Q 1. (20 points) A sequential implementation of the Sieve of Earthosthenes marks about 2.2 million cells in order to compute all primes less than 1 million. Estimate the maximum speedup achievable by the control-parallel (shared memory) version of the Sieve of Erathosthenes as it finds all the primes less than 1 million.

Sequential implementation = 2.2 million cells

$$\text{Speed up} = \frac{\text{Time taken by single processor}}{\text{Time taken by multiple processors}}$$

For control-parallel approach, we use multiple processors to achieve speed up.

Calculate the time taken by many processors to achieve the same in a much lesser time.

Formula:

$$\text{Number of multiple of given prime} = \left\lceil \frac{(n+1)-p^2}{P} \right\rceil$$

$$\text{Number of multiple of 2} = \left\lceil \frac{(1000000+1)-2^2}{2} \right\rceil = 499{,}999$$

$$\text{Number of multiple of 3} = \left\lceil \frac{(1000000+1)-3^2}{3} \right\rceil = 333{,}331$$

And so on..

For maximum speed up we can make use of multiple processors and hence time taken would be 499,999 as all the processors have to wait till the last processor completes its computation in this case the processor computing multiple of 2.

The speedup achieved by multiple processors is:

Speedup = 2,200,000 / 499,999 ~= 4.4

Q 2.
(20 points) Assume the communication network connecting the message passing multiprocessor system for solving the Sieve of Eratosthenes (using the data parallel approach taught in the class) supports concurrent message passing. Propose a faster method of communication which is better than λ (P -1).
(a) Describe the faster communication algorithm (7 points)
(b) Describe its time expression (6 points)
(c) Analyze the execution time and speedup on 1, 2, 3, ..., 16 processors of the new Sieve algorithm that will use your communication scheme, assuming n = 1,000,000 and λ = 100X . (7 points)

a)

Algorithm:

Step 1: First mark multiples of 2
Step 2: Check for the next prime p (in first iteration it will be 3)
Step 3: Send all unmarked numbers lesser than minimum (p^2, $\sqrt{n}$) to all processors as they are all primes
Step 4: Wait for sync
Step 5: Goto step 2 if we have primes yet to mark
Step 6: End

The formula for communication time in the Sieve of Erathosthenes is

Communication time = primes*(p-1)* λ

The number of processors is a fixed as well as time taken for every transaction is fixed as well. Hence, to decrease communication time we need to reduce the number of times the communication occurs.

Let us consider marking all primes under 1,000,000. Let us consider markings for the first 130 numbers on the first processor

Below is the bitmap (shows 1-130) after marking the multiples of number 2.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130

Below are the multiples of 3.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130

We can see that the markings begin from 9 ($3^2 = 9$). The numbers which are lesser than 9 are untouched by the markings done by 3. We can infer that the unmarked numbers which are below 9 are all primes.

1	2	3	4	5	6	7	8	9	10

Hence, 3, 5 and 7 can be concluded prime. We can send all the three numbers at once in order to save communication time which would otherwise take 3 different communication cycles to achieve. If all three numbers are sent to all the processors then the bitmap after marking would like below.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130

From the above table, we find that the next unmarked number is 11. As marking multiples of 11 begins from 121 ($11^2 = 121$), we can safely conclude that all unmarked numbers lesser than the number 121 are primes. Hence, the numbers 13, 17, 19, 23, 29, 31, 37 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109 and 113 are primes and can be sent in one single pass to all the processors.

Let us see the number of passes that would happen for finding primes less than 1,000,000.

Pass 1

Numbers to be sent to different processors: 2

Pass 2

Last prime processed: 2

Next prime: 3

Unmarked numbers lesser than 9 (3^2=9) = 3, 5 and 7

Pass 3

Last prime processed: 7

Next prime: 11

Unmarked numbers lesser than 121 ($11^2 = 121$) = 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71 , 73, 79, 83, 89, 97, 101, 103, 107, 109, 113

Pass 4

Last prime processed: 113

Next prime: 127

Note that $127^2 = 16129$ but marking numbers greater than 1000 is useless as $1000^2 = 1,000,000$.

Unmarked numbers lesser than 1000: 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997

Hence, we have reduced the number of passes to 3.

b)

The calculations for time expression for the new communication time is as below.

Algorithm:

Step 1: First mark multiples of 2
Step 2: Check for the next prime p (in first iteration it will be 3)
Step 3: Send all unmarked numbers lesser than minimum (p2, $\sqrt{n}$) to all processors as they are all primes (Explained below)
Step 4: Wait for sync/communication that marking has happened on all processors
Step 5: Goto step 2 if we have primes yet to mark
Step 6: End

The steps that marked in green attribute to communications. The first step which is outside the loop attributes to 1 communication while the others depend on the number n.

The time expression for the loop:

Let the numbers being found be expressed by the function f(x).

From pass 2 from above we know that f(1) marks numbers till the number 9.

f(1) = send unmarked numbers till 9($32 = 9;\ 3^{2^1}$)

f(2) marks till the number 121 (112 = 121). But as this number cannot be calculated and varies with every case we can find a upper bound on this function. To find the upper bound let us assume that for pass 3 we will be sending numbers until 81 (92 = 81) instead of 121. As we will be marking lesser numbers it will take more passes and hence more communication time than achieved. So, this would lead us to the big O notation for f.

f(2) = send unmarked numbers till 81 ($92 = 81; 34 = 81;\ 3^{2^2}$))

Similarly,

f(3) = send unmarked numbers till 6581 ($812 = 6581; 38 = 6581;\ 3^{2^3}$)

f(x) = 3^{2^x}

Considering we need to find x which indicates the number of iterations while f(x) indicates the number until which markings have happened, we will need to take log in order to find x.

Taking log to base 3 we get

$$log_3(f(x)) = 2^x$$

Taking log to base 2 we get

$$log_2 log_3(f(x)) = x$$

As we need marking until f(x) = $\sqrt{n}$, thus the final expression of marking will be

$$x = log_2 log_3(\sqrt{n})$$

Hence, the total passes can be computed by

$$1 + log_2 log_3(\sqrt{n})$$

Hence total communication time = (p-1) * λ * (x + 1), where

$$x = log_2 log_3(\sqrt{n})$$

$$P = number\ of\ processors$$

λ = *time per communication*

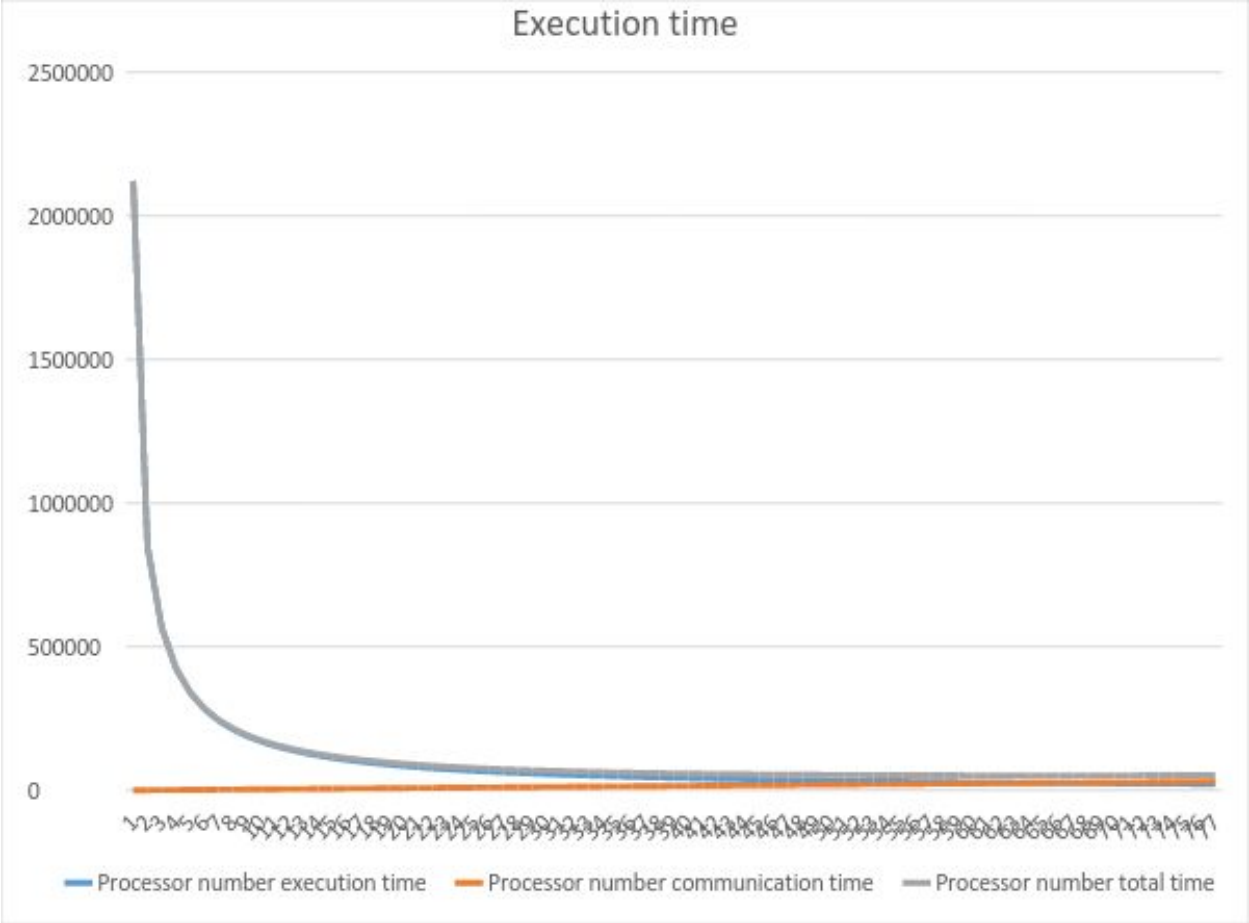
Time expression = Computation time + Communication time

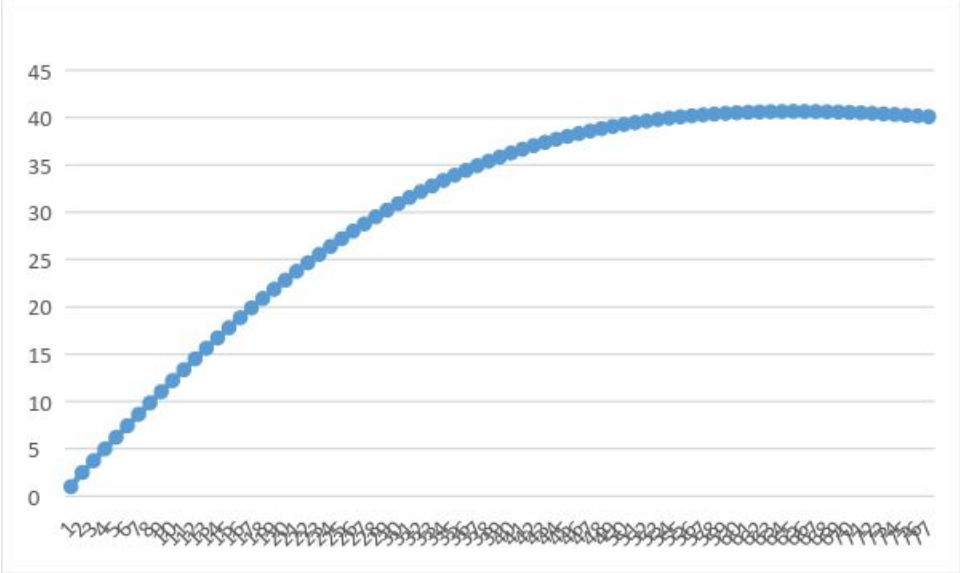
Computation time = Total number of markings per run

c)

Number Of processor	Processor number execution time	Processor number communication time	Processor number total time	Speed up
1	2122048	0	2122048	1
2	849126	800	849926	2.496744
3	566122	1200	567322	3.740465
4	424607	1600	426207	4.978914
5	339699	2000	341699	6.210284
6	283102	2400	285502	7.43269
7	242670	2800	245470	8.644836
8	212340	3200	215540	9.845263
9	188766	3600	192366	11.0313
10	169889	4000	173889	12.20346
11	154459	4400	158859	13.35806
12	141588	4800	146388	14.49605
13	130700	5200	135900	15.61478
14	121376	5600	126976	16.7122
15	113286	6000	119286	17.78958
16	106220	6400	112620	18.84255

The graph below has been extended in order to show the point at which communication time becomes more significant than processing time.


Execution time
2500000
2000000
1500000
1000000
500000
0
Processor number execution time
Processor number communication time
Processor number total time


45
40
35
30
25
20
15
10
5
0

Q 3. (20 points) Since 2 is the only even prime, one way to save memory and improve the speed of the sequential Sieve of Eratosthenes algorithm is to have the elements of the Boolean array represent only odd numbers. In this scheme, the first sieve step would mark multiples of the prime number 3. Then

(a) Estimate the reduction in execution time of the sequential algorithm resulting from this improvement for n = 1,000, and n = 1,000,000 (5 point).

(b) The improved sequential algorithm can be used as the basis for an improved data-parallel algorithm. Using the machine model of non-shared distributed memory, and assuming $\lambda = 100X$, estimate the execution time of the improved data-parallel algorithm for 1, 2,..., 16 processors (5 points)

(c) Compute the speedup of the improved data-parallel algorithm over the improved sequential algorithm. Compare this speedup with the speedup estimated for the original data-parallel algorithm (5 points).

(d) Why does the improved data-parallel algorithm achieve different speedup than the original data-parallel algorithm? (5points).

(a)

The modified calculations for the Sieve of Erathosthenes algorithm as we are only marking the odd numbers is

Formula:

Number of multiple of given prime = $\left\lfloor \frac{(n+1) - p^2}{2p} \right\rfloor$

Calculations for 1000

Number of multiple of 3 marked = $\left\lfloor \frac{(1000+1) - 3^2}{6} \right\rfloor = 166$

Number of multiple of 5 marked = $\left\lfloor \frac{(1000+1) - 5^2}{10} \right\rfloor = 98$

Total number of markings = 457

Savings in total markings = 1411 – 457 = 954

Savings in terms of previous algorithm = 954/1411 = 67.6%

Similarly, doing the calculations for 1,000,000

Total number of markings = 811,068

Savings in total markings = 1,388,932

Savings in percentage of previous algorithm = 1388932/2200000 = 63.13%

(b)
1 proc no execution time 811068X
1 proc no communication time 0
1 proc no total time 811068X

2 proc no execution time 424607X
2 proc no communication time 16700X
2 proc no total time 441307X

3 proc no execution time 283102X
3 proc no communication time 33400X
3 proc no total time 316502X

4 proc no execution time 212340X
4 proc no communication time 50100X
4 proc no total time 262440X

5 proc no execution time 169889X
5 proc no communication time 66800X
5 proc no total time 236689X

6 proc no execution time 141588X
6 proc no communication time 83500X

6 proc no total time 225088X

7 proc no execution time 121376X
7 proc no communication time 100200X
7 proc no total time 221576X

8 proc no execution time 106220X
8 proc no communication time 116900X
8 proc no total time 223120X

9 proc no execution time 94423X
9 proc no communication time 133600X
9 proc no total time 228023X

10 proc no execution time 84986X
10 proc no communication time 150300X
10 proc no total time 235286X

11 proc no execution time 77269X
11 proc no communication time 167000X
11 proc no total time 244269X

12 proc no execution time 70833X
12 proc no communication time 183700X
12 proc no total time 254533X

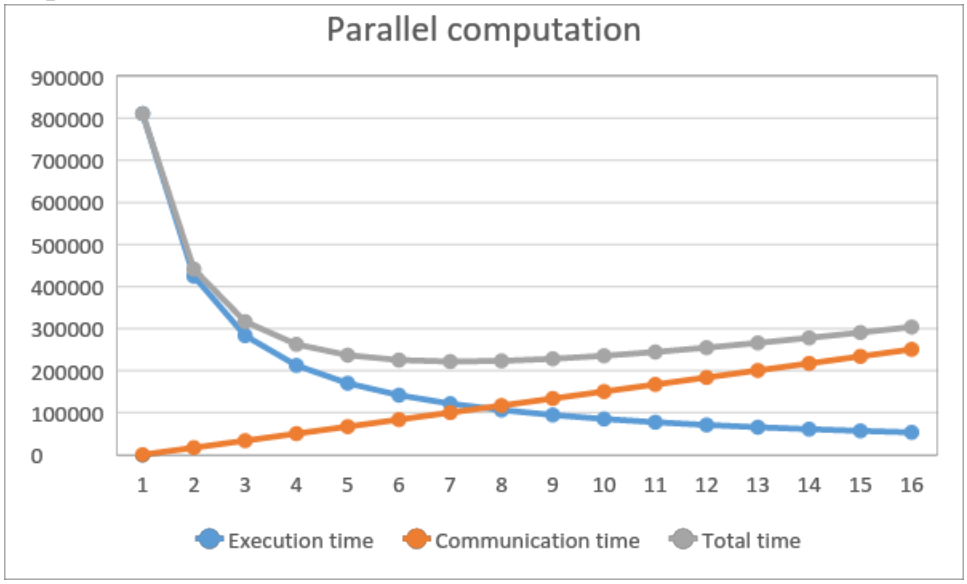
13 proc no execution time 65391X
13 proc no communication time 200400X
13 proc no total time 265791X

14 proc no execution time 60732X
14 proc no communication time 217100X
14 proc no total time 277832X

15 proc no execution time 56682X
15 proc no communication time 233800X
15 proc no total time 290482X

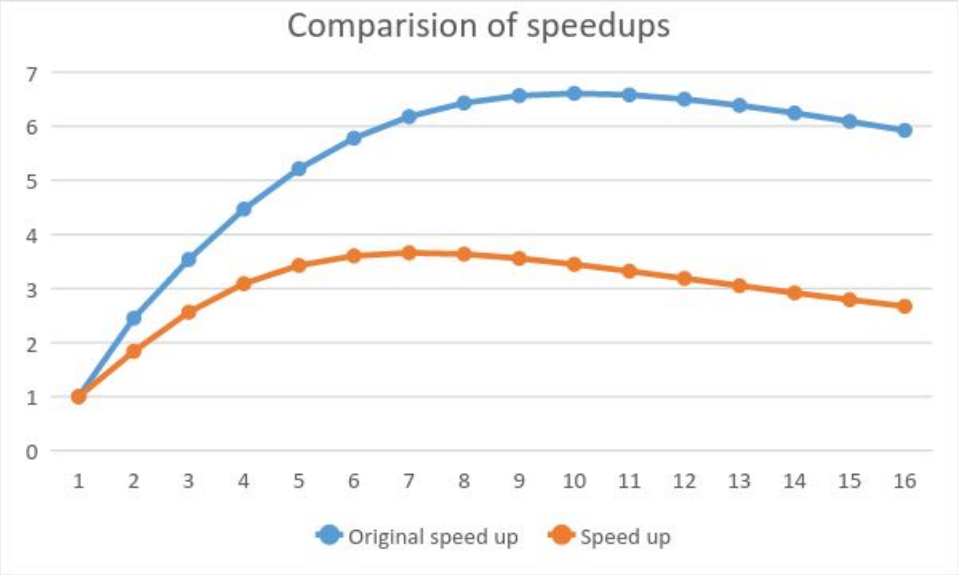
16 proc no execution time 53150X

16 proc no communication time 250500X
16 proc no total time 303650X



(c)

Original speed up	Speed up
1	1
2.450611253	1.837877
3.538386119	2.5626
4.467403638	3.090489
5.215171332	3.426725
5.780540558	3.603337
6.178263022	3.660451
6.43161787	3.63512
6.56643335	3.556957
6.608909056	3.447158
6.580830431	3.320389
6.501611579	3.186494
6.385940415	3.051525
6.245432285	2.919275
6.089335009	2.792145
5.923868014	2.671062



(d)

The speed up of the older algorithm is more than the newer algorithm as the sequential time of the actual algorithm is more than the sequential time of the modified algorithm.

Speed up = Time for single process/Time for multiple processors

Time for multiple processors = Computation time + Communication time

In the actual algorithm, the computation time was huge and hence the savings played a key role in speed up. But in the new algorithm, the computation time is lesser and hence the savings soon gets matched by the communication time. The difference in the sequential time that leads to this result is the important thing to be noted.

Q 5. (20 points) The task graph shown in figure below represents an image processing application. Each bubble represents an inherently sequential task. There are 12 tasks: an input task, 10 computation tasks, and an output task. Each of the 12 tasks can be accomplished in 1 unit of time on one processor. The input task must complete before any computational tasks begin. Likewise, all 10 computational tasks must complete before the output task begins. The input task

consumes the entire bandwidth of the input device. The output task consumes the entire bandwidth of the output device.

(a) What is the maximum speedup that can be achieved if this problem is solved on two processors? (Hint: Processors do not have to receive the message elements in order.) (4 points)
(b) What is an upper bound on the speedup that can be achieved if this problem is solved with parallel processors? (4 points)
(c) What is the smallest number of processors sufficient to achieve the speedup given in part (b)? (3 points)
(d) What is the maximum speedup that can be achieved solving five instances of this problem on two processors? Continue to assume that there is one input device and one output device. (3 points)
(e) What is an upper bound on the speedup that can be achieved solving 100 instances of this problem with parallel processors? Continue to assume that there is one input device and one output device. (3 points)
(f) What is the smallest number of processors sufficient to achieve the speedup given in part (e). (3 points)

a)
Step1 – The 1st processor gets input and 2nd processor remains idle in 1 unit of time.
Step2- The 1st and 2nd processor is used for computation in 1 unit of time.
Step3- The 1st and 2nd processor is used for computation in 1 unit of time.
Step4- The 1st and 2nd processor is used for computation in 1 unit of time.
Step5- The 1st and 2nd processor is used for computation in 1 unit of time.
Step6- The 1st and 2nd processor is used for computation in 1 unit of time.
Step7- The 1st processor outputs and 2nd processor remains idle in 1 unit of time.
Computation time for 2 processor = 1+5+1=7 units.
Computation time for 1 processor=1+10+1=12 units.

Speedup=12/7=1.71

b)
Let's assume we take multiple processors than the computation would be as follows:
Step 1 – Proc 1 gets the input for 1 unit.
Step 2 – Multiple processors can do the computation in 1 unit of time.

Step 3 – Proc 1 outputs for 1 unit.
Total computation time with 10 processors = 1+1+1=3 units

So the maximum speedup = 12/3 = 4

c)

Let's assume we take multiple processors than the computation would be as follows:
Step 1 – Proc 1 gets the input for 1 unit.
Step 2 – Multiple processors can do the computation in 1 unit of time.
Step 3 – Proc 1 outputs for 1 unit.

We need to find the minimum number of processors that are sufficient to achieve the maximum speed up. Step 1 and 3 use 1 processor each and execute only in serial. Hence, we need to use multiple processors only in step2. The minimum number of processors required to achieve maximum speedup in step 2 is 10.

Thus, the minimum number of processors used to achieve maximum speed for the entire problem is 10 processors.

d)

5 instances means: 10*5 =50 computations, 1*5= 5 input and 1*5=5 output.

There are 2 processors that are used to solve this:
Step1-The 1st processor gets the input for 1 unit.
Step2- The 1st processor and 2nd processor can do the computation in 1 unit of time.
Step3- The 1st processor and 2nd processor can do the computation in 1 unit of time.
Step4- The 1st processor and 2nd processor can do the computation in 1 unit of time.
Step5- The 1st processor and 2nd processor can do the computation in 1 unit of time.
Step6- The 1st processor and 2nd processor can do the computation in 1 unit of time.
Step7- The1st processor can do the output and 2nd processor can do input in 1 unit of time.

So on….

Hence the computation is = 1+5+1+5+1+5+1+5+1+5+1=31 units
Sequence computation =5+50+5=60 units

Maximum speedup = 60/31=1.94

e)

The 100 instance means: 100 input +1000 computation +100output.
This can be best achieved by using multiple processors as that's the maximum number of processor that be used at one time.

Proc i = Processor i (i lies in 1 to 10)
Ins j = Instance j (j lies in 1 to 100)
Proc x = multiple processors

Step 1: Proc 1 does input for Ins 1
Step 2: Proc 1 does input for Ins 2 and Proc x do computation for Ins 1
Step 3: Proc 1 does input for Ins 3 and Proc x do computation for Ins 2 and Proc 12 outputs Ins 1
Step 4: Proc 1 does input for Ins 4 and Proc x do computation for Ins 3 and Proc 12 outputs Ins 2
….
Step n: Proc 1 does input for Ins n and Proc x do computation for Ins (n-1) and Proc 12 outputs Ins (n-2)
…
Step 100: Proc 1 does input for Ins 100 and Proc x do computation for Ins 99 and Proc 12 outputs Ins 98
Step 101: Proc x do computation for Ins 100 and Proc 12 outputs Ins 99
Step 102: Proc 12 outputs Ins 100

The computation time for 100 instances with multiple processors = 102 units
The computation time for 100 instances with 1 processor = 1200 units

Speedup=1200/102=11.76

f)

Proc i = Processor i (i lies in 1 to 10)
Ins j = Instance j (j lies in 1 to 100)

The smallest number of processors we can use to achieve the same speed is 12.

Step 1: Proc 1 does input for Ins 1

Step 2: Proc 1 does input for Ins 2 and Proc 2 - Proc 11 do computation for Ins 1
Step 3: Proc 1 does input for Ins 3 and Proc 2 - Proc 11 do computation for Ins 2 and P12 outputs Ins 1
Step 4: Proc 1 does input for Ins 4 and Proc 2 - Proc 11 do computation for Ins 3 and Proc12 outputs Ins 2
….
Step n: Proc 1 does input for Ins n and Proc 2 – Proc 11 do computation for Ins (n-1) and Proc 12 outputs Ins (n-2)
…
Step 100: Proc 1 does input for Ins 100 and Proc 2 – Proc 11 do computation for Ins 99 and Proc 12 outputs Ins 98
Step 101: Proc 2 – Proc 11 do computation for Ins 100 and Proc 12 outputs Ins 99
Step 102: Proc 12 outputs Ins 100

The computation time for 100 instances with 12 processor = 102 units
The computation time for 100 instances with 1 processor = 1200 units

Speedup=1200/102=11.76

Q 5. (20 points) Describe major differences between SIMD and MIMD computers and their advantages and disadvantages over each other).

DESCRIPTION	SIMD	MIMD
Abbreviation	Single Instruction multiple data	Multiple Instruction multiple data
Definition	At each cycle all processors must execute the same instruction, and there is only one instruction stream.	Each processors can be simultaneously executing different instructions on different data.
Type	Synchronous nature.(All processors do some operations)	Asynchronous in nature.(All processors work independently)
Lower program memory requirements	One copy of the program is stored.	Each processor stores its own program.

Efficient execution of variable-time instructions	Total execution time equals the sum of maximal execution times through all processors.	Total execution time equals the maximum execution time on a given processor.
Usage of the processors	SIMD cannot act like MIMD.	MIMD can act like SIMD.
Lower instruction cost	The main processor acts as a decoder for all the processors.	The cost increases in this case as the individual processors act as a decoder.
Low PE-to-PE communication overheads	Automatic synchronization of all "send" and "receive" operations	Explicit synchronization and identification protocols needed.
Size and Performance	Scalability in size and performance.	Complex size and good performance.
Complexity of architectures	Simple	Complex
Efficient execution of variable-time instructions	The total execution time is the sum of the maximum execution time for every synchronization.	The total execution time is the maximum execution time among the processors.
Low synchronization overheads	This happens implicitly.	This requires data structures and operations for synchronization.

Advantage of SIMD:

- SIMD computers are very well suited for regular problems with large data sizes. That is, it is suitable if there are large number of operations that need can be done in parallel.
- The efficiency of SIMD depends on how many accounts are processed in parallel.
- There is no need for explicit synchronization because processors are implicitly synchronized.
- Programs are easier to write and debug.
- Generally less memory is need because of a single program.

Advantage of MIMD:

- More general and flexible in parallelizing an application.
- MIMD computers do not need the added cost associated with a global control unit.
- They are very well suited for coarse grained problems.
- MIMD can computer can simulate SIMD but converse is not true.