensure any bugs caused by sequential assumptions I may have made while coding had a higher chance of appearing. I did this in my shared memory version, however this time I used longer sleeps, mainly to ensure that the messaging functions would correctly block until they had all the data needed. This time, the test conditions were:

1. The result is equal to that given in the sequential test
2. According to the added prints, no two processes work on the same element.

Tests 2.6 and 2.8 are particularly interesting ones, as they use more processes than there are available elements in the array. Although this is not a sensible use of the program as some processes will be idle, it is important that the program still produces a correct result in these cases.

Test ID	Array Size	Precision	Nodes	Tasks Per Node	Conditions		Result
					1	2	
2.1	8	0.001	2	8	y	y	Positive
2.2	6	0.01	4	3	y	y	Positive
2.3	7	0.001	3	1	y	y	Positive
2.4	9	0.1	1	16	y	y	Positive
2.5	4	0.0001	1	16	y	y	Positive
2.6	6	0.01	4	22	y	y	Positive
2.7	12	0.00001	2	22	y	y	Positive
2.8	3	0.0001	2	5	y	y	Positive

Table 2: Parallel test results

2.3 Compiling

Using the GCC compiler, compiling with -Wall -Wextra -Wconversion (as shown in Figure 2) does not produce any warnings or errors.

```
jjg62@cm30225-login $ mpicc -Wall -Wextra -Wconversion -o main main.c
jjg62@cm30225-login $
```

Figure 2: Compiling as shown does not produce warnings or errors.

3 Scalability Investigation

3.1 Large Input Arrays

I ran the program with a 2000x2000 input array (of the same form shown in Figure 1) and a precision value of 0.0001, with various amounts of nodes and processes. To time them, I use the Linux time command. I compared this with OpenMPI's *MPI_Wtime()* function, and each time there was a difference of less than a second, which arises because the Linux command also takes into account the time taken to run *MPI_Init()* and *MPI_Finalize()*. The time these functions take to run may depend on the amount of nodes and processes, so I believe that keeping them in will give a slightly more accurate speedup value. The results are shown in Table 3.

For smaller amounts of processes, the efficiency values seem significantly lower than the ones I measured in the shared memory version. This is to be expected, as the processes must use the messaging interface each iteration in order to read the previous result array, and create the next one. As OpenMPI is more generalised, it is natural for it to have a higher overhead than shared memory access.

However, OpenMPI allows the use of multiple nodes on the cloud computer, giving access

Nodes	Tasks Per Node	Processes	Time (s)	Speedup	Efficiency	Parallel Overhead (s)
1	1	1	320.025	-	-	-
1	4	4	109.319	2.927	0.732	117.251
1	8	8	65.323	4.899	0.612	202.559
1	12	12	62.888	5.089	0.424	434.631
1	16	16	57.428	5.573	0.348	598.823
1	32	32	51.265	6.243	0.195	1320.455
1	44	44	48.534	6.594	0.150	1815.471
2	44	88	37.482	8.438	0.097	2974.519
3	44	132	44.584	7.178	0.054	5565.063
4	44	176	48.762	6.563	0.037	8262.087

(Parallel overhead is calculated by: $T_o = pT_p - T_s$)

Table 3: 2000x2000 input arrays with 0.0001 precision

to much more processing power and higher speedup overall. The highest speedup measured this time was 8.438, which is significantly better than 5.310, the highest for the shared memory version.

It seems that somewhere between 88 and 132 processes is optimal for this problem size, as there is a speedup decrease at 3x44 processes. This is the point where the overhead required for message passing overtakes the time saved by splitting up the workload.

Plotting the parallel overhead against the number of processes shows that the correlation is almost linear (Figure 3).

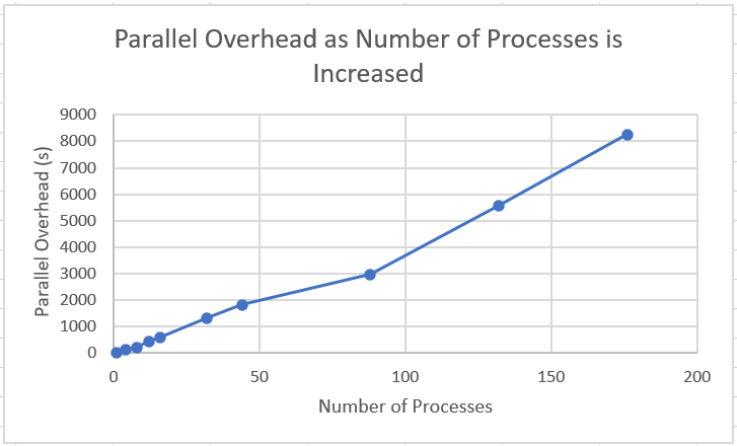

Figure 3: Parallel Overhead for various amounts of processes (2000x2000)

After making some simplifying assumptions (mainly that processing any element takes the same amount of time), the time taken to complete the problem in sequential time can be modeled as:

$$T_{seq} \propto n = d^2$$

$$T_{seq} = k_1 n$$

where d is the dimension of the array, k_1 is a constant and n is the 'size of the problem', i.e. the number of elements in the array. When the program is ran in parallel, it should follow that:

$$T_{par} = \frac{k_1 n}{p} + T_o$$

where p is the number of processes, and T_o is the parallel overhead caused by message passing, and the overhead of setting up the MPI environment. However, the measurements above seem to show that, approximately, $T_o \propto p$, so:

$$T_o = k_2 p$$

for some constant k_2. This gives a way to calculate the isoefficiency:

$$T_{seq} = cT_o$$

$$k_1 n = ck_2 p$$

$$n = O(p)$$

Therefore, the measurements above seem to suggest a linear isoefficiency. This would mean that increasing the array dimension by a factor k (and therefore increasing the problem size by a factor of k^2) and increasing the amount of processes by a factor of k^2 should give approximately the same efficiency. However, Table 4 shows that this is certainly not the case.

Array Dimension	Processes	Time (s)	Speedup	Efficiency
1000	1	77.511	-	-
1000	2	41.184	1.882	0.941
2000	1	320.025	-	-
2000	8	65.323	4.899	0.612
3000	1	746.950	-	-
3000	18	132.229	5.649	0.314

Table 4: Scaling array dimension by k and processes by k^2 to see if there is linear isoefficiency (there isn't!)

What I didn't account for when working out the isoefficiency was that the parallel overhead also appears to scale with the array dimension. This (to an extent) is to be expected, as the larger the array, the more elements have to be sent using message passing. However in this case, at very high sizes, increasing array dimension dramatically lowers the efficiency - seemingly in contrast to Gustafson's law. I also ran the same tests on a program which, using much more complicated code, only sent the bare minimum elements needed to each process, and this was still the case. One possible explanation is that the underlying messaging may scale worse than linearly with the size of the array being sent, compared

to the computation being done which does scale linearly, and so there is a point where increasing the elements sent to each process slows the program down.

To illustrate this effect, Table 5 shows that when the number of processes is constant (at 16), the Karp-Flatt metric increases as the array dimension increases at very high sizes, showing that the amount of communication required relative to computation is increasing.

Array Dimension	Processes	Time (s)	Speedup	**Karp-Flatt**
1000	1	77.511	-	-
1000	16	9.882	7.844	0.069
2000	1	320.025	-	-
2000	16	57.658	5.596	0.124
3000	1	746.950	-	-
3000	16	137.393	5.432	0.130

Table 5: Karp-Flatt metric increases as array size increases

3.2 Smaller Input Arrays

This isn't the case however for smaller array sizes. For example, for 250x250 and 2 processes, the efficiency is 0.7964, which is less than that of the 1000x1000 array (in Table 4), as expected. At 500x500 with 4 processes, the efficiency is very similar at 0.7327 - so it seems that for smaller problem sizes there may be a square isoefficiency: $n = d^2 \propto p^2$. Indeed, Table 6's results, which are from a 500x500 array, show a parallel overhead which seems to have a correlation to the number of processors squared (as shown by Figure 4) which using the same method as above, would give a square isoefficiency. In addition, here it is clear that the efficiency values have decreased from decreasing the problem size, which more in line with Gustafson's law.

Nodes	Tasks Per Node	Processes	Time (s)	Speedup	Efficiency	Parallel Overhead (s)
1	1	1	18.626	-	-	-
1	4	4	6.355	2.931	0.733	6.794
1	8	8	4.329	4.303	0.538	16.006
1	12	12	4.334	4.298	0.358	33.382
1	16	16	4.629	4.024	0.251	55.438
1	32	32	6.844	2.722	0.085	200.382
1	44	44	8.597	2.167	0.049	359.642

(Parallel overhead is calculated by: $T_o = pT_p - T_s$)

Table 6: 500x500 input arrays with 0.0001 precision

To conclude: while input arrays have dimensions smaller than around 1000, they seem to have a square isoefficiency. For larger arrays, this does not apply and in fact, efficiency starts to decrease as array size increases. Going forward, I should look further into the implementations of functions that do a lot of messaging like *MPI_Gatherv()* to see if perhaps they are causing these strange scalability results, and if so, use other functions instead. I could also try to use non-blocking message functions in certain places, to overlap communication and computation.

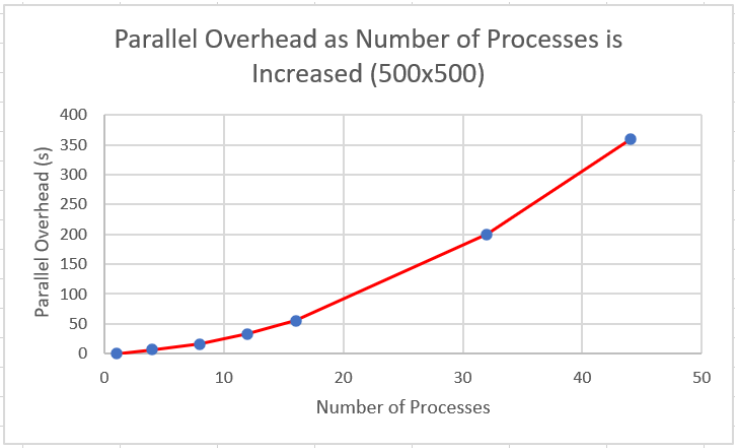

Figure 4: Parallel Overhead for various amounts of processes (500x500)

3.3 Scaling with Precision

Processes	Precision	Time (s)	Efficiency
1	0.001	2.338	-
4	0.001	1.305	0.448
8	0.001	1.174	0.249
16	0.001	1.651	0.089
1	0.0001	18.626	-
4	0.0001	6.355	0.733
8	0.0001	4.303	0.538
16	0.0001	4.629	0.251
1	0.00001	180.330	-
4	0.00001	54.651	0.825
8	0.00001	37.038	0.609
16	0.00001	28.009	0.402
1	0.000001	714.099	-
4	0.000001	213.292	0.837
8	0.000001	143.980	0.620
16	0.000001	108.341	0.412

Table 7: Overview of efficiencies at different precision values

Table 7 shows efficiency values at different precision values for a 500x500 input array. The program seems to scale well with precision. Interestingly, my shared memory implementation had the opposite results; the efficiency decreased as the precision was increased. This distributed memory version therefore seems to be better for task parallelism (lowering precision gives a 'longer' problem), but worse for data parallelism (increasing array size gives each process more elements to compute) than the shared memory version.

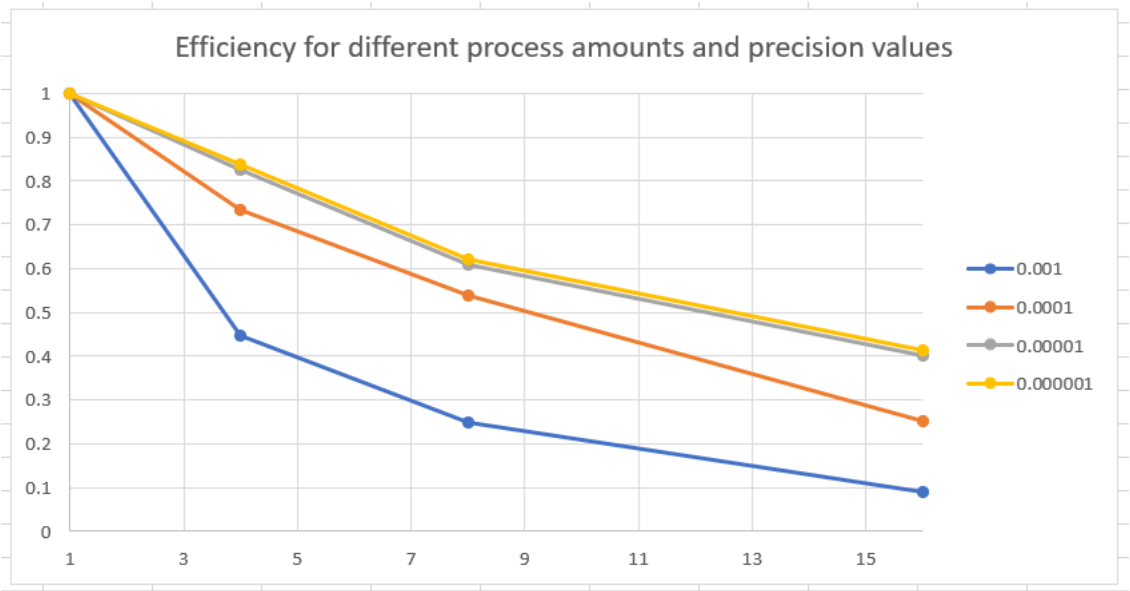

Figure 5: Efficiency for various precisions values and amounts of processes (Table 7)

3.4 Splitting Processes Across Nodes

One final aspect that I wanted to investigate was the effect of processes having to communicate across nodes. I expected that having all processes on one node would be faster as they can use shared memory access, however this turned out not to be the case. Table 8 shows various measurements that all use the same amount of processes, but split across different amounts of nodes, on a 2000x2000 input array with precision 0.0001.

Nodes	Tasks Per Node	Time (s)	Efficiency
1	24	55.780	0.239
2	12	41.981	0.318
3	8	38.267	0.348
4	6	37.242	0.358

Table 8: Effect of splitting processes between nodes

Contrary to my assumption, the efficiency seems to improve as processes are scattered across nodes. This may be because the compiler isn't making optimisations using the fact that the processes are all one one node, and having less processes on one node gives access to more resources like cache memory, allowing more of the sub-array being processed to be stored closer to the cores.

4 Appendix

4.1 Table 1 Logs

Test ID: 1.1

```
Processor 0: 0-63
  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.952237   0.904881   0.855051   0.793305   0.695783   0.497949   0.000000
  1.000000   0.904881   0.813704   0.723849   0.624213   0.493342   0.296827   0.000000
  1.000000   0.855051   0.723849   0.605722   0.489647   0.359182   0.197480   0.000000
  1.000000   0.793305   0.624213   0.489647   0.373572   0.259546   0.135733   0.000000
  1.000000   0.695783   0.493342   0.359182   0.259546   0.172979   0.087729   0.000000
  1.000000   0.497949   0.296827   0.197480   0.135733   0.087729   0.043662   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.952441   0.905248   0.855509   0.793762   0.696149   0.498152   0.000000
  1.000000   0.905248   0.814365   0.724672   0.625035   0.494001   0.297193   0.000000
  1.000000   0.855509   0.724672   0.606748   0.490672   0.360003   0.197935   0.000000
  1.000000   0.793762   0.625035   0.490672   0.374596   0.260366   0.136189   0.000000
  1.000000   0.696149   0.494001   0.360003   0.260366   0.173637   0.088094   0.000000
  1.000000   0.498152   0.297193   0.197935   0.136189   0.088094   0.043864   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.952624   0.905579   0.855921   0.794173   0.696479   0.498335   0.000000
  1.000000   0.905579   0.814960   0.725414   0.625777   0.494595   0.297522   0.000000
  1.000000   0.855921   0.725414   0.607672   0.491596   0.360744   0.198346   0.000000
  1.000000   0.794173   0.625777   0.491596   0.375519   0.261106   0.136599   0.000000
  1.000000   0.696479   0.494595   0.360744   0.261106   0.174230   0.088423   0.000000
  1.000000   0.498335   0.297522   0.198346   0.136599   0.088423   0.044047   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Test ID: 1.2

```
Processor 0: 0-35
  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.888222   0.784585   0.671101   0.479419   0.000000
  1.000000   0.784585   0.604921   0.446118   0.262298   0.000000
  1.000000   0.671101   0.446118   0.287315   0.148814   0.000000
  1.000000   0.479419   0.262298   0.148814   0.070616   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.892292   0.791061   0.677530   0.483350   0.000000
  1.000000   0.791061   0.615351   0.456409   0.268588   0.000000
  1.000000   0.677530   0.456409   0.297466   0.155057   0.000000
  1.000000   0.483350   0.268588   0.155057   0.074407   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.895530   0.796294   0.682705   0.486530   0.000000
  1.000000   0.796294   0.623735   0.464734   0.273704   0.000000
  1.000000   0.682705   0.464734   0.305733   0.160115   0.000000
  1.000000   0.486530   0.273704   0.160115   0.077529   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Test ID: 1.3

```
Processor 0: 0-48
  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

...
```

1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.000000	0.935424	0.871405	0.800141	0.697669	0.498051	0.000000
1.000000	0.871405	0.751727	0.633721	0.494153	0.295649	0.000000
1.000000	0.800141	0.633721	0.492205	0.352915	0.192063	0.000000
1.000000	0.697669	0.494153	0.352915	0.236580	0.121913	0.000000
1.000000	0.498051	0.295649	0.192063	0.121913	0.060679	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.000000	0.935702	0.871823	0.800699	0.698087	0.498330	0.000000
1.000000	0.871823	0.752563	0.634557	0.494989	0.296067	0.000000
1.000000	0.800699	0.634557	0.493318	0.353750	0.192620	0.000000
1.000000	0.698087	0.494989	0.353750	0.237414	0.122330	0.000000
1.000000	0.498330	0.296067	0.192620	0.122330	0.060957	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.000000	0.935912	0.872241	0.801117	0.698504	0.498538	0.000000
1.000000	0.872241	0.753190	0.635392	0.495615	0.296484	0.000000
1.000000	0.801117	0.635392	0.494153	0.354585	0.193037	0.000000
1.000000	0.698504	0.495615	0.354585	0.238040	0.122748	0.000000
1.000000	0.498538	0.296484	0.193037	0.122748	0.061165	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Test ID: 1.4

Processor 0: 0-80

1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.000000	0.500000	0.250000	0.250000	0.250000	0.250000	0.250000	0.250000	0.000000
1.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
1.000000	0.625000	0.437500	0.375000	0.375000	0.375000	0.375000	0.312500	0.000000
1.000000	0.437500	0.125000	0.062500	0.062500	0.062500	0.062500	0.062500	0.000000
1.000000	0.375000	0.062500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.375000	0.062500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.375000	0.062500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.375000	0.062500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.312500	0.062500	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
  1.000000   0.718750   0.531250   0.468750   0.453125   0.453125   0.437500   0.359375   0.000000
  1.000000   0.531250   0.250000   0.140625   0.125000   0.125000   0.125000   0.093750   0.000000
  1.000000   0.468750   0.140625   0.031250   0.015625   0.015625   0.015625   0.015625   0.000000
  1.000000   0.453125   0.125000   0.015625   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.453125   0.125000   0.015625   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.437500   0.125000   0.015625   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.359375   0.093750   0.015625   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.765625   0.609375   0.531250   0.511719   0.503906   0.484375   0.382812   0.000000
  1.000000   0.609375   0.335938   0.218750   0.183594   0.179688   0.167969   0.125000   0.000000
  1.000000   0.531250   0.218750   0.078125   0.042969   0.039062   0.039062   0.027344   0.000000
  1.000000   0.511719   0.183594   0.042969   0.007812   0.003906   0.003906   0.003906   0.000000
  1.000000   0.503906   0.179688   0.039062   0.003906   0.000000   0.000000   0.000000   0.000000
  1.000000   0.484375   0.167969   0.039062   0.003906   0.000000   0.000000   0.000000   0.000000
  1.000000   0.382812   0.125000   0.027344   0.003906   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Test ID: 1.5

```
Processor 0: 0-15
  1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000
  1.000000   0.749756   0.499756   0.000000
  1.000000   0.499756   0.249756   0.000000
  1.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000
  1.000000   0.749878   0.499878   0.000000
  1.000000   0.499878   0.249878   0.000000
  1.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000
  1.000000   0.749939   0.499939   0.000000
  1.000000   0.499939   0.249939   0.000000
  1.000000   0.000000   0.000000   0.000000
```

Test ID: 1.6

```
Processor 0: 0-35
  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

```
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.888222   0.784585   0.671101   0.479419   0.000000
  1.000000   0.784585   0.604921   0.446118   0.262298   0.000000
  1.000000   0.671101   0.446118   0.287315   0.148814   0.000000
  1.000000   0.479419   0.262298   0.148814   0.070616   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.892292   0.791061   0.677530   0.483350   0.000000
  1.000000   0.791061   0.615351   0.456409   0.268588   0.000000
  1.000000   0.677530   0.456409   0.297466   0.155057   0.000000
  1.000000   0.483350   0.268588   0.155057   0.074407   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.895530   0.796294   0.682705   0.486530   0.000000
  1.000000   0.796294   0.623735   0.464734   0.273704   0.000000
  1.000000   0.682705   0.464734   0.305733   0.160115   0.000000
  1.000000   0.486530   0.273704   0.160115   0.077529   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Test ID: 1.7

```
Processor 0: 0-143
  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.250000   0.250000   0.250000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  ...

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.981710   0.963422   0.944942   0.925686   0.904432   0.878860   0.844476   0.791772   0.697917   0.499979   0.000000
  1.000000   0.963422   0.927043   0.890670   0.853382   0.813195   0.766542   0.707284   0.624703   0.499923   0.302003   0.000000
  1.000000   0.944942   0.890670   0.837328   0.783999   0.728444   0.666854   0.593435   0.499850   0.375083   0.208117   0.000000
  1.000000   0.925686   0.853382   0.783999   0.716871   0.649761   0.579026   0.499783   0.406204   0.292458   0.155389   0.000000
  1.000000   0.904432   0.813195   0.728444   0.649761   0.574742   0.499743   0.420502   0.332753   0.233177   0.120994   0.000000
  1.000000   0.878860   0.766542   0.666854   0.579026   0.499743   0.424744   0.349766   0.271163   0.186525   0.095421   0.000000
  1.000000   0.844476   0.707284   0.593435   0.499783   0.420502   0.349766   0.282694   0.215640   0.146360   0.074179   0.000000
  1.000000   0.791772   0.624703   0.499850   0.406204   0.332753   0.271163   0.215640   0.162372   0.109116   0.054946   0.000000
  1.000000   0.697917   0.499923   0.375083   0.292458   0.233177   0.186525   0.146360   0.109116   0.072804   0.036498   0.000000
```

```
1.000000   0.499979   0.302003   0.208117   0.155389   0.120994   0.095421   0.074179   0.054946   0.036498   0.018248   0.000000
1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
1.000000   0.981711   0.963424   0.944945   0.925689   0.904435   0.878863   0.844479   0.791774   0.697919   0.499980   0.000000
1.000000   0.963424   0.927046   0.890674   0.853388   0.813200   0.766548   0.707289   0.624707   0.499926   0.302005   0.000000
1.000000   0.944945   0.890674   0.837334   0.784007   0.728452   0.666862   0.593443   0.499856   0.375087   0.208119   0.000000
1.000000   0.925689   0.853388   0.784007   0.716880   0.649771   0.579035   0.499792   0.406211   0.292463   0.155392   0.000000
1.000000   0.904435   0.813200   0.728452   0.649771   0.574752   0.499753   0.420511   0.332761   0.233182   0.120997   0.000000
1.000000   0.878863   0.766548   0.666862   0.579035   0.499753   0.424755   0.349776   0.271171   0.186530   0.095424   0.000000
1.000000   0.844479   0.707289   0.593443   0.499792   0.420511   0.349776   0.282703   0.215647   0.146365   0.074182   0.000000
1.000000   0.791774   0.624707   0.499856   0.406211   0.332761   0.271171   0.215647   0.162378   0.109120   0.054948   0.000000
1.000000   0.697919   0.499926   0.375087   0.292463   0.233182   0.186530   0.146365   0.109120   0.072807   0.036499   0.000000
1.000000   0.499980   0.302005   0.208119   0.155392   0.120997   0.095424   0.074182   0.054948   0.036499   0.018249   0.000000
1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
1.000000   0.981712   0.963425   0.944947   0.925692   0.904438   0.878866   0.844481   0.791776   0.697920   0.499981   0.000000
1.000000   0.963425   0.927049   0.890678   0.853393   0.813206   0.766554   0.707294   0.624711   0.499929   0.302006   0.000000
1.000000   0.944947   0.890678   0.837340   0.784014   0.728460   0.666870   0.593450   0.499862   0.375091   0.208121   0.000000
1.000000   0.925692   0.853393   0.784014   0.716889   0.649780   0.579044   0.499800   0.406218   0.292468   0.155395   0.000000
1.000000   0.904438   0.813206   0.728460   0.649780   0.574762   0.499763   0.420521   0.332769   0.233188   0.121000   0.000000
1.000000   0.878866   0.766554   0.666870   0.579044   0.499763   0.424765   0.349785   0.271179   0.186536   0.095427   0.000000
1.000000   0.844481   0.707294   0.593450   0.499800   0.420521   0.349785   0.282712   0.215654   0.146370   0.074184   0.000000
1.000000   0.791776   0.624711   0.499862   0.406218   0.332769   0.271179   0.215654   0.162384   0.109124   0.054950   0.000000
1.000000   0.697920   0.499929   0.375091   0.292468   0.233188   0.186536   0.146370   0.109124   0.072810   0.036501   0.000000
1.000000   0.499981   0.302006   0.208121   0.155395   0.121000   0.095427   0.074184   0.054950   0.036501   0.018250   0.000000
1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Test ID: 1.8

```
Processor 0: 0-8
  1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000
  1.000000   0.000000   0.000000

  1.000000   1.000000   1.000000
  1.000000   0.500000   0.000000
  1.000000   0.000000   0.000000

  1.000000   1.000000   1.000000
  1.000000   0.500000   0.000000
  1.000000   0.000000   0.000000
```

4.2 Table 2 Logs

For the purpose of clarity, I have separated the processor element prints and the array prints, however in practice they can be mixed together due to processes taking different amounts of time to write to the output stream.

Test ID: 2.1

```
Processor 14: 56-59
Processor 12: 48-51
Processor 9: 36-39
Processor 15: 60-63
Processor 10: 40-43
Processor 8: 32-35
Processor 11: 44-47
Processor 6: 24-27
Processor 13: 52-55
Processor 0: 0-3
```

```
Processor 2: 8-11
Processor 5: 20-23
Processor 4: 16-19
Processor 3: 12-15
Processor 1: 4-7
Processor 7: 28-31

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.952237   0.904881   0.855051   0.793305   0.695783   0.497949   0.000000
  1.000000   0.904881   0.813704   0.723849   0.624213   0.493342   0.296827   0.000000
  1.000000   0.855051   0.723849   0.605722   0.489647   0.359182   0.197480   0.000000
  1.000000   0.793305   0.624213   0.489647   0.373572   0.259546   0.135733   0.000000
  1.000000   0.695783   0.493342   0.359182   0.259546   0.172979   0.087729   0.000000
  1.000000   0.497949   0.296827   0.197480   0.135733   0.087729   0.043662   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.952441   0.905248   0.855509   0.793762   0.696149   0.498152   0.000000
  1.000000   0.905248   0.814365   0.724672   0.625035   0.494001   0.297193   0.000000
  1.000000   0.855509   0.724672   0.606748   0.490672   0.360003   0.197935   0.000000
  1.000000   0.793762   0.625035   0.490672   0.374596   0.260366   0.136189   0.000000
  1.000000   0.696149   0.494001   0.360003   0.260366   0.173637   0.088094   0.000000
  1.000000   0.498152   0.297193   0.197935   0.136189   0.088094   0.043864   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.952624   0.905579   0.855921   0.794173   0.696479   0.498335   0.000000
  1.000000   0.905579   0.814960   0.725414   0.625777   0.494595   0.297522   0.000000
  1.000000   0.855921   0.725414   0.607672   0.491596   0.360744   0.198346   0.000000
  1.000000   0.794173   0.625777   0.491596   0.375519   0.261106   0.136599   0.000000
  1.000000   0.696479   0.494595   0.360744   0.261106   0.174230   0.088423   0.000000
  1.000000   0.498335   0.297522   0.198346   0.136599   0.088423   0.044047   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Processes are allocated correct elements. Parallel and sequential versions match!

Test ID: 2.2

```
Processor 2: 6-8
Processor 0: 0-2
Processor 3: 9-11
Processor 6: 18-20
Processor 10: 30-32
Processor 1: 3-5
Processor 4: 12-14
Processor 9: 27-29
Processor 5: 15-17
Processor 7: 21-23
Processor 8: 24-26

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.888222   0.784585   0.671101   0.479419   0.000000
  1.000000   0.784585   0.604921   0.446118   0.262298   0.000000
  1.000000   0.671101   0.446118   0.287315   0.148814   0.000000
  1.000000   0.479419   0.262298   0.148814   0.070616   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.892292   0.791061   0.677530   0.483350   0.000000
  1.000000   0.791061   0.615351   0.456409   0.268588   0.000000
  1.000000   0.677530   0.456409   0.297466   0.155057   0.000000
  1.000000   0.483350   0.268588   0.155057   0.074407   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.895530   0.796294   0.682705   0.486530   0.000000
  1.000000   0.796294   0.623735   0.464734   0.273704   0.000000
  1.000000   0.682705   0.464734   0.305733   0.160115   0.000000
  1.000000   0.486530   0.273704   0.160115   0.077529   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Processes are allocated correct elements. Parallel and sequential versions match!

Test ID: 2.3

```
Processor 0: 0-16
Processor 2: 33-48
Processor 1: 17-32

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
```

```
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.935424   0.871405   0.800141   0.697669   0.498051   0.000000
  1.000000   0.871405   0.751727   0.633721   0.494153   0.295649   0.000000
  1.000000   0.800141   0.633721   0.492205   0.352915   0.192063   0.000000
  1.000000   0.697669   0.494153   0.352915   0.236580   0.121913   0.000000
  1.000000   0.498051   0.295649   0.192063   0.121913   0.060679   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.935702   0.871823   0.800699   0.698087   0.498330   0.000000
  1.000000   0.871823   0.752563   0.634557   0.494989   0.296067   0.000000
  1.000000   0.800699   0.634557   0.493318   0.353750   0.192620   0.000000
  1.000000   0.698087   0.494989   0.353750   0.237414   0.122330   0.000000
  1.000000   0.498330   0.296067   0.192620   0.122330   0.060957   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.935912   0.872241   0.801117   0.698504   0.498538   0.000000
  1.000000   0.872241   0.753190   0.635392   0.495615   0.296484   0.000000
  1.000000   0.801117   0.635392   0.494153   0.354585   0.193037   0.000000
  1.000000   0.698504   0.495615   0.354585   0.238040   0.122748   0.000000
  1.000000   0.498538   0.296484   0.193037   0.122748   0.061165   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Processes are allocated correct elements. Parallel and sequential versions match!

Test ID: 2.4

```
Processor 14: 71-75
Processor 1: 6-10
Processor 2: 11-15
Processor 9: 46-50
Processor 4: 21-25
Processor 12: 61-65
Processor 6: 31-35
Processor 7: 36-40
Processor 15: 76-80
Processor 8: 41-45
Processor 13: 66-70
Processor 11: 56-60
```

```
Processor 0: 0-5
Processor 5: 26-30
Processor 10: 51-55
Processor 3: 16-20

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.625000   0.437500   0.375000   0.375000   0.375000   0.375000   0.312500   0.000000
  1.000000   0.437500   0.125000   0.062500   0.062500   0.062500   0.062500   0.062500   0.000000
  1.000000   0.375000   0.062500   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.375000   0.062500   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.375000   0.062500   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.375000   0.062500   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.312500   0.062500   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.718750   0.531250   0.468750   0.453125   0.453125   0.437500   0.359375   0.000000
  1.000000   0.531250   0.250000   0.140625   0.125000   0.125000   0.125000   0.093750   0.000000
  1.000000   0.468750   0.140625   0.031250   0.015625   0.015625   0.015625   0.015625   0.000000
  1.000000   0.453125   0.125000   0.015625   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.453125   0.125000   0.015625   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.437500   0.125000   0.015625   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.359375   0.093750   0.015625   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.765625   0.609375   0.531250   0.511719   0.503906   0.484375   0.382812   0.000000
  1.000000   0.609375   0.335938   0.218750   0.183594   0.179688   0.167969   0.125000   0.000000
  1.000000   0.531250   0.218750   0.078125   0.042969   0.039062   0.039062   0.027344   0.000000
  1.000000   0.511719   0.183594   0.042969   0.007812   0.003906   0.003906   0.003906   0.000000
  1.000000   0.503906   0.179688   0.039062   0.003906   0.000000   0.000000   0.000000   0.000000
  1.000000   0.484375   0.167969   0.039062   0.003906   0.000000   0.000000   0.000000   0.000000
  1.000000   0.382812   0.125000   0.027344   0.003906   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Processes are allocated correct elements. Parallel and sequential versions match!

Test ID: 2.5

```
Processor 12: 12-12
Processor 13: 13-13
Processor 6: 6-6
Processor 5: 5-5
Processor 1: 1-1
Processor 2: 2-2
Processor 10: 10-10
Processor 15: 15-15
Processor 8: 8-8
Processor 9: 9-9
Processor 11: 11-11
Processor 4: 4-4
Processor 14: 14-14
Processor 7: 7-7
Processor 3: 3-3
Processor 0: 0-0

  1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000
  1.000000   0.749756   0.499756   0.000000
  1.000000   0.499756   0.249756   0.000000
  1.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000
  1.000000   0.749878   0.499878   0.000000
  1.000000   0.499878   0.249878   0.000000
  1.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000
  1.000000   0.749939   0.499939   0.000000
  1.000000   0.499939   0.249939   0.000000
  1.000000   0.000000   0.000000   0.000000
```

Processes are allocated correct elements. Parallel and sequential versions match!

Test ID: 2.6

```
Processor 8: 8-8
Processor 19: 19-19
Processor 13: 13-13
Processor 65: 36-35
Processor 80: 36-35
Processor 16: 16-16
Processor 48: 36-35
Processor 68: 36-35
Processor 12: 12-12
```

```
Processor 62: 36-35
Processor 71: 36-35
Processor 20: 20-20
Processor 51: 36-35
Processor 84: 36-35
Processor 1: 1-1
Processor 64: 36-35
Processor 86: 36-35
Processor 14: 14-14
Processor 56: 36-35
Processor 5: 5-5
Processor 54: 36-35
Processor 10: 10-10
Processor 55: 36-35
Processor 6: 6-6
Processor 59: 36-35
Processor 0: 0-0
Processor 60: 36-35
Processor 4: 4-4
Processor 44: 36-35
Processor 3: 3-3
Processor 9: 9-9
Processor 11: 11-11
Processor 7: 7-7
Processor 2: 2-2
Processor 17: 17-17
Processor 21: 21-21
Processor 18: 18-18
Processor 15: 15-15
Processor 33: 33-33
Processor 24: 24-24
Processor 37: 36-35
Processor 61: 36-35
Processor 41: 36-35
Processor 45: 36-35
Processor 27: 27-27
Processor 53: 36-35
Processor 34: 34-34
Processor 49: 36-35
Processor 25: 25-25
Processor 47: 36-35
Processor 39: 36-35
Processor 52: 36-35
Processor 42: 36-35
Processor 57: 36-35
Processor 30: 30-30
Processor 63: 36-35
Processor 40: 36-35
Processor 46: 36-35
Processor 22: 22-22
Processor 58: 36-35
Processor 23: 23-23
Processor 50: 36-35
Processor 29: 29-29
Processor 32: 32-32
Processor 36: 36-35
Processor 28: 28-28
Processor 38: 36-35
Processor 35: 35-35
Processor 43: 36-35
```

```
Processor 26: 26-26
Processor 31: 31-31
Processor 76: 36-35
Processor 87: 36-35
Processor 81: 36-35
Processor 72: 36-35
Processor 83: 36-35
Processor 73: 36-35
Processor 78: 36-35
Processor 69: 36-35
Processor 85: 36-35
Processor 66: 36-35
Processor 82: 36-35
Processor 70: 36-35
Processor 74: 36-35
Processor 79: 36-35
Processor 67: 36-35
Processor 77: 36-35
Processor 75: 36-35

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.888222   0.784585   0.671101   0.479419   0.000000
  1.000000   0.784585   0.604921   0.446118   0.262298   0.000000
  1.000000   0.671101   0.446118   0.287315   0.148814   0.000000
  1.000000   0.479419   0.262298   0.148814   0.070616   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.892292   0.791061   0.677530   0.483350   0.000000
  1.000000   0.791061   0.615351   0.456409   0.268588   0.000000
  1.000000   0.677530   0.456409   0.297466   0.155057   0.000000
  1.000000   0.483350   0.268588   0.155057   0.074407   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.895530   0.796294   0.682705   0.486530   0.000000
  1.000000   0.796294   0.623735   0.464734   0.273704   0.000000
  1.000000   0.682705   0.464734   0.305733   0.160115   0.000000
  1.000000   0.486530   0.273704   0.160115   0.077529   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Note: When a processor prints n-$(n-1)$, it does so because it prints its start point, followed by start point + chunk size - 1 (so it has chunk size 0 as expected)

Processes are allocated correct elements. Parallel and sequential versions match!

Test ID: 2.7

```
Processor 5: 20-23
Processor 4: 16-19
Processor 28: 96-98
Processor 2: 8-11
Processor 42: 138-140
Processor 18: 66-68
Processor 34: 114-116
Processor 20: 72-74
Processor 29: 99-101
Processor 16: 60-62
Processor 24: 84-86
Processor 15: 57-59
Processor 43: 141-143
Processor 11: 44-47
Processor 30: 102-104
Processor 32: 108-110
Processor 38: 126-128
Processor 33: 111-113
Processor 27: 93-95
Processor 35: 117-119
Processor 36: 120-122
Processor 22: 78-80
Processor 41: 135-137
Processor 26: 90-92
Processor 40: 132-134
Processor 31: 105-107
Processor 23: 81-83
Processor 25: 87-89
Processor 37: 123-125
Processor 19: 69-71
Processor 39: 129-131
Processor 6: 24-27
Processor 17: 63-65
Processor 3: 12-15
Processor 14: 54-56
Processor 10: 40-43
Processor 7: 28-31
Processor 21: 75-77
Processor 8: 32-35
Processor 13: 51-53
Processor 12: 48-50
Processor 1: 4-7
Processor 9: 36-39
Processor 0: 0-3
  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.500000   0.250000   0.250000   0.250000   0.250000   0.250000   0.250000   0.250000   0.250000   0.250000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.250000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

...

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.981710   0.963422   0.944942   0.925686   0.904432   0.878860   0.844476   0.791772   0.697917   0.499979   0.000000
  1.000000   0.963422   0.927043   0.890670   0.853382   0.813195   0.766542   0.707284   0.624703   0.499923   0.302003   0.000000
  1.000000   0.944942   0.890670   0.837328   0.783999   0.728444   0.666854   0.593435   0.499850   0.375083   0.208117   0.000000
  1.000000   0.925686   0.853382   0.783999   0.716871   0.649761   0.579026   0.499783   0.406204   0.292458   0.155389   0.000000
  1.000000   0.904432   0.813195   0.728444   0.649761   0.574742   0.499743   0.420502   0.332753   0.233177   0.120994   0.000000
  1.000000   0.878860   0.766542   0.666854   0.579026   0.499743   0.424744   0.349766   0.271163   0.186525   0.095421   0.000000
  1.000000   0.844476   0.707284   0.593435   0.499783   0.420502   0.349766   0.282694   0.215640   0.146360   0.074179   0.000000
  1.000000   0.791772   0.624703   0.499850   0.406204   0.332753   0.271163   0.215640   0.162372   0.109116   0.054946   0.000000
  1.000000   0.697917   0.499923   0.375083   0.292458   0.233177   0.186525   0.146360   0.109116   0.072804   0.036498   0.000000
  1.000000   0.499979   0.302003   0.208117   0.155389   0.120994   0.095421   0.074179   0.054946   0.036498   0.018248   0.000000
  1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

  1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
  1.000000   0.981711   0.963424   0.944945   0.925689   0.904435   0.878863   0.844479   0.791774   0.697919   0.499980   0.000000
  1.000000   0.963424   0.927046   0.890674   0.853388   0.813200   0.766548   0.707289   0.624707   0.499926   0.302005   0.000000
```

```
1.000000   0.944945   0.890674   0.837334   0.784007   0.728452   0.666862   0.593443   0.499856   0.375087   0.208119   0.000000
1.000000   0.925689   0.853388   0.784007   0.716880   0.649771   0.579035   0.499792   0.406211   0.292463   0.155392   0.000000
1.000000   0.904435   0.813200   0.728452   0.649771   0.574752   0.499753   0.420511   0.332761   0.233182   0.120997   0.000000
1.000000   0.878863   0.766548   0.666862   0.579035   0.499753   0.424755   0.349776   0.271171   0.186530   0.095424   0.000000
1.000000   0.844479   0.707289   0.593443   0.499792   0.420511   0.349776   0.282703   0.215647   0.146365   0.074182   0.000000
1.000000   0.791774   0.624707   0.499856   0.406211   0.332761   0.271171   0.215647   0.162378   0.109120   0.054948   0.000000
1.000000   0.697919   0.499926   0.375087   0.292463   0.233182   0.186530   0.146365   0.109120   0.072807   0.036499   0.000000
1.000000   0.499980   0.302005   0.208119   0.155392   0.120997   0.095424   0.074182   0.054948   0.036499   0.018249   0.000000
1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000

1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000   1.000000
1.000000   0.981712   0.963425   0.944947   0.925692   0.904438   0.878866   0.844481   0.791776   0.697920   0.499981   0.000000
1.000000   0.963425   0.927049   0.890678   0.853393   0.813206   0.766554   0.707294   0.624711   0.499929   0.302006   0.000000
1.000000   0.944947   0.890678   0.837340   0.784014   0.728460   0.666870   0.593450   0.499862   0.375091   0.208121   0.000000
1.000000   0.925692   0.853393   0.784014   0.716889   0.649780   0.579044   0.499800   0.406218   0.292468   0.155395   0.000000
1.000000   0.904438   0.813206   0.728460   0.649780   0.574762   0.499763   0.420521   0.332769   0.233188   0.121000   0.000000
1.000000   0.878866   0.766554   0.666870   0.579044   0.499763   0.424765   0.349785   0.271179   0.186536   0.095427   0.000000
1.000000   0.844481   0.707294   0.593450   0.499800   0.420521   0.349785   0.282712   0.215654   0.146370   0.074184   0.000000
1.000000   0.791776   0.624711   0.499862   0.406218   0.332769   0.271179   0.215654   0.162384   0.109124   0.054950   0.000000
1.000000   0.697920   0.499929   0.375091   0.292468   0.233188   0.186536   0.146370   0.109124   0.072810   0.036501   0.000000
1.000000   0.499981   0.302006   0.208121   0.155395   0.121000   0.095427   0.074184   0.054950   0.036501   0.018250   0.000000
1.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000   0.000000
```

Processes are allocated correct elements. Parallel and sequential versions match!

Test ID: 2.8

```
Processor 1: 1-1
Processor 3: 3-3
Processor 4: 4-4
Processor 6: 6-6
Processor 7: 7-7
Processor 2: 2-2
Processor 9: 9-8
Processor 0: 0-0
Processor 8: 8-8
Processor 5: 5-5

  1.000000   1.000000   1.000000
  1.000000   0.000000   0.000000
  1.000000   0.000000   0.000000

  1.000000   1.000000   1.000000
  1.000000   0.500000   0.000000
  1.000000   0.000000   0.000000

  1.000000   1.000000   1.000000
  1.000000   0.500000   0.000000
  1.000000   0.000000   0.000000
```

Processes are allocated correct elements. Parallel and sequential versions match!