## Question 1

With regard to process synchronisation describe what is meant by race conditions?

(5)

Describe two methods that allow mutual exclusion with busy waiting to be implemented. Ensure you state any problems with the methods you describe.

(10)

Describe an approach of mutual exclusion that does not require busy waiting.

(10)

## Question 2

a) With regards to I/O design principles describe the layers of the I/O system and justify this structure.

(15)

b) Consider the following C-language I/O statement:

```
count = read(fd, buffer, nbytes);
```

where fd is an integer (called the file descriptor)
buffer is a memory address pointing into the processes data memory
nbytes is an integer indicating the number of bytes to be read.

Supposing that fd refers to a file on disk with a controller which uses DMA, describe how this statement may be handled in the layers of the I/O system?

(5)

c) How are devices represented in the UNIX operating system? How are the drivers specified? Could the file descriptor in question (b) be referring to a terminal? What would be the difference if this were so?

(5)

## Question 3

a) Describe the four generations of computing and how operating systems developed as a result.

(12)

b) There is some debate as to what will constitute a fifth generation computer. Assume such a computer is available. What do you think will differentiate it from the computers of today?
What advances do you think need to be made in order to produce a fifth generation computer?

(13)

## Question 4

(a) Given a disk with 200 tracks, where track requests are received in the following order

55, 58, 39, 18, 90, 160, 150, 38, 184.

The starting position for the arm is track 100. Calculate the number of tracks crossed when the following algorithms are used

- First Come First Serve
- Shortest Seek First
- The elevator algorithm starting in the direction UP.

(15)

b) Briefly explain, in which of the following cases can the algorithms in (a) augment its performance?

- A process is reading 10,000 blocks with consecutive disk addresses.
- A process is reading 10,000 blocks with random disk addresses.
- A process is creating child processes to read 10,000 blocks with random addresses.
- Processes are communicating with each other by writing and reading blocks to disk.

(8)

c) In the last case of question (b), could the algorithm influence the synchronisation between the processes?

(2)

## Question 5

Describe two file system implementations that use linked lists. Describe the advantages and disadvantages of each method.

(12)

Describe the I-node method of implementing a file system.

(8)

It has been suggested that the first part of each UNIX file be kept in the same disk block as its I-node. What, if any, would be the advantage of doing this?

(5)

## Question 6

a) Every file in a filing system has a set of attributes (read only, date created etc.). Assume a filing system allows an attribute of *temporary*, meaning the creating process only uses the file during the time it is executing and has no need for the data thereafter. Assume the process is written correctly, so that it deletes the file at the end of its execution. Do you see any reason for an operating system to have *temporary* file attribute? Give your reasons.

(5)

b) An operating system supplies system calls to allow you to COPY, DELETE and RENAME a file.
Discuss the differences between using COPY/DELETE and RENAME to give a file new name?

(5)

c) An operating system only allows a single directory hierarchy but allows arbitrary long filenames. Could you simulate something approximating a hierarchical file system? Give your reasons.

(5)

d) When a file is removed, the blocks it occupies are simply placed back onto the free list. Can you see any problems with this? If so, how would you overcome them and what problems, if any, would now exist and how would you resolve these?

(5)

e) When the UNIX filling system opens a file its i-node is copied into memory. It has been suggested, that with memory getting cheaper that if *n* processes open the same file then *n* copies of the I-node could be held in memory. Is this a good idea? Give your reasons.

(5)