THE UNIVERSITY OF HONG KONG

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE

COMP3230A Principles of Operating Systems (Answer Book)

Date: December 09, 2020 **Time: 2:30 – 5:30pm**

You should type all your answers in this answer book and upload it to Online Exam (OLEX) system before the end of examination. The submitted file can be a MS Words file (.doc) or a pdf file (.pdf).

University No. ___________________________

Problem No.	Score
1 (30%)	/30
2 (10%)	/10
3 (20%)	/20
4 (20%)	/20
5 (20%)	/20
Total	/100

Problem 1. Multiple Choice Questions with Justifications. (30%)

Your choice(s)	Justifications and Correction
1. (c)	c. Static libraries do not have a main() function by themselves and thus cannot be directly executable.
2. ( )	
3. ( )	
4. ( )	
5. (c, d)	c. HRRN just keeps track of waiting time.
	d. HRRN is non-preemptive. It will not interrupt long tasks just because a short task arrived. HRRN selects task with highest response ratio, not shortest duration time.

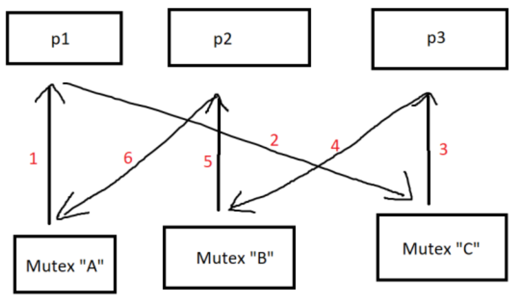
6. ( )	
7. (a,)	
8. ( )	
9. (a, c, d)	a. TLB is not a cache.
	c. Not necessary when using tags.
	d. Not necessary as they have the same address space.
10. ( )	
11. ( )	

12. ( )	
13. (a, b, c, d)	a. Thrashing starts happening when too many programs run at once.
	b. Recommended swap size is 20% of RAM.
	c. System would just crash when swap size runs out.
	d. No correlation. It will only reduce waste of CPU resources.
14. ( )	
15. ( )	

(For marking purpose: X = ___, Y= ____, Z= ____, Final Marks of Q1 = ____________)

Problem 2: (10%) Score: _____ (for marking)

Answer for (a): 5% (score: ____)



Pthread1 (p1) obtains mutex of resource “A”. Then, it tries to obtain the mutex of resource “B”, but it has been obtained by Pthread2 (p2). Pthread2 (p2) tries to obtain the mutex of resource “C”, but it has been obtained by Pthread3 (p3). Pthread3 (p3) tries to obtain the mutex of resource “A”. Thus, a deadlock occurs based on the RAG. This can happen given unfortunate scheduling of the thread execution.

(you can add more space)

Answer for (b): 5% (score: ____)

Change the arg_t into:

arg_t a1 = {&rs1, &rs3};
arg_t a2 = {&rs2, &rs3};
arg_t a3 = {&rs1, &rs2};

In this manner, deadlock will never occur. If p1 obtains mutex of Resource “A” first, it will try to obtain mutex of resource “C”. Because p1 has mutex of resource “A”, p3 cannot proceed until p1 has released its mutex of resource “A”. if p2 proceeds obtained mutex of resource “C” first, then p1 has to wait till p2 releases mutex “B”. In this way, we enforce resource access order and p1 and p3 must fight for mutex “A” first before they can obtain their second mutex.

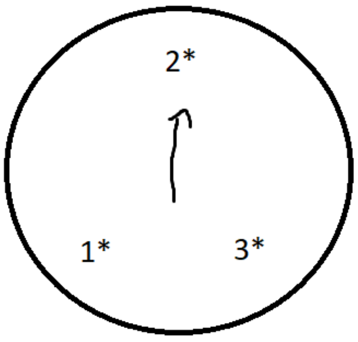
(you can add more space)

Problem 3. (20%) Score: ______ (for marking)

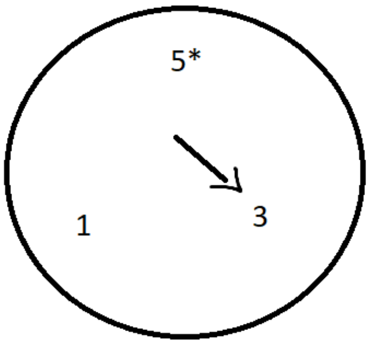
Reference string	2	3	2	1	5	2	4	5	3	5
Frame 1	2*	2*	2*	2*	5*	5*	5*	5*	3*	3*
Frame 2		3*	3*	3*	3	2*	2*	2*	2	5*
Frame 3				1*	1	1	4*	4*	4	4
Fault count	1	2	2	3	4	5	6	6	7	8

Answer for (a): 4% (score: _____)

The Clock algorithm scans the pages in a cyclic manner, starting from where it left off last time (pointing to 2*) and sets R-bit to 0 if R-bit of currently scanned page is 1. If R-bit of currently scanned page is 0, it will choose the current page as victim page and replace the current page with a new page. It scans 2*, 3*, 1* until it finally arrives back at 2 and replaces it with 5*. The new position of the clock arm is on Frame 2, pointing to 3.



Before



After

(you can add more space)

Answer for (b): 2% (score: _____)

The Clock algorithm scans the pages in a cyclic manner, starting from where it left off last time (pointing to 5*) and sets R-bit to 0 if R-bit of currently scanned page is 1. If R-bit of currently scanned page is 0, it will choose the current page as victim page and replace the current page with a new page. It scans and skips 5*, 2*, 4* until it finally arrives back at 5 and replaces it with 3*. The new position of the clock arm is on Frame 2, pointing at 2.

Answer for (c): 2% (score: _____)

8 counts.

Answer for (d): 6% (score: _____)

If all pages have R-bit of 1, the Clock algorithm must scan the entire clock and update the R-bit to 0 and return again to its position to just replace one page. This cost might get prohibitively expensive as the number of page frames is much larger than 3.

Linked List are not contiguous in memory. The items are stored in disjoint areas of memory. This results in poor locality and thus higher TLB miss rate and page fault rate. Cache lines cannot be used effectively to traverse the list. More memory is used to store the data structure, as we need extra space to keep the pointer to the next node / element.

(you can add more space)

Answer for (e): 6% (score: ____)

When a page is first added to the frame, set its R-bit value to 0, instead of 1. R-bit is set to 1 only if a page is referenced. Thus, if a page is referenced the second time, it will have a second chance, so the Clock algorithm skips through it.

(you can add more space)

Problem 4: (20%) Score: ____ (for marking)

Answer for (a): 8% (score: ____)

In Version A,
There is good temporal locality for s, namely B.
There is poor spatial locality for C.
There is poor spatial locality for A.

In Version B,
There is good temporal locality for s.
There is good spatial locality for A.
There is poor spatial locality for B.
There is good temporal locality for C.

Version B will most likely run faster, as it has better locality overall as all multidimensional arrays in C are in row-major order.

Swap size will not be an issue as swap size in workbench is huge, and multiplying matrixes should not be an issue, unless *n* is ridiculously large. In that case, we should just use a library for multiplying matrices.

(you can add more space)

Answer for (b): 12% (score: ____)

In Version B, there is poor spatial locality for B. One way we might employ to speed up its execution speed is to transpose matrix B to be able to do the multiplication by rows. This will allow us to access elements of matrix B in succession.

However, one pitfall of this potential approach is that I do not know how expensive it is to transpose B prior to the multiplication.

In essence,

Transpose(B);

```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++) {
        s = 0;
        for (k = 0; k < n; k++)
            s += A[i][k]*B[j][k];
        C[i][j] += s;
}
```

Code for transposing B:

```
for (int c = 0; i < n; c++)
    for (int r = 0; j < n; r++) {
        B[r][c] = B[c][r];
    }
}
```

(you can add more space)

Problem 5: (20%) Score: _____ (for marking)

Task 1: 10% Score: _____ (for marking)

Code segment	C code to be added by you. Leave it blank if nothing is needed.
A	sem_wait(&sem0);
B	sem_post(&sem3);
C	sem_wait(&sem1);
D	sem_post(&sem2);
E	sem_wait(&sem2);
F	sem_post(&sem0); sem_post(&sem0);
G	sem_wait(&sem3); sem_wait(&sem3);
H	sem_post(&sem1);

I	sem_init(&sem0, 0, 0);
	sem_init(&sem1, 0, 1);
	sem_init(&sem2, 0, 0);
	sem_init(&sem3, 0, 0);

Space for discussion (score: ____)

The program will ensure proper execution order, as in the beginning only makeSkeleton can proceed as all other semaphore values are 0. Then, it posts to sem2, which lets the makeBattery() method, which in turn posts twice to sem0, which results in two wheels being made. Each wheel posts once to sem3, but makeBicycle() consumes two sem3 in order to wait for both wheels to finish before making the bicycle. Lastly, it posts to sem1, allowing a new skeleton and cycle to be made again.

(you can add more space)

Task 2: 10% Score: _____ (for marking)

Code segment	C code to be added by you. Leave it blank if nothing is needed.
A	sem_wait(&sem0);
B	
C	sem_wait(&sem1);
D	
E	sem_wait(&sem2);
F	
G	

H	sem_post(&sem0);
	sem_post(&sem0);
	sem_post(&sem1);
	sem_post(&sem2);

I	sem_init(&sem0, 0, 2);
	sem_init(&sem1, 0, 1);
	sem_init(&sem2, 0, 1);

Space for discussion (score: _____)

This program would ensure the production ratio of (wheel : skeleton : battery : bicycle = 2 : 1 : 1 : 1) as the semaphores for the ratio have been initialized in section I. There would not be any resources wasted as when makeBicycle() have been called, it would wait for two wheels, one skeleton and one battery to be made before it produces the bicycle. Thus, this algorithm ensures improved production speed with maximum concurrency and maintaining the production ratio due to how the algorithm behaves.

When the threads are initialized, two wheels can be produced and one skeleton and one battery concurrently, but they can't proceed anymore until the bicycle has been assembled and has increased the semaphore values of sem0, sem1, and sem2 to 2, 1, and 1 respectively.

(you can add more space)

--------------------------------------- END OF PAPER ---------------------------------------