UNIVERSITY OF LONDON

BSc EXAMINATION 2019

For Internal Students of
Royal Holloway

DO NOT TURN OVER UNTIL TOLD TO BEGIN

CS2850: Operating Systems
CS2850R: Operating Systems – PAPER FOR FIRST SITS/RESIT CANDIDATES

Time Allowed: **TWO hours**

Answer ALL questions
Calculators are NOT permitted

1. (a) Modern CPUs use a pipeline in order to increase performance.

   i. What are the three main unit types in this pipeline? [3 marks]

   ii. What is a superscalar CPU? [2 marks]

   (b) Briefly explain what is memory paging and what is its main purpose. [6 marks]

   (c) Describe the Second Chance page replacement algorithm. [4 marks]

   (d) Give two advantages and two disadvantages of using user-level threads rather than kernel-level threads. [4 marks]

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

```
P1:   x = -1;
      x = x + 2;
```

```
P2:   x = 0;
      if(x < 0)
        x = x - 2;
      else
        x = x + 4;
```

   How many different values may the variable `x` have when the program terminates? Give a possible execution sequence for each case. [6 marks]

2. (a) There are four properties that are required for any correct algorithm to solve the mutual exclusion problem. Explain the importance of the property "No assumptions may be made about speeds or the number of CPUs." in this context. [6 marks]

   (b) Consider the following proposed solution to the mutual exclusion problem for processes P1 and P2, where the variable `turn` is initially set to `1`:

```
P1:  while(1) {
       while(turn == 1) {
         critical_region1();
         turn = 2;
         non_critical_region1();
       }
     }

P2:  while(1) {
       while(turn == 2) {
         critical_region2();
         turn = 1;
         non_critical_region2();
       }
     }
```

   i. Briefly describe the operation of the algorithm, giving particular emphasis to any dependencies between the two processes. [5 marks]

   ii. Explain why the above algorithm does not fully comply with the conditions required for solving the mutual exclusion problem. [5 marks]

   iii. What could be gained with a revised version of this program using a semaphore? [2 marks]

   (c) Briefly describe Peterson's algorithm for solving the mutual exclusion problem for two processes. [7 marks]

3. (a) Briefly describe the monitor synchronisation primitive. Give an advantage of using monitors instead of semaphores. [6 marks]

   (b) What is an i-node in the context of file systems? Explain what is gained by using i-nodes instead of a file allocation table (FAT).

   [6 marks]

   (c) In the context of deadlock avoidance algorithms, what is the difference between a safe state and an unsafe state?

   [7 marks]

   (d) In the context of Cloud Computing, the National Institute of Standards and Technology (NIST - USA) determines the following as essential characteristics of a “cloud”:

   i. On-demand self-service
   ii. Broad network access
   iii. Resource pooling
   iv. Rapid elasticity
   v. Measured service

   Select **three** of these characteristics and briefly describe them.

   [6 marks]

4. (a) Consider the following C program.

```
#include <stdio.h>
int main() {
  int a, b;
  int *p, *q;

  a = -8;
  b = 3;
  printf("%d %d\n", a, b * 3); // i
  p = &a;
  *p = b;
  printf("%d %d\n", a, b);     // ii
  q = &b;
  b = 0;
  printf("%d %d %d\n", *p, *q, b);    // iii
  p = q;
  *p = 4;
  p = &a;
  *p = 21;
  printf("%d %d %d\n", *p, a, b);     // iv
}
```

Write down what each of the statements labelled i – iv is going to print when the program is run. [8 marks]

(b) A programmer has written the code below to execute two functions "task1" and "task2" in two child processes of the main process. After waiting for the child processes to finish, the main function is supposed to indicate to the user that both tasks are done.

Upon testing the program, the programmer finds that "`Tasks finished!`" is output four times instead of just once! Explain this erroneous behaviour in detail, and modify the code so that it behaves correctly.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int task1() { ... }
int task2() { ... }
```

```
int main(int argc, char *argv[]) {
    if (fork() == 0) {
        task1();
    }
    if (fork() == 0) {
        task2();
    }
    wait(NULL);
    wait(NULL);
    printf("Tasks finished!\n");
    return 0;
}
```

[5 marks]

(c) Consider the following definition of a node of a linked list of strings:

```
struct node {
  struct node *next;
  char *s;
};
```

Write a C function `char *concat_all(struct node *nd)`, which returns a new string that is the concatenation of all individual strings in the list starting at node `nd`.

Note that your solution must allocate enough memory to hold all characters of all strings, for lists of arbitrary length. The order in which the strings are concatenated does not matter.

In your code, you may use the standard library functions `strlen(char *s)`, `strcat(char *to, char *from)`, `strcpy(char *to, char *from)`, `malloc(size_t n)`, and `realloc(void *ptr, size_t n)`. [12 marks]

END