Where an algorithm is asked for, you may write in any suitable pseudocode. Correct syntax for any computer language is not expected.

Answer 3 questions.

1) A system for dealing with insurance claim forms submitted by people with policies sold by the ACME insurance corp. is broken up into several sections, and has the following properties:

    - The average individual will submit one claim form every 10 years.
    - All forms arrive at the dispatcher. On average the dispatcher takes 1 day to do anything with a form.
    - At any given point in time a form in the possession of the dispatcher has:
        - a 65% chance of being returned to the claimant.
        - a 10% chance of being sent to the deputy director in charge of form checking. She takes 4 days to check them on average and forwards them to the director in charge of form checking.
        - a 5% chance of being sent directly to the director of form checking. The director of form checking takes 3 days to check any form, on average. The form is then returned to the dispatcher.
        - a 20% chance of being sent by the dispatcher to the accountant in charge of payment. The accountant takes 3 days to deal with a form and then sends it back to the dispatcher.

    a) Draw and label a queuing diagram of the form checking process [6]

    b) Determine the bottleneck in the system [9]

    c) If the dispatcher is busy 90% of the time, what is the average response time if there are 2,200 people with ACME insurance policies? [9]

    d) Assume that the system is running at optimum throughput. How many people could the system support if the average time a form spent with the dispatcher was reduced to ½ day, and the average response time should not exceed 50 days? [9]

[TURN OVER]

2)

a) Digital signatures have different properties to traditional signatures placed on a document with a pen. What is a digital signature and what are the functional differences between them and those written in ink? [4]

b) In PGP, an MD5 hash value is used to produce digital signatures. Why are hash values used, and what properties do we require of such a hash value. [5]

c) What threats are there to data being passed over insecure links, and how can they be avoided? [12]

d) Present and explain an algorithm by which a user may authenticate themselves to a computer system using public key cryptography. You should show that your algorithm avoids the threats you identified in c), or else demonstrate that this is not possible. [12]

3)

a) Explain the operation of a multilevel paging system. Use as an example a system with a 32 bit address, 4K word pages, and a 2 level paging system with page tables of equal size at each level. Your answer should include details of the sizes of page tables and the way that you calculated these [11]

b) Given the above system and assuming a memory size of 8 Mbyte , explain what happens when the following program is run.

```
int array[4096][4096]

for (int i = 0; i < 4096; i++)
  for (int j = 0; j < 4096; j++)
    array[i][j]++
```

[3]

c) What happens when, instead, the following program is run:

```
int array[4096][4096]

for (int j = 0; j < 4096; j++)
  for (int i = 0; i < 4096; i++)
    array[i][j]++
```

[3]

d) If all pages are initially on disk, memory access time is 200ns, and disk access time = 2ms + (250ns/byte transferred), what is the time to run each of the programs above programs? State any assumptions you make [10]

e) What is a TLB and how could one improve the running time in the above cases? Assume a TLB access time of 1ns and a 512 Kbyte cache size, what running times would one now expect for the above programs? Comment on your results. [6]

[CONTINUED]

4)

a) What three criteria must be satisfied in any solution to the critical section problem, and why? [6]

b) The following are two attempts at solving the two process critical section problem, using only an assumption that read and assignment operations are atomic. With reference to your answer to part a), say how the solutions are deficient. [16]

```
//
// Solution A
//
int turn = 1   // Global variable shared by both processes

process p1:                                        process p2:
while (TRUE)                                       while (TRUE)
{ while (turn == 2)                                { while (turn == 1)
    no_op                                              no_op

  CRITICAL SECTION                                   CRITICAL SECTION

  turn = 2                                           turn = 1

  NON CRITICAL SECTION                               NON CRITICAL SECTION
}                                                  }


//
// Solution B
//
int c1 = FALSE   // Global variables shared by both processes
int c2 = FALSE

process p1:                                        process p2:
while (TRUE)                                       while (TRUE)
{ c1 = TRUE                                        { c2 = TRUE
  while (c2 == TRUE)                                 while (c1 == TRUE)
  { c1 = FALSE                                       { c2 = FALSE
    sleep (10 milliseconds)                            sleep (10 ms)
    c1 = TRUE                                          c2 = TRUE
  }                                                  }

  CRITICAL SECTION                                   CRITICAL SECTION

  c1 = FALSE                                         c2 = FALSE

  NON CRITICAL SECTION                               NON CRITICAL SECTION
}                                                  }
```

c) Using only the assumption that read and write operations are atomic, give a correct solution to the 2-process critical section problem and say why it is correct. [11]

[TURN OVER]

5)

a) What metrics could one use to measure the performance of a scheduling algorithm? [5]

b)

i) What is the convoy effect and why might it be a problem? Illustrate your answer with an example. [4]

ii) Show how a different scheduling algorithm of your choice overcomes the problem. [4]

iii) Is this second scheduling algorithm better in all circumstances? Justify your answer. [4]

c) On a particular computer, the operating system has a context switch time of 1ms, and a preemptive scheduling system with three priority levels, A, B, and C. Each priority level has a separate queue associated with it, all scheduled using round robin, with the constraint that if there is a process in the level A queue it is scheduled in preference to anything in levels B and C, and a process in level B is scheduled in preference to anything in level C. In level A, the timeslice is 20ms, in level B it is 40ms, and in level C it is 80ms. Processes initially arrive at level A but their priority may change according to the following two rules:

i) If a process uses its entire timeslice, it sinks one level, unless already at level C when it remains there.

ii) After 100ms, any process which has not run in the previous 100ms is promoted by one level, unless already in level A.

iii) After returning from i/o, a process re-enters the level A queue.

iv) If a process is preempted, it does not change its place in the queue.

Given the process arrivals below, and assuming that each process executes in an endless sequence of CPU and i/o bursts, draw a Gantt chart to show how this system performs over the first 202ms. [16]

Process	P_0	P_1	P_2	P_3	P_4
CPU burst	5	10	50	100	5
i/o burst	20	10	10	10	50
Arrival time	0	10	20	30	40

[END OF PAPER]