CS342
Operating Systems

Project 1

Name: Sabit Gökberk Karaca

Student ID: 21401862

Section: 2

Date: 28.02.2018

Part C

1. prime - (Pipe Implementation)

Table 1: Experiment with 1000 integers

	10 Children	20 Children	30 Children	40 Children	50 Children
Trial 1	0.151	0.141	0.144	0.125	0.135
Trial 2	0.144	0.134	0.125	0.136	0.128
Trial 3	0.151	0.133	0.132	0.135	0.132
Average	0.148667	0.136	0.133667	0.132	0.131667

Table 2: Experiment with 10000 integers

	10 Children	20 Children	30 Children	40 Children	50 Children
Trial 1	6.857	5.497	4.981	4.786	4.349
Trial 2	6.921	5.607	5.001	4.508	4.541
Trial 3	6.801	5.643	5.012	4.551	4.421
Average	6.859667	5.582333	4.998	4.615	4.437

Table 3: Experiment with 25000 integers

	10 Children	20 Children	30 Children	40 Children	50 Children
Trial 1	26.423	20.851	18.796	17.599	17.186
Trial 2	25.951	21.195	18.261	17.592	17.224
Trial 3	26.634	20.823	18.555	17.672	16.871
Average	26.336	20.95633	18.53733	17.621	17.09367

Table 4: Experiment with 50000 integers

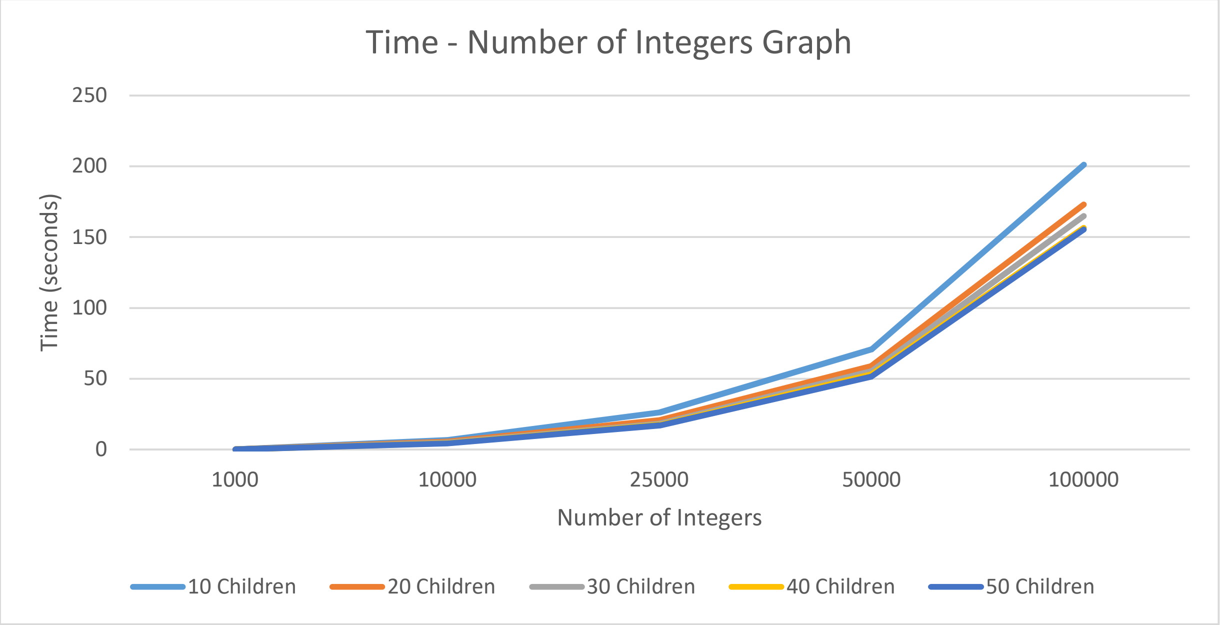
	10 Children	20 Children	30 Children	40 Children	50 Children
Trial 1	70.944	58.462	55.587	54.076	50.972
Trial 2	70.807	59.709	55.959	52.477	52.245
Trial 3	70.818	59.283	54.567	53.082	52.058
Average	70.85633	59.15133	55.371	53.21167	51.75833

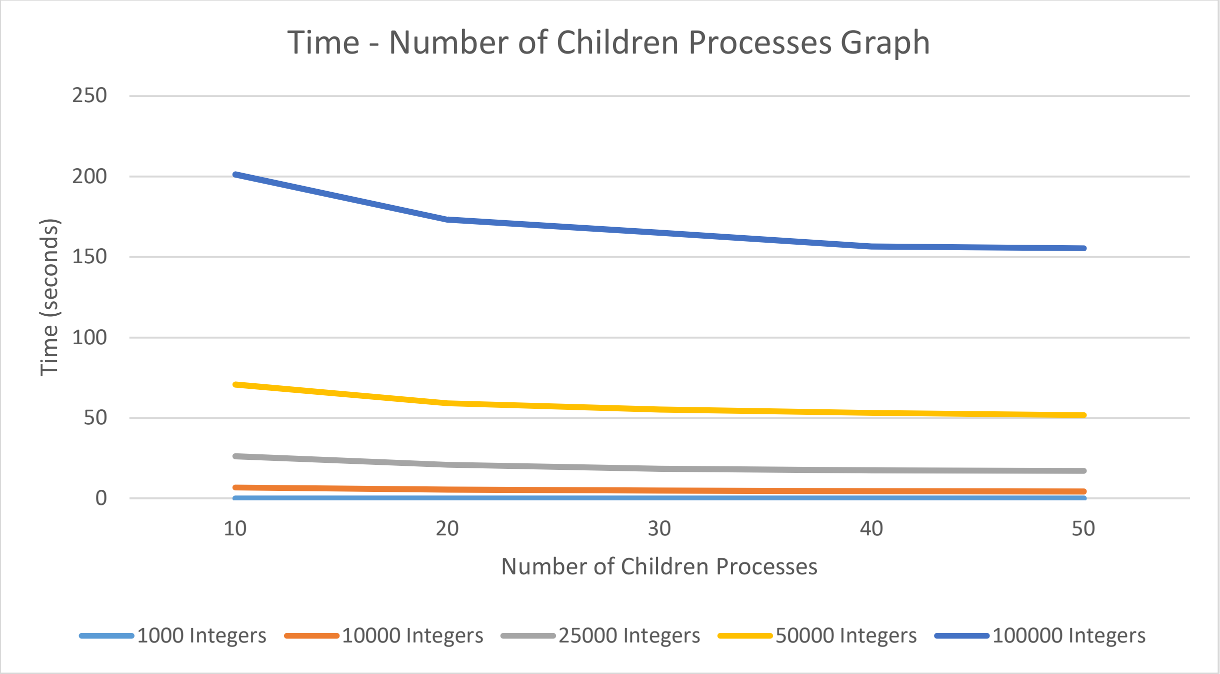
Table 5: Experiment with 100000 integers

	10 Children	20 Children	30 Children	40 Children	50 Children
Trial 1	200.087	168.997	167.503	155.177	155.951
Trial 2	198.711	175.956	165.145	158.535	154.276
Trial 3	205.017	174.449	162.257	156.2	156.078
Average	201.2717	173.134	164.9683	156.6373	155.435

Table 6: Averages

	10 Children	20 Children	30 Children	40 Children	50 Children
1000 Integers	0.148667	0.136	0.133667	0.132	0.131667
10000 Integers	6.859667	5.582333	4.998	4.615	4.437
25000 Integers	26.336	20.95633	18.53733	17.621	17.09367
50000 Integers	70.85633	59.15133	55.371	53.21167	51.75833
100000 Integers	201.2717	173.134	164.9683	156.6373	155.435





2. mqprime – (Message Queue Implementation)

Table 1: Experiment with 1000 integers

	1 Children	2 Children	3 Children	4 Children	5 Children
Trial 1	0.324	0.246	0.226	0.264	0.269
Trial 2	0.316	0.279	0.255	0.269	0.268
Trial 3	0.318	0.269	0.297	0.208	0.292
Average	0.319333	0.264667	0.259333	0.247	0.276333

Table 2: Experiment with 10000 integers

	1 Children	2 Children	3 Children	4 Children	5 Children
Trial 1	14.728	13.385	10.659	10.413	8.996
Trial 2	15.019	13.392	10.496	9.48	9.802
Trial 3	14.557	12.546	10.939	9.905	8.865
Average	14.768	13.10767	10.698	9.932667	9.221

Table 3: Experiment with 25000 integers

	1 Children	2 Children	3 Children	4 Children	5 Children
Trial 1	71.918	57.597	49.396	45.282	42.524
Trial 2	71.715	66.019	49.616	50.123	43.045
Trial 3	70.208	59.516		43.151	38.597
Average	71.28033	61.044	49.506	46.18533	41.38867

Table 4: Experiment with 50000 integers

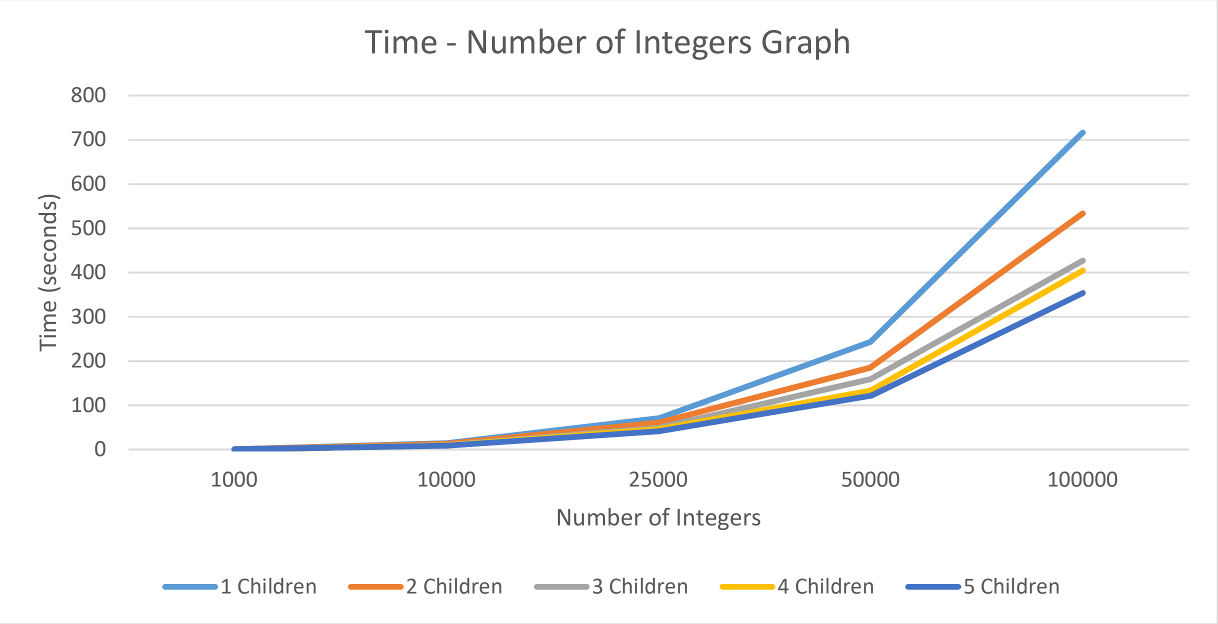
	1 Children	2 Children	3 Children	4 Children	5 Children
Trial 1	247.893	186.972	160.231	133.283	123.951
Trial 2	238.911	185.212	157.328	135.428	121.492
Trial 3	242.718	185.879	160.995	131.671	118.809
Average	243.174	186.021	159.518	133.4607	121.4173

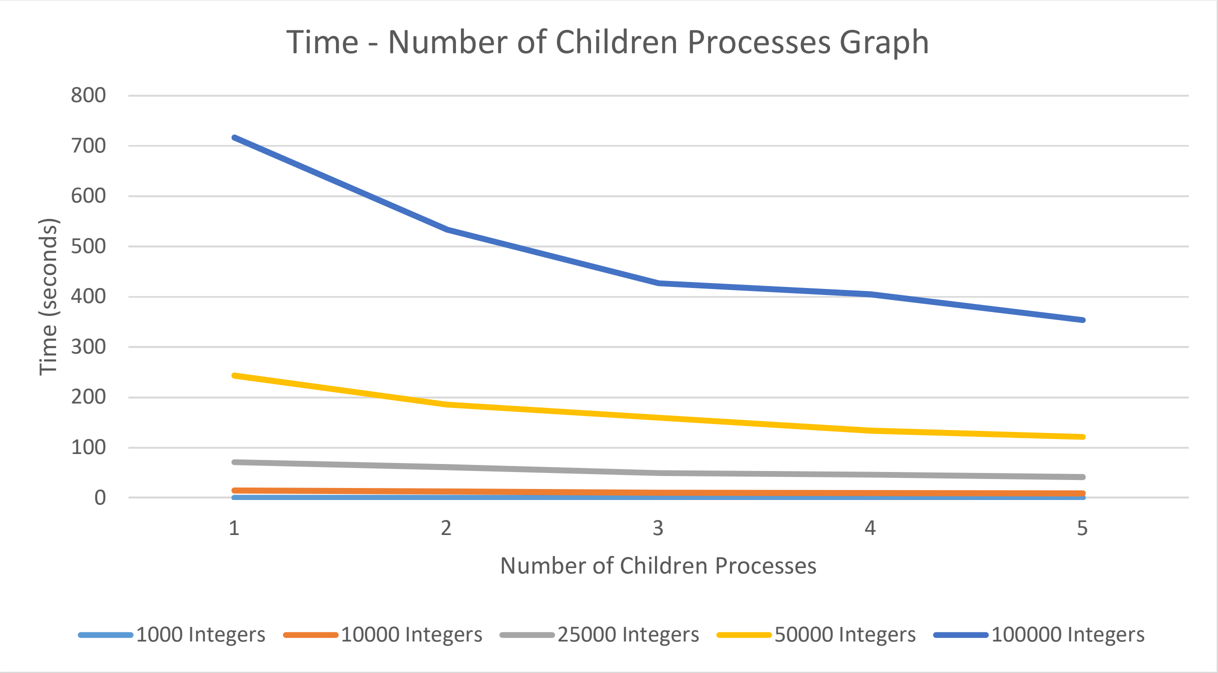
Table 5: Experiment with 100000 integers

	1 Children	2 Children	3 Children	40 Children	50 Children
Trial 1	713.868	532.584	427.319	404.067	357.355
Trial 2	719.862	540.512	417.989	408.742	350.784
Trial 3	715.679	528.356	435.769	401.867	353.275
Average	716.4697	533.8173	427.0257	404.892	353.8047

Table 6: Averages

	1 Children	2 Children	3 Children	4 Children	5 Children
1000 Integers	0.319333	0.264667	0.259333	0.247	0.276333
10000 Integers	14.768	13.10767	10.698	9.932667	9.221
25000 Integers	71.28033	61.044	49.506	46.18533	41.38867
50000 Integers	243.174	186.021	159.518	133.4607	121.4173
100000 Integers	716.4697	533.8173	427.0257	404.892	353.8047





Conclusion

In this project, I have studied two different ways of communication between different processes. In the first part of project, I implemented pipes between children processes and main process to read and write data. In the second part of project, I implemented message queues to send and receive data.

After completing the project, I have conducted an experiment to see the effect of parameters on the running time of designed program. Experiment is separated into two parts, firstly I have tested the program that uses pipes for communication and then I have tested the program that uses message queues. Since the effect of changing parameters was the same on these two programs, they will be discussed together in this part of the report.

In this experiment, 5 sample values for number of integers parameter and 5 sample values for number of children processes parameter are chosen and they are changed one by one. Each pair of parameters give a different result in terms of the running time of program. In order to achieve a precise experiment, each pair of parameters are used 3 times and average value of these 3 results are used as the result of that pair. Detailed information about the results of experiments can be found in the related tables (See Tables 1-5). After conducting the experiment for all pairs of parameters, the average values are collected in one table (See Table 6).

This table gives detailed information about individual parameter changes. For example, going right on the same row means increasing the number of children while keeping the number of integers same. On the other hand, going down in a column means increasing the number of integers while keeping the number of children same. Since data visualization is a better way of interpreting the results, these tables are transformed into 2 different graphs, Time – Number of Children Processes and Time – Number of Integers graphs.

When Time – Number of Integers graphs are investigated, it is possible to see that the running time increases exponentially when number of integers are increased. This can be explained by the relation between number of prime numbers detected in each iteration and number of total integers. After each iteration, prime numbers and their multiples are removed from the integers’ list. If the total number of integers is too big, it is not possible to decrease them quickly and the rate of decrease is very small at the beginning. Therefore, an exponential line is seen for each series in the graph. Another interpretation for this graph can be about the number of child processes and their effect on the exponential growth. As seen in the graph, if the number of children processes are small, the growth rate is bigger because in each iteration a small number of primes are detected and removed which results in small decrease rate of total number of integers.

Similarly, Time – Number of Children Processes graphs also gives information about the relationship between running time of program and changes of parameters. In this graph, each line indicates an experiment where the number of integers are fixed and number of children processes are manipulated. This time, an exponential decrease is seen in the graph. This is also a reasonable result since when the number of children processes are increased, the number of detected prime numbers in each iteration also increases. As a result of this, total number of integers decreases with a high rate and program terminates in a shorter time. Although some lines seem like linear in the graph, this is because of unbalanced values of different series. When these lines are observed under appropriate zoom levels, exponential decrease can be seen for them too.

Finally, as mentioned above, all of these results were expected theoretically. After the experiment, experimental data also supports the theoretical knowledge. Therefore, it is possible to say that the experiment was successful.