Problem 1

$\frac{1}{0.1} = 10ms$. $\frac{1000}{100} = 10\ ms$

The trajectory planning subsystem of NASA's Perseverance rover is highly critical and is comprised of 4 tasks. The first task performs motion course (MC) calculation every 10 ms, which takes at most 3.5 ms. The second is in charge of performing sensor fusion (SF) between inertial measurements and incoming video feed. SF operates at 100 Hz and its runtime varies anywhere between 3 and 7 ms. The third task is responsible for object detection (OD) and is activated at 25 Hz. OD can be carried out by analyzing the last batch of video frames, which takes 8 ms in the worst case. Finally, a diagnostic (DG) task has a period of 24 ms and a WCET of 12 ms.

25 | $\frac{1000}{25} = 40ms$

worst-case execution time

a) Compute the minimum number of CPUs (if you had the perfect scheduler) necessary to consolidate all the workload necessary to perform trajectory planning. (2).

b) Consider the tasks in the order in which they appear in this question. Is is possible to schedule the system via EDF-FF on a 2-CPU system?

not FF but still partitioned.

c) Is it possible to use partitioned EDF to schedule the considered taskset on 2 CPUs?

d) Consider only the MG and DG tasks. Would it be possible to schedule them on the same CPU using RM?

$2(2^{\frac{1}{2}}-1) = 0.828$ $0.35+0.5 = 0.85$

e) Is the system schedulable under Global EDF? Motivate your answer. (2 CPUs) graph.

	T (ms)	C (ms)
MC	10	3.5
SF	10	3~7
OD	40	8
DG	24	12

a).

$\frac{3.5}{10} + \frac{7}{10} + \frac{8}{40} + \frac{12}{24} = 1.75$

$U = [0.35 \quad 0.7 \quad 0.2 \quad 0.5] \leq (2)$

So, 2CPUs

b). EDF-FF.

$\beta = \lfloor \frac{1}{\frac{1}{10}} \rfloor = 1$ → $\frac{N+1}{2} = \frac{3}{2} = 1.5$.

$1.75 > 1.5$, no conclusion.

0.5 + DG (x) | OD 0.2, MC 0.35 | SF 0.7

$0.5+0.7 = 1.2 > 1$

so, no. unschedulable.

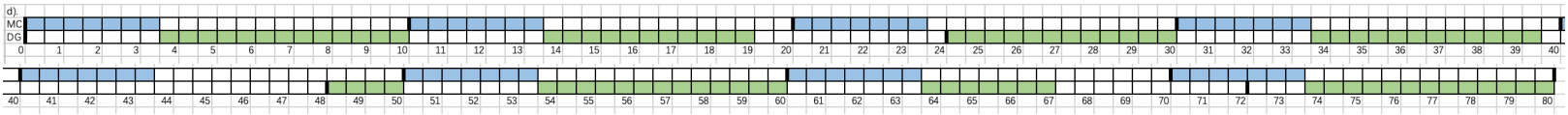
c).

DG 0.5, MC 0.35 | OD 0.2, SF 0.7

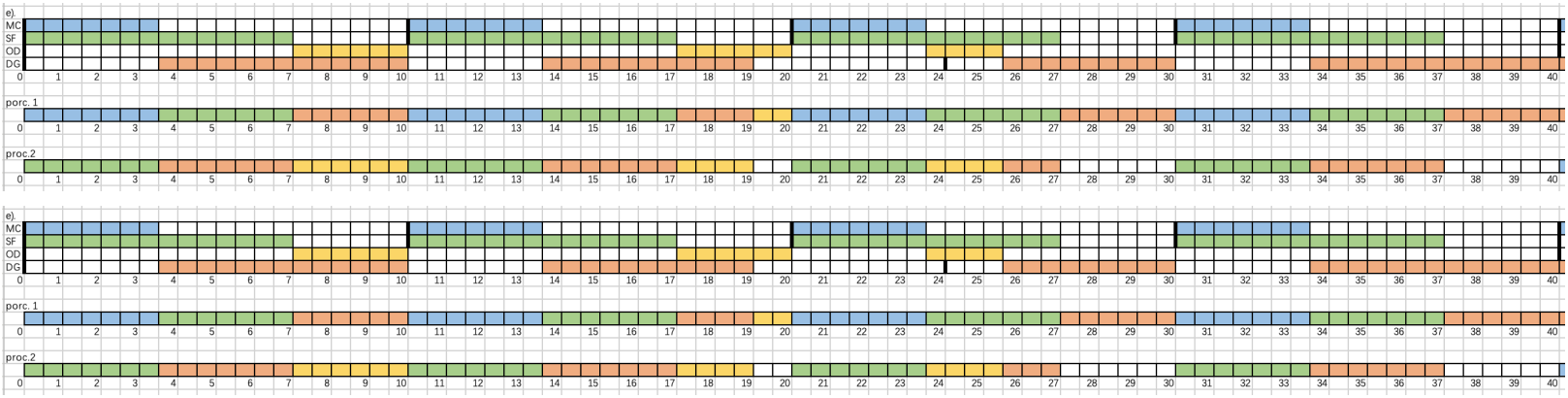
$0.5+0.35 = 0.85 < 1$ $0.2+0.7 = 0.9 < 1$

so, yes

d).


d).
MC
DG
0 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
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

e).


e).
MC
SF
OD
DG
0 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
porc. 1
0 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
proc.2
0 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

e).
MC
SF
OD
DG
0 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
porc. 1
0 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
proc.2
0 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

Problem 2

200 Hz = 1000/200 = 5ms.

A sensor node needs to acquire and pre-process the data from 4 different onboard sensors. It is crucial that pre-processing completes always before a new data sample is available at a given sensors to prevent loss of samples. The sensors are the Gyroscope (G), Magnetometer (M), Accelerometer (A), and Baroscope (B). The gyroscope produces data at a rate of 200 Hz, with each data sample requiring 2 ms for pre-processing in the worst case. Pre-processing for the magnetometer takes no longer than 1 ms with samples being produced at 125 Hz. A new accelerometer sample is produced every 19 ms and pre-processing takes at most 4 ms. Finally, pre-processing a barometer sample takes at most 6 ms, but the sampling period is configurable.

1000/125 = 8ms

a) Without drawing the schedule, perform a back-of-the-envelope calculation to determine a suitable configuration for the sampling period of the barometer such that the system always operates correctly under RM. For this part, consider only values of periods that are integers when expressed in milliseconds.

b) Because the calculation above returned a value of sampling period which is too large, you decide to set the barometer sampling period to 29. Will the system operate correctly under RM?

c) You decide to set the barometer sampling period to 30 ms and stick to RM. You notice however that once in a while (very rarely) the pre-processing of the accelerometer takes 5 ms instead of its nominal 4 ms worst-case execution time. When this happens, the system can still operate correctly if **the faulty accelerator pre-processing job is killed when it misses the deadline**. Killing any other job is not acceptable. Also, you can apply a utilization-preserving transformation to all or some of your tasks. Specifically, you can release any pre-processing task twice as fast but only perform half of the required processing in each release. Explain how you can make the system operate correctly under RM.

CPU 杀 kill

$4(2^{\frac{1}{4}}-1) = 0.757$ · $\frac{2}{5}+\frac{1}{8}+\frac{5}{19}+\frac{6}{30} = 0.936$ · $T = \frac{T}{2}$ $C = \frac{C}{2}$ $U = U$

d) Consider the same system as the question above—i.e., where the period of the barometer has been set to 30 ms. Compute the worst-case response (WCRT) time for all the jobs of each task.

e) Consider the same system as the question above—i.e., where the period of the barometer has been set to 30 ms. How much slower can the processor be while still ensuring that all the deadlines are met? no.

f) Consider the same system as the question above—i.e., where the period of the barometer has been set to 30 ms and the CPU has not been slowed down. Is the system schedulable under SJN?

	T(ms)	C(ms)
G	5	2
M	8	1
A	19	4
B	?	6

29.?

no.

a). m=4.

$4(2^{\frac{1}{4}}-1) = 0.757$

$\frac{2}{5}+\frac{1}{8}+\frac{4}{19}+\frac{6}{T_B} \le 0.757$

$= 0.7355$ $\quad 279.41 \le T_B$

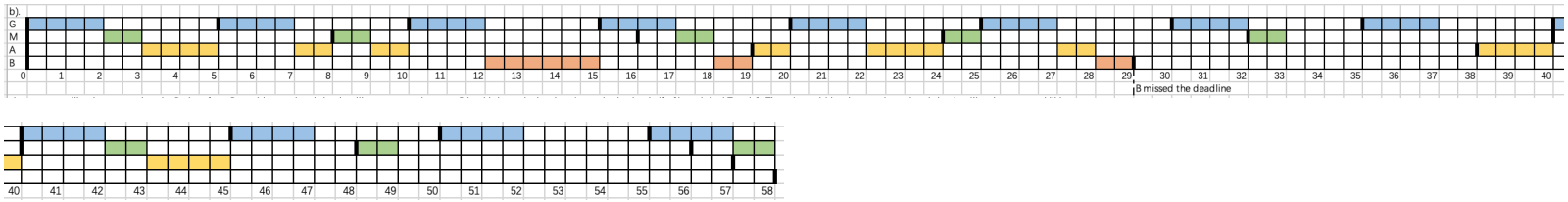
280

b). $\frac{2}{5}+\frac{1}{8}+\frac{4}{19}+\frac{6}{29} = 0.9424$

c. utilization-preserving

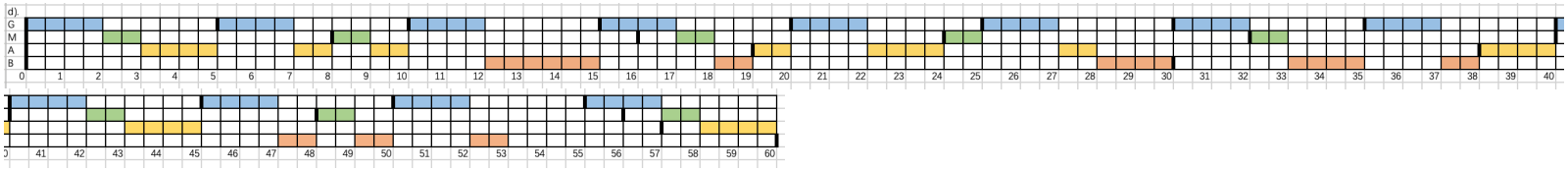
B higher priority → kill A

b).



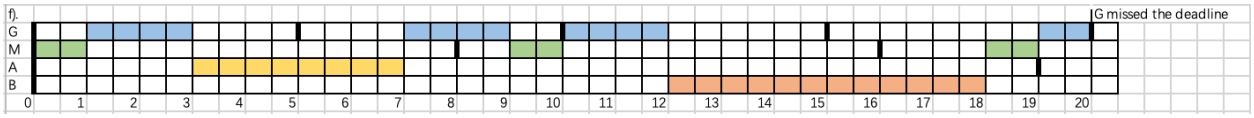
c). we can utilization–preserving the B, therefore, B would not missed the deadline anymore, because B has higher priority than A now by having half of its original T and C. Then, A would be the one that missed the deadline, but we can kill it.

d).



e). No, the processor cannot be slower, from the above graph, we can see that B just hit the deadline in the first period, if we slower the processor, then B would be missed the deadline

f).



Problem 3.

prev = queue.pop	j	tmp = array[j]	array[j] = prev	prev = temp
prev = 8	j = 0	tmp = 4	array = [**8**, **9**, 5, 3, 6]	prev = 4
	j = 1	tmp = 9	array = [8, 4, 5, 3, 6]	prev = 9
	j = 2	tmp = 5	array = [8, 4, **9**, 3, 6]	prev = 5
	j = 3	tmp = 3	array = [8, 4, 9, 5, 6]	prev = 3
	j = 4	tmp = 6	**array = [8, 4, 9, 5, 3]**	prev = 6
prev = 1	j = 0	tmp = 8	array = [1, 4, 9, 5, 3]	prev = 8
	j = 1	tmp = 4	array = [1, 4, 9, 5, 3]	prev = 4
	j = 2	tmp = 9	array = [1, 4, 9, 5, 3]	prev = 9
	j = 3	tmp = 5	array = [1, 4, 9, 5, 3]	prev = 5
	j = 4	tmp = 3	**array = [1, 8, 4, 9, 5]**	prev = 3
prev = 7	j = 0	tmp = 1	array = [7, 8, 4, **9**, 5]	prev = 1
	j = 1	tmp = 8	array = [7, 1, 4, 9, 5]	prev = 8
	j = 2	tmp = 4	array = [7, 1, 8, 9, 5]	prev = 4
	j = 3	tmp = 9	array = [7, 1, 8, 4, 5]	prev = 9
	j = 4	tmp = 5	**array = [7, 1, 8, 4, 9]**	prev = 5
prev = 2	j = 0	tmp = 7	array = [2, 1, 8, 4, **9**]	prev = 7
	j = 1	tmp = 1	array = [2, 7, 8, 4, 9]	prev = 1
	j = 2	tmp = 8	array = [2, 7, 1, 4, 9]	prev = 8
	j = 3	tmp = 4	array = [2, 7, 1, 8, 9]	prev = 4
	j = 4	tmp = 9	**array = [2, 7, 1, 8, 4]**	prev = 9

a). queue is empty after polling 2, signal(go) has been call after the polling. We haven't go through the for loop at this time, so the array is still [7, 1, 8, 4, 9]. Therefore, tot = 7 + 1 + 8 + 4 + 9 = 29

b). we can go through the for loop every call queue.pop() and see which value is the largest, and we have to go through the whole for loop every time. The four arrays we can get is [8, 4, 9, 5, 3], [1, 8, 4, 9, 5], [7, 1, 8, 4, 9], and [2, 7, 1, 8, 4]. The maximum value is 8 + 4 + 9 + 5 + 3 = 7 + 1 + 8 + 4 + 9 = 29

c). If P1, P2, S1, and S2 are all empty, which means we can let whichever process to run at anytime. Therefore, to get the maximum, the largest first element in array we can get is 8, which can be get by signaling Printer before starting running the for loop for the second time when prev = 1 (we can know the largest value for each element in the array by checking the largest value for each column, which has the same j, of all the arrays above in the graph). Then, we can signal shifter after we get 8 as our first element in array. In the same way, we can get 9 as the largest second, third, forth, and fifth element in array by signaling printer when we array[j] = 9 in our for loop (which means we can interpret the for loop and let Shifter wait). To avoid Printer adding anything in the array that is not the largest number, we can also interpret the for loop and let Printer wait until we (for) loop to the largest value in Shifter and signaling Printer to continue adding. Therefore, tot = 8 + 9 + 9 + 9 + 9 = 44