NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

FINAL ASSESSMENT FOR
Semester 1 AY2018/2019

CG2271 – Real-Time Operating Systems

Nov/Dec 2018 **Time Allowed: 2 hours**

INSTRUCTIONS TO STUDENTS:

1. This assessment paper contains **FOUR (4)** questions and comprises **TWENTY (20)** printed pages, including this page.
2. Students are required to answer ALL the questions.
3. Write your answers within the space provided. Answers written on other parts of the answer script will not be graded unless you specify explicitly.
4. This is a **CLOSED BOOK** assessment.
5. Allow a help sheet in a single A4 paper, and calculator.
6. Please write your student number below only. Do not write your name.

Student Number: ______________________

This portion is for examiner's use only

Question	Marks	Remarks
Q1 [6 marks]		
Q2 [18 marks]		
Q3 [28 marks]		
Q4 [28 marks]		
Total [80 marks]		

QUESTION 1: Indicate whether the following statements are true or false. Justify your answers. [6 marks]

(A) If the utilization of a task set is lower than one, the task set is definitely schedulable under a dynamic-priority scheduling algorithm.

[1 mark]

(B) If the resource allocation graph contains more than one cycle, a deadlock must occur in the system.

[1 mark]

(C) Data inconsistency does not occur, if processes do not have shared data.

[1 mark]

QUESTION 2: Short Q&A [18 marks]

(A) What is priority inheritance and why is it important?

[3 marks]

(B) What is a "page fault" and what actions are taken by an operating system when a page fault occurs?

[3 marks]

(D) External fragmentation may occur if memory allocation uses variable sizes.

[1 mark]

(E) In a multi-processing system that is executing multiple tasks, if the total input size of all the tasks is smaller than the main memory capacity, then page faults will never happen.

[1 mark]

(F) Using Translation Look-aside Buffer (TLB) will always reduce the effective memory access time.

[1 mark]

(C) Rate Monotonic Analysis (RMA) can give a necessary and sufficient condition for a feasible schedule. Why don't we use RMA for any input task set to determine whether the input task set is schedulable or not?

[3 marks]

(D) When we learnt the Rate Monotonic Scheduling (RMS) algorithm, we assume that we have a "real time operating system with negligible overheads". (1) What will these overheads come from? (2) How will they impact your RMS analysis?

[3 marks]

(E) Consider a system with virtual memory paging support. The hard disk is used as paging disk. CPU utilization measures the ratio of the time of executing instructions from the running processes to the total CPU time. Bandwidth utilization of an I/O device is the percentage of the device's bandwidth that is currently being consumed by I/O traffic. At some time, we observed the following time-measured utilizations:

CPU utilization 10%

Paging disk bandwidth utilization 98.7%

Bandwidth utilization of other I/O devices 0.5%

For each of the following actions, describe whether it will (probably) improve CPU utilization, and then briefly justify your answer.

[6 marks]

(1) Install a hard disk with bigger capacity.

(2) Kill the process that is consuming the highest paging disk bandwidth.

(3) Create more new processes.

QUESTION 3: Calculations [28 marks]

(A) Suppose that the p=calloc(n, b) requests a consecutive memory space of n consecutive units of memory each b bytes long, and it returns a pointer p that pointing to this memory space. free(p) frees the memory pointed to by p, and sizeof(x) returns the size of variable x in bytes.

A char variable is 1 byte long, and a double variable is 8 bytes long.

We have a memory system that consists of 256 bytes in total and memory is allocated in units of 16 bytes (e.g., even requesting for 1 byte will still return a 16-bytes of memory). We assume starting at address 0. Consider the following program fragment:

```
double *A = calloc(15, sizeof(double));
double *B = calloc(15, sizeof(double));
char *C = calloc(10, sizeof(char));
free(B);
```

(1) We are using linked-lists to manage free memory. Sketch the linked-lists after each memory operation for a **best fit** policy. In each sketch, you need to indicate a memory region associated with the variable name and the memory size. Replace the variable name with "unused" if a memory region is unused. Indicate FAIL if you are unable to perform the operation shown, and explain why the operation fails. Please show the sketch of each memory operation.

[2 marks]

double *A = calloc(15, sizeof(double));

<Draw your sketch here>

double *B = calloc(15, sizeof(double));

<Draw your sketch here>

```
char *C = calloc(10, sizeof(char));
```

<Draw your sketch here>

```
free(B);
```

<Draw your sketch here>

(2) With the reference to (1), will the sketch be different for a worst fit policy? Justify your answer.

[2 marks]

(B) Consider a memory system with 12-bit virtual address, 512-byte main memory, and 64-byte page size.

(1) Fill in the blanks in the following table.

[4 marks]

Number of bits for page offset	
Number of bits of physical address space	
Maximum number of physical pages	
Maximum number of virtual pages	

(2) Consider the below memory accesses in a sequence. Translate the following virtual addresses to physical addresses. Draw the page table to show the address translation and indicate if there is any page fault. Assume that you will start with an empty page table. When there is a page fault caused by a virtual page access, you will first choose any of the unmapped physical pages to map to this virtual page. Otherwise, you can choose any physical page to be mapped to this virtual page for simplicity.

Access time	Virtual addresses
1	0
2	513
3	32
4	2049

[4 marks]

(C) There are three tasks as shown below. Assume all of them are ready at time 0. The deadline for each task is the same as the period.

Task	Worst Case Execution Time	Period
T1	2	8
T2	3	10
T3	5	20

(1) Is T3 RM-schedulable? Justify by the critical instant analysis.

[4 marks]

(2) Give the EDF schedule of this task set for the first 20 units of time.

[4 marks]

(D) Consider the following snapshot of a system's state, with four processes and three resource types (A, B and C). The available numbers of instances in A, B and C are 2, 1 and 1, respectively. The current Allocation and Max matrices are shown in Table Q3a. Suppose P2 and P3 issue the same requests with the request vector of (1, 0, 1). Assume that P2's request comes earlier than P3's request. How does the system handle those requests so that the system will not go into unsafe state?

Table Q3a

	Allocation	Max
Process	A B C	A B C
P0	2 0 0	2 2 4
P1	1 1 2	4 1 3
P2	1 1 5	3 1 6
P3	0 0 1	2 2 3

[4 marks]

(E) Given a set of periodical tasks with identical timing characteristics. They have the same execution time, period (equal to deadline). Suppose there are N tasks, and the period of each is P. What is the maximum execution time for each task so that this task set can be scheduled using Earliest Deadline First scheduling, assuming that all system overheads such as context switches are negligible.

[4 marks]

QUESTION 4: Design and Analysis [28 marks]

(A) The following solution attempts to solve critical section problem for two processes. The solution uses a shared variable "flag" declared as:

```
        int  flag[2];
```

which are initialized to be 0. The program for two processes are as follows:

```
Process 0:                                 Process 1:
while (1) {                                while (1) {
        while (flag[1] == 1) ;                     while (flag[0] == 1) ;
        flag[0] = 1;                               flag[1] = 1;
        critical section;                          critical section;
        flag[0] = 0;                               flag[1] = 0;
        remainder section;                         remainder section;
};                                         };
```

Can this solution solve the critical section problem for Process 0 and Process 1? Explain your answer.

[6 marks]

(B) Operating systems may have different implementations on buddy memory management systems. For instance, the Linux kernel uses a so-called binary buddy scheme for managing a machine's memory. The binary buddy scheme is a very simple allocator that uses a strict memory allocation scheme. The kernel keeps 10 free lists for blocks of size 4KB, 8KB, 16KB, and so on to 1MB. The memory allocation scheme is strict in that all blocks on a free list are of the same size. Allocation requests are rounded up to the next available free block size. If the free list for this size is empty, the free list of the next higher size is consulted until a non-empty free list is found. If no free list is found, an exception will be returned. The splitting policy splits blocks in half and adds blocks to the appropriate free lists. For instance, suppose the 4KB, 8KB, and 16KB free lists are empty and a request for 3KB arrives. In this case, a 32KB block would be split into a 16KB block, an 8KB block, and 2 x 4KB blocks. One 4 KB block would be used to satisfy the allocation request, and the other would go on to the 4KB free list, and the 16KB and 8KB blocks would go onto their respective free lists.
Buddy allocators perform immediate coalescing. However, a block can only be coalesced with its buddy, which is of the same size and is located before or after it in the memory space, depending on the block's address. For instance, the buddy of the 4KB block at 0x0000 is the 4KB block at 0x1000. The buddy of the 8KB block at 0x2e000 is the 8KB block at 0x2c000. If the buddy of a block of size N is free, the coalesced block will be added to the free list of blocks of size 2*N. In this way, all blocks of size N are aligned at multiples of N.

(1) Analyse the time complexity of its described allocation and free operations.

[2 marks]

(2) Analyze one advantage and one disadvantage of the immediate coalescing scheme, in comparison with the scheme that we periodically perform coalescing.

[2 marks]

(3) How much internal fragmentation would you expect from this scheme?

[3 marks]

(4) How will the coalescing scheme affect the amount of external fragmentation?

[3 marks]

(C) A programmer Jim is going to implement a system with two processes, P0 and P1. P0 will run on a machine, and P1 will run on another machine. The pseudo program for two processes are as follows:

```
Process P0:                     Process P1:

        ...                             ...
        Code segment A                  Code segment B
        ...                             ...
```

Due to the data dependency, code segment A in P0 must be executed before code segment B in P1. Should Jim use shared memory or message passing to ensure this execution order? Justify your answer and sketch your implementation by adding synchronization codes into the above pseudo program of P0 and P1.

[4 marks]

(D) Consider a bank branch with one service counter and ten client seats. Both the clerk at the service counter and clients are modelled as processes, as shown in Table Q4a.

When a client arrives, she first takes a service number from the kiosk. Note that the kiosk can serve only one client at a time. Then, she queues and waits for a seat. Whenever a seat is available, one client in the queue will move to take the seat, and continue to wait for service. The clerk will call the service numbers one by one and serve the corresponding client. Once the service number is called, the client will leave the seat and move to the service counter. Use semaphores to implement the synchronization between these two kinds of processes: Clerk and Client, as shown in Table Q4a. You are only allowed to add semaphore definitions and semaphore operations (P and V) into the pseudo code of Table Q4a. Your implementation should be deadlock free.

Table Q4a

Process Clerk:	Process Client i:
While (1) {	Get the service number;
Call the service number;	Wait for a seat;
Serve the client;	Wait for her turn to be served;
}	Get the service;

[8 marks]

BLANK PAGE
END OF ASSESSMENT