ROYAL HOLLOWAY, UNIVERSITY OF LONDON
BSc EXAMINATION 2021

CS2850: Operating Systems
CS2850R: Operating Systems – for FIRSTSIT/RESIT CANDIDATES

Time allowed: **TWO hours**

Please answer **ALL** questions.

- Handwrite your answers on paper, and write your candidate number and the module number at the top of each page. Photograph/scan the pages and keep the original paper versions, as they may be required by the examiners.
- For each question you attempt, please clearly state the question number.
- Please DO NOT include your name or Student ID anywhere on your work.
- **Academic Misconduct:** We will check all assignments for academic misconduct. Suspected offences will be dealt with under the College's formal Academic Misconduct procedures. Please remember:
  - The work submitted is expected to be your own work and only your work. You may not ask for help from any source, or copy anyone else's work.
  - You must not give help to anyone else, including sending them any parts of the questions or copies of your solutions.
  - You must not discuss the questions or solutions with anyone else.
- **Submitting your work:**
  - Your document must be submitted through Moodle using the submission link in the module Moodle page. If possible please convert your document into a PDF document to make the submission process quicker and easier.
  - Emailed submissions will not be accepted.
  - **You must complete your exam upload within 1 hour of the exam finish time.**

1. (a) Consider the following table that shows the time of loading and the R and M bits for each memory page (the times are in clock ticks). Assuming a page fault occurs at clock tick 50, indicate which page would be replaced by the Second Chance algorithm and describe what else would be updated in the information present in this table. [6 marks]

Page	Loaded	R	M
1	18	0	1
2	40	1	0
3	22	0	1
4	7	1	1

(b) In a mainframe system without multiprogramming (i.e., programs cannot be run concurrently), consider four jobs A,B, C, and D. Suppose that their estimated running times are:

	running time (minutes)
A	20
B	4
C	3
D	14

Give the turnaround times for each of these jobs and the average turnaround time in the context of the following batch system scheduling strategies, making sure to show all your working.

i. First-Come First-Served (FCFS), assuming the order A, B, C, D. [5 marks]

ii. Shortest Job First (SJF). [5 marks]

(c) Consider the program that consists of the following two concurrent processes that share the variable `x`:

```
P1:   x = 1;                    P2:  x = 16;
      if(x > 9)                      if(x < 12)
        x = x - 6;                     x = x + 14;
```

How many different values may the variable `x` have when the program terminates? Give a possible execution sequence for each case. [8 marks]

NEXT PAGE

2. Consider the following proposed solution to the mutual exclusion problem for two processes, using a shared array `s` of two integers. We assume the existence of two distinct integer constants `IN` and `OUT`; initially `s[0]` and `s[1]` are both set to `OUT`.

```
P0:
while(1) {
    while(s[1] == IN);
    s[0] = IN;
    critical_region0();
    s[0] = OUT;
    non_critical_region0();
}
```

```
P1:
while(1) {
    while(s[0] == IN);
    s[1] = IN;
    critical_region1();
    s[1] = OUT;
    non_critical_region1();
}
```

(a) Briefly describe the operations of the algorithm, giving particular emphasis to any dependencies between the two processes. [5 marks]

(b) Give one example of execution order that shows the above algorithm does not fully comply with the conditions required for solving the mutual exclusion problem. [4 marks]

(c) i. Write a revised version of the program using a semaphore. Your solution should fully comply with the conditions required for solving the mutual exclusion problem. You can use pseudo-code in your answer. [5 marks]

ii. What is gained by using a semaphore? [4 marks]

(d) Could the barrier synchronisation mechanism be used to achieve mutual exclusion in the above program, instead of introducing a semaphore? Justify your answer. [4 marks]

3. Consider a virtual memory system with 13-bit virtual addresses, 12-bit physical addresses, and page size 1K.

   (a) How many virtual pages are there in the virtual address space of a process? How many physical pages are there in the physical memory of the system? Justify your answers. [4 marks]

   (b) Consider the following page table:

Virtual Page Number	Physical Page Number	Present
7	-	0
6	01	1
5	11	1
4	-	0
3	00	1
2	-	0
1	10	1
0	-	0

Which physical address is the following virtual address translated to?

1010010001100

Explain how this translation is carried out by the MMU. [7 marks]

4. In the context of deadlock avoidance algorithms, give one example of a safe state and one example of an unsafe state in the following scenario:

   - four processes are active and each of these needs to use at least one additional resource from the system (for each state, you decide how many resources each process is holding and the maximum each will need);
   - only one type of resource exists, with five resources available in the system in total.

   Represent the examples in tables. Each row should show the information for a process. The columns should show the number of resources held, and the maximum number of resources simultaneously required by each process.

   In addition, for each example, demonstrate why it is a safe/unsafe state. [8 marks]

5. **Generating processes with `fork()` (20 marks)**

```
1 #include <unistd.h>
2 #include <sys/wait.h>
3
4
5 int N = 5;
6
7 int main(){
8
9     int i = 0;
10
11     while (i<N){
12
13         i++;
14
15         if (!fork()){
16
17             return i;
18
19         }
20
21         wait(NULL);
22
23     }
24
25     return 0;
26
27 }
```

Read the C code above and answer the following questions:

(a) What does the program do? Draw a diagram that visualizes what happens when the code runs. Explain what is the role of the function call

`wait(NULL);`

on line 21, what is its return value, and how many times it is called. [4 marks]

(b) What is the total number of processes generated when you run the code by setting `N = 1, 2, 4, 10` on line 5? Justify your answers. [3 marks]

(c) Explain what happens if the statment

`i++;`

is moved from line 13 to line 16. [2 marks]

NEXT PAGE

(d) What is the value of `i` when the parent process returns? Justify your answer. [2 marks]

(e) How can you make the program print the process identifier (`PID`) of the child processes? Write the needed instructions and specify the line where you would insert them. [2 marks]

(f) How can you make the program print the (virtual) address of the integer variable `i` for each running process? Do you expect the memory address printed on the terminal to be the same in all processes? Briefly justify your answer. [2 marks]

(g) How could you make the child processes communicate their return value to the parent? Do you need to set up an anonymous pipe for this? [2 marks]

(h) Briefly explain how `fork` can be used to create an interactive program (max 2-3 lines) [3 marks]

6. **A simple linked list (15 marks)**

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define N 10
5
6 struct node{
7     int label;
8     struct node *prev;
9 };
10
11 int main() {
12     struct node *head = NULL;
13     struct node *tail = NULL;
14     struct node *cur = NULL;
15     for (int i=0; i<N; i++) {
16         cur = malloc(sizeof(struct node));
17         if (i==0) tail = cur;
18         cur->label = i;
19         cur-> prev = head;
20         head = cur;
21     }
22     tail->prev = head;
23
24     struct node *temp;
25     while(cur != tail){
26         temp = cur;
27         cur = cur->prev;
28         printf("%d-th node \n", temp->label);
29     }
30     printf("%d-th node\n", cur->label);
31     free(cur);
32     return 0;
33 }
```

Consider the code given above and answer the following questions:

(a) Briefly explain what the program does. In particular, explain why three pointers to `struct`, `head`, `tail`, and `cur`, are needed and what is the role of the statements

`head = cur;`

on line 20 and

```
tail->prev = head;
```

on line 22. [8 marks]

(b) The output of the program is

```
9-th node
8-th node
7-th node
6-th node
5-th node
4-th node
3-th node
2-th node
1-th node
0-th node
```

and the allocated memory is not freed correctly. Add a few statements to the code above (below line 23) so that no memory leaks are possible and the output of the program is

```
2-th node
1-th node
0-th node
9-th node
8-th node
7-th node
6-th node
5-th node
4-th node
3-th node
```

[7 marks]

END