COMP 2432 Operating Systems
Written Assignment Submission to BlackBoard
Deadline: 31 March 2023

1. CPU Scheduling.

Draw Gantt charts for the following set of processes using different scheduling algorithms: (*a*) **SRT**, (*b*) **Priority with preemption** (Linux convention), (*c*) **Priority with preemption** (Windows convention), (*d*) **RR** with quantum $Q = 5$, and (*e*) **RR** with quantum $Q = 3$. Let us assume the standard tie-breaking rules when multiple processes are eligible, i.e., the process with smaller ID will go first. *Compute* the *waiting time* and *turnaround time* for each process under the algorithms (*a*) to (*e*).

Process	*Burst Time*	*Arrival Time*	*Priority*
P_1	11	0	4
P_2	8	1	2
P_3	5	3	1
P_4	12	6	5
P_5	14	8	3

(*f*) So far, we have ignored the context switching overhead during scheduling. In reality, the context swiching overhead T is non-zero. In other words, $T > 0$. Let us study the impact of this overhead T on the overall system performance. For the set of processes above, *complete* the following table for the *turnaround time* of each process in order to compare the performance with different context switching overhead T. Note that your turnaround times from (*a*) to (*e*) can be used to fill the column for $T = 0$. Do you have any observation here?

Process	SRT			PR-Linux			PR-Win			RR-5			RR-3		
	T=0	T=1	T=2	T=0	T=1	T=2	T=0	T=1	T=2	T=0	T=1	T=2	T=0	T=1	T=2
P_1															
P_2															
P_3															
P_4															
P_5															
Average															

2. Multi-level Scheduling.

A *multi-level feedback queue* is adopted to schedule the following processes for execution. Processes first enter the *high priority queue* based on **RR** with quantum 2. A process that cannot complete its execution after a service time of 4 will be demoted to the *medium priority queue* based on **RR** with quantum 4. A process that cannot complete its execution after a service time of 8 in this queue will be demoted to the *low priority queue* based on **FCFS**.

(*a*) Assume that **Fixed Priority scheduling** is adopted. *Draw* a Gantt chart for the process execution schedule. *Compute* the process *waiting* and *turnaround time* and their average.

(*b*) Assume that **Time Slicing scheduling** is adopted with a ratio of 3:4:3 in allocated time slices, in the format of 6 time units, 8 time units and 6 time units respectively on the three queues. *Draw* a Gantt chart for the process execution schedule. *Compute* the process *waiting* and *turnaround time* and their average.

Process	*Burst Time*	*Arrival Time*	*Priority*
P_1	11	0	4
P_2	8	1	2
P_3	5	3	1
P_4	12	6	5
P_5	14	8	3

3. Contiguous Memory Allocation.

A memory system currently has three holes of size 234K, 321K, and 108K and the total memory is 1000K. You are given 12 requests: 50, 88, 96, 37, 93, 35, 63, 27, 49, 68, 33, and 58K, arriving in that order. *Indicate* how the requests are satisfied with (*a*) **first-fit** algorithm, (*b*) **best-fit** algorithm, (*c*) **worst-fit** algorithm, and (*d*) an ***optimal*** algorithm that makes a decision *after* seeing all requests using an oracle. There are *multiple possible answers* for the optimal algorithm. You are to give *two* possible answers to (*d*). What are the *utilizations* for the four algorithms?

(*e*) It is found that the size of one of the holes has been *under-reported* by 1K. In other words, one of the three holes should be slightly larger, perhaps of size 235K, 321K, 108K, or 234K, 322K, 108K, or 234K, 321K, 109K. The utilization may increase under the revised configuration. In fact, it is found that the utilization indeed does increase from (*d*) when the size of that hole is corrected. *Determine* what *that hole* is, and give a *possible way* in satisfying the requests with an increased utilization from (*d*).

(*f*) You are given some tasks of size *x*K, *y*K and *z*K respectively and there are at most **9** tasks each. *Determine* the *best* that you could achieve if you are to fill up a hole of size *S*K, for the following values of *x*, *y*, *z* and *S*, by *maximizing* the space used (up to *S*), i.e. minimizing the used space, if any. *Show* how many tasks of size *x*, how many tasks of size *y* and how many tasks of size *z* are used in each of the three cases. (*g*) Now if we relax the requirement so that there are no upper limits on the number of tasks for each type, *determine* the *maximal* space usage and the *task mix*. (*h*) If we *tighten* the requirement so that there are still at most 9 tasks of each size, but we also require that each type of tasks must be used *at least once*, *determine maximal* space usage and the *task mix*.

Case	**1**	**2**	**3**
x	22	26	28
y	33	43	65
z	50	77	74
S	488	556	777

You could complete the following table. Fill in the number of tasks inside the brackets and the corresponding maximal usage.

	(*f*) At most 9 tasks each			(*g*) Unlimited number of tasks			(*h*) Strictly 1 to 9 tasks each		
Case	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
x	22×( )	26×( )	28×( )	22×( )	26×( )	28×( )	22×( )	26×( )	28×( )
y	33×( )	43×( )	65×( )	33×( )	43×( )	65×( )	33×( )	43×( )	65×( )
z	50×( )	77×( )	74×( )	50×( )	77×( )	74×( )	50×( )	77×( )	74×( )
S	488	556	777	488	556	777	488	556	777
Max usage									

4. Segmentation.

Consider the segment table for process P_1 containing the following.

P_1

Segment	*Base*	*Length / Limit*
0	1433	121
1	4431	777
2	3131	345
3	2432	304
4	5678	531
5	1002	246
6	3911	432

Suppose that **segment 3** of P_1 is a shared segment with **segment 4** of P_2, and P_2 has 7 segments of size 212, 88, 345, 511, 304, 321, and 72 respectively, starting from segment 0 to segment 6. Note that **segment 3** of P_1 has the same size as **segment 4** of P_2, since it is shared. Assume that the computer has a memory of 8KB allocated for users, with physical address from 0 to 8191. Those bytes at the lowest end of the memory as well as those at the highest end of the memory, from 0 to 999, and from 6789 to the end, are reserved by the operating system. Segments for P_2 are to be allocated from the free memory in the order of 0, 1, 2, 3, 4, 5, and 6, if necessary, using (*a*) **first-fit** algorithm, (*b*) **best-fit** algorithm, (*c*) **worst-fit** algorithm. *Show* the *three segment tables* for P_2 for segments allocated under the three algorithms by *filling* in the table below. Note that it is useful to draw diagrams showing the memory allocation to the used segments for clarity.

Segment table for P_2	*Base*			*Length / Limit*
	(*a*) **FF**	(*b*) **BF**	(*c*) **WF**	
0				
1				
2				
3				
4				
5				
6				

(*d*) *Translate* the following logical addresses for P_1 and P_2 by *filling* in the table below.

Allocation algorithm for P_2		**FF**	**BF**	**WF**
Logical address	Physical address for P_1	Physical address for P_2		
(0, 19)				
(1, 88)				
(2, 246)				
(3, 304)				
(4, 188)				
(5, 234)				
(6, 33)				