COMP25111

<u>Two hours</u>

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Operating Systems

Date: Thursday 22nd January 2014?

Time: 14:00 – 16:00?

Please answer Question ONE and any TWO Questions
of your choice from the other THREE questions provided

For full marks your answers should be concise as well as accurate.
Marks will be awarded for reasoning and method as well as being correct

---

This is a CLOSED book examination

The use of electronic calculators is NOT permitted

[PTO]

1. **Compulsory**

a) What is the key difference between a *system call* and a call to an ordinary method or function. Briefly explain why this difference is important.

(2 marks)

A system call forces the CPU into system mode (a.k.a. privileged, supervisor or kernel mode) as the call is made; the activated OS code is thus protected. This does not happen for ordinary calls, which simply activate other user mode code. For security, some things must only be done by the OS (e.g. changing page table entries, dealing with interrupts, swapping processes, etc.) and hardware enforcement is necessary (otherwise nothing can stop the user from doing whatever they want). So system mode adds some functionality which the user cannot get at directly (only indirectly, by making a system call). Typical examples of additional functionality are: access to certain parts of memory and certain parts of hardware (peripheral registers, etc.), special instructions, and so on.

b) In Linux, how does a shell implement a *pipe* between commands?

(2 marks)

A pipe connects output from one process to input of another process. The Unix way of doing this is inherited by Linux; every input or output looks like a file. In order to implement a pipe, the OS needs to link the two corresponding files together, and direct the writes to the output from the first process to become the reads for the input for the second process. This is achieved by means of a FIFO queue managed by the OS.

c) What does the term *starvation* mean in process scheduling? How may it arise?

(2 marks)

At least one process gets continuously overlooked (starved of CPU access) by the scheduler. A typical scenario is where large numbers of brief, new incoming processes are systematically placed at the head of a single scheduling queue, thereby repeatedly deferring later entries. Other scenarios are possible.

d) Briefly explain the difference between a *process* and a *program*. What is the difference between a *process* and a *thread*?

(2 marks)

A process is a complete executing program, including its virtual memory space as defined by page tables, links to IO/files, and whatever else is needed. A program is the object code and constant initial data produced by the compiler/linker/loader; it does not contain any space for computed data, including any stack or heap. Nor does it contain 'in-flight' information such as file and device descriptors etc. A thread is an execution image living in (and sharing) some existing process' virtual memory space; the 'private' memory attached to this execution image is much smaller than the entire virtual memory space (effectively a stack and the CPU state).

e) What is the difference between a *monolithic* operating system and one constructed around a *microkernel*?

(2 marks)

A monolithic OS executes entirely in system (a.k.a. privileged, supervisor or kernel) mode, so it is large and often unforgiving, resulting in possible denial of service for what might be quite critical operations. The microkernel approach features large amounts of unprivileged OS execution (which can thus be interrupted by more critical operations), executing in system mode only when absolutely necessary.

f) Direct memory access (DMA) is interrupt driven. Given that a processor writes to a disk, utilizing DMA, describe the four-step DMA process for writing data.

(2 marks)

Bookwork (2 marks):

f) Direct memory access (DMA) is interrupt driven. Given that a processor writes to a disk, utilizing DMA, describe the four-step DMA process for writing data.

(2 marks)

The following points should be covered to some degree in the answer:

Answer should cover the 'basic points' and for higher marks put these basic points into context.

A processor writes to a disk; the process can be speeded up utilising direct memory access (DMA).

The 4-steps DMA process is:

1) Processor inform the disk DMA I/O - to write data by writing to a command register in the DMA I/O device;
2) DMA controller starts write process;
3) Process gets on with another work [process] until;
4) DMA controller interrupts to finish data transfer.

2 marks for an answer that describes and contextualises all four and gives the salient facts in a sensible way; and provides a well-defined set of brief descriptions of each;
1 mark for some basic understanding (or attempt).

g) One method for implementing virtual memory is *paged virtual memory*. The procedure for this can be viewed as a sequence of steps. Explain the paged virtual memory procedure by outlining the sequence of steps for translating a virtual address to a physical address.

(2 marks)

<u>Bookwork (2 marks)</u>:

g) There are two major methods for implementing virtual memory; one is paged virtual memory. Explain the paged virtual memory procedure.

(2 marks)

The following points should be covered to some degree in the answer:

1. The processor generates a logical address;
2. The page number field is used by the MMU to look to see whether the page is in memory or not;
3. If it is in memory, a physical address is computed by replacing the page number with the page frame number of where the page can be found
4. Together with the offset this is used as a physical address to memory; and
5. If it is not in memory, the transfer is aborted (page fault) and the operating system will load the page from disk to memory.

2 marks for an answer that depicts all the salient facts in a sensible way; and correctly delineated and briefly described;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lecture 11: Virtual Memory (1).

TOTAL marks (2 marks) [4]

h) Given that the purpose of a *page table* is to translate the page number (in the virtual memory) into a page frame (in the physical memory), and that the current partial view of the page table is:

[…]
[07, 00]
[06, xx]
[05, 03]
[04, 02]
[…]

where [X, Y] = [page number, page frame].

i) Calculate the page frames and page offsets given that the page numbers and page offsets to be sequentially translated are ‹07, 06› and ‹06, 01›, where ‹X,Y› = ‹page number, page offset›.

(1 mark)

ii) Then state which page translation, ‹07, 06› or ‹06, 01›, causes a *page fault*.

(1 mark)

Application (2 marks)
Example answer:- The following points should be covered to some degree in the answer:

i) Calculate the page frames and page offsets given the page numbers and page offsets are: [07, 06] and [06, 01]:

[page number, page offset] -> [page frame, page offset]

[07, 06] -> [00, 06]
[06, 01] -> [,] no page frame allocated to page number 06

(1 mark)

ii) Then state which page translation, [07, 06] or [06, 01], causes a *page fault*; also briefly state what happens: "if the page is not in memory":

Page number 06 causes a *page fault* as: [06, 01] -> [,] no page frame allocated to page number 06.
IF the page is not in memory THEN the transfer is aborted (page fault) and the operating system will load the page from disk to memory.

(1 mark)

2 marks for a totally correct part i & ii; and explicit concise explanation for second part of ii.
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 11; Memory Management (1).

TOTAL marks (2 marks) [6]

j) State how to avoid *external fragmentation* in memory. Your answer should include a brief description of the solution and a diagram supporting your description which explicitly covers the ‘before’ and ‘after’ scenarios.

(2 marks)

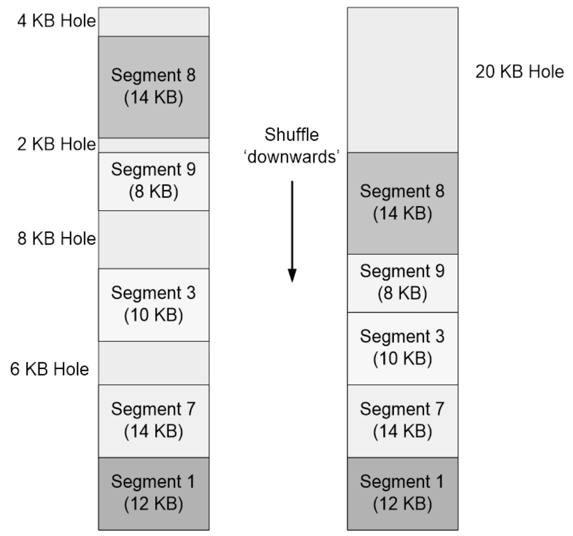
Bookwork and Diagrammatic depiction (2 marks)

j) State how to avoid external fragmentation. Your answer should include a brief description of the solution and a diagram supporting your description which explicitly covers the ‘before’ and ‘after’ scenarios. (2 marks)

Example answer:- The following points should be covered to some degree in the answer:

The process is termed shuffling.
Compact the memory by shuffling segments in memory to fill the holes.
External Fragmentation can occur, where memory space is wasted due to holes in the physical memory. Shuffling requires extensive copying of data and is excessively time consuming.

‘before’ and ‘after’



2 marks for a totally correct explicit delineation of the process using keywords in context – plus a complete – well drawn – and labelled diagram.
1 mark for some basic table (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 12; Virtual Memory (2) Segmented Virtual Memory.

TOTAL marks (2 marks) [8]

k) What does the *dirty* bit in a page table entry indicate? State how it is utilised.
(2 marks)

Bookwork (2 marks)

k) What does the *dirty* bit in a page table entry indicate? State how it is utilised.
(2 marks)

Example answer:- The following points should be covered to some degree in the answer:

The dirty bit indicates when the memory has been modified. The dirty bit is a status bit which marks the block that has been modified. Set if the page is written to (or dirty). The dirty is utilised when a page (or segment) is replaced – ejected or removed from physical memory – if it is set the page (or segment) must be written back to secondary memory (or hard disk) prior to its replacement. While if the dirty bit is clear (= "0") the location can be overwritten – without a write back cycle being performed.

2 marks for a totally correct answer using keywords in context and explicit description including good explanation of utilisation,
1 mark for some basic table (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 11 & 12, Virtual Memory (1 & 2), Paged (and segmented) Virtual Memory.

TOTAL marks (2 marks) [10]

2. a) What is the difference between *preemptive* and *non-preemptive* scheduling?
(2 marks)

Preemption is the ability for the scheduler to remove the currently running process (or thread) from the CPU and place it in the ready queue, then select a different ready process and run it. A non-preemptive scheduler does not have this ability and so any process it selects to run will only yield the CPU when it next blocks. If it does not block at all, it will run to completion. A long running, non-blocking process will thus be able to 'hog' the CPU. A preemptive scheduler is able to remove such a process from the CPU, but it needs a timer interrupt to allow it to intervene. The timed interval is known as the time slice (or quantum).

b) Why is the size of the time slice in preemptive scheduling algorithms chosen to be significantly higher than the time taken for a context switch?
(2 marks)

A context switch is the procedure for removing the current running process and replacing it by the newly selected process. The time this takes is essentially 'wasted' – no process benefits from it. Hence the number of context switches needs to be minimised, which in general can only be achieved by making them less frequent, subject to other necessary constraints. It is easy to see that a time slice of the same size as a context switch will lead to only 50% utilisation of the CPU for processes, so in practice we need a time slice that is one or two orders of magnitude larger (one order larger implies around 90% CPU utilisation – two orders implies around 99%).

c) Explain briefly what a CPU burst is and what an I/O burst is. What is a CPU-bound process, and what is an I/O-bound process? Why is it a good strategy in process scheduling to give higher priority to I/O-bound processes?
(4 marks)

A CPU burst is where a process uses the CPU continuously for some (relatively long) period of time. An I/O burst is where a process uses the CPU intermittently while it is mostly waiting for data transfer to occur between CPU and peripheral devices. A CPU-bound process is mostly executing in CPU bursts, while an I/O-bound process is mostly executing in I/O bursts. I/O-bound processes voluntarily yield the CPU more readily than CPU-bound processes, thereby ensuring that "someone else gets a go" more often. Progress towards the final goal of completing execution is thus quicker (and there are other benefits, such as reduced contention for resources).

d) Three processes A, B and C all alternate between fixed duration CPU bursts and I/O bursts, as follows. Process A has CPU bursts of 3 time units and I/O bursts of 4 units. Process B has CPU bursts and I/O bursts of 4 time units each. Process C has CPU bursts of 1 time unit and I/O bursts of 6 time units. Draw a diagram showing the states of these processes as they are run by a preemptive Round Robin scheduler for a total of 40 time-units, assuming that they all start ready at time-unit 0 and are queued in the order A (first), B, C (last), the time-slice adopted by the scheduler is 2 time-units, and the time for a context switch is negligible. For what fraction of the time is the CPU executing user processes?

(5 marks)

Diagram should look something like the following (=== is I/O burst, `--` is "ready but not running"):

```
AA---A====AA---A====AA--A====AA--A====AA
--BB--BB====-BB-BB====BB-BB====BB-BB====
----C======-C======C======-C======--C===
        **        *         *        *
```

CPU is idle at `*`s above, so utilisation for user processes is 35/40 = 7/8ths.

e) A new scheduler is introduced using priority queues. There are two queues, Q1 (responsible for scheduling processes A and B) and Q2 (responsible for scheduling process C). Assuming they have ready processes, the two queues access the CPU alternately, as follows: Q1 gets 5 time-units, then Q2 gets 2 time-units, then Q1 gets 5 time-units, and so on. Processes in each queue are executed in Round Robin fashion with a time-slice of 2 time-units. If a queue runs out of ready processes before its allocated time-units have been used, it yields access to the CPU to the other queue. Apart from this, the situation is as described for part d) above. Draw a diagram showing the states of the three processes A, B and C as they are run by the scheduler for a total of 40 time-units. For what fraction of the time is the CPU executing user processes?

(5 marks)

Diagram should now look something like the following:

```
AA--A====AAA====-AAA====-A-AA====AAA====
--BB--BB====BB-BB====BBBB====BBBB====BBB
-----C======--C======-----C======---C===
      + *   #  +    *#     +   + #   +
```

There are alternative possible choices at # above when two processes finish their I/O at the same time. Any consistent diagram will get the marks.

Notes: Q1 yields the CPU at the first * above because it has no “ready” process to run. Q2 yields the CPU at all + and * above for the same reason.

CPU is idle at `*`s above, so utilisation for user processes is 38/40 = 19/20ths.

f) Briefly explain how *dynamic adjustment* of the priority of each process can be used to improve the behaviour of a scheduler.

(2 marks)

Dynamic priority adjustment is an appropriate mechanism for implementing scheduling policies that vary the way CPU time is given to a process as its behaviour varies during its execution. For example, it could be used to change the scheduler’s actions when a process moves from being I/O bound to being CPU bound (and vice versa). Evidently some means of monitoring the process behaviour is needed in order to achieve this.

3. a) To address the question: “given a single user program, where does it fit in memory?”, state the steps an operating system must take in order to load a single user program into memory (this is termed *uniprogramming*).

(5 marks)

3.a

Application (5 marks),

a) To address the question: “given a single user program where does it fit in memory?” State the steps an operating system takes to load a single user program into memory; given this is uniprogramming. (5 marks)

Example answer:- The following points should be covered to some degree in the answer:

OS like KOMODO or original MSDOS undertake the following steps when a program is loaded into memory:

1) OS given a file name;
2) Loads it from disc;
3) Jumps to start of program;
4) Executes the program; may do I/O etc. via OS;
5) Returns to OS when done.

5 marks for a totally correct content – all five issues covered comprehensively,
3 marks for all issues covered – NOT comprehensively [though].
2 marks for half issues covered,
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 10; Memory Management (1).

TOTAL marks (5 marks) [7]

b) In the context of converting an address generated by a program [a compiler] to the actual address, state:

i) The names of the two memory spaces involved; and (1 mark)

ii) The unit that performs [undertakes] this translation process. (1 mark)

3.b
Bookwork (2 marks),

b) In the context of converting an address generated by a program [a compiler] to the actual address; state:

i) The names of the two memories involved; and (1 mark)
ii) The unit that performs [undertakes] this translation process. (1 mark)

Example answer:- The following points should be covered to some degree in the answer:

In the context of converting an address generated by a program [a compiler] to the actual address:
i) The names of the two memories involved are:

a. Virtual memory; and
b. Physical memory.

ii) The unit that performs [undertakes] this translation process is the:
This process is undertaken by the MMU (Memory management unit).

2 marks for an answer that mentions all the salient facts in a sensible way (2 marks for a clear, correct answer, 1 mark for a 'right lines' solution),
1 mark for correct answer but not detailed [enough].

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lecture 11: Memory Management (1).
TOTAL marks (2 marks) [2]

c) With respect to *multiprogramming*, and the diagram in Figure 3.c, state the following:

i) How many programs can be loaded into the partitions in Figure 3.c? (1 mark)

ii) What operation is performed when all partitions are full and no more programs need to be loaded? (1 mark)

iii) What does the operating system normally do when a program performs an I/O operation? (1 mark)

iv) What happens when one of the programs finishes? (1 mark)

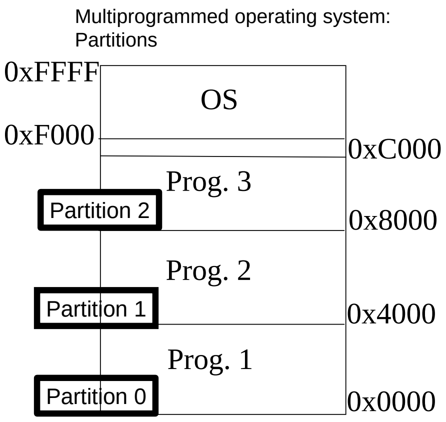


Figure 3.c. Typical diagram showing multiprogrammed operating system and associated partitions.

3.c.
Application of your knowledge - Bookwork (4 marks),

i) How many programs can be loaded into the partitions in figure 3.b?(1 mark)

Example answer:- The following points should be covered to some degree in the answer:

The OS can load multiple programs; in this case it can load a maximum of 3 programs to fill all three partitions.

ii) What operation does the operating system perform when all partitions are full and no more programs need to be loaded? (1 mark)

Example answer:- The following points should be covered to some degree in the answer:

When 3 programs are running in a multiprogramming [fixed Partition] system – the OSs function it to switch between them – or time division multiplex – or context switch at regular intervals.

iii) What does the operating system normally do when a program performs an I/O operation?(1 mark)

Example answer:- The following points should be covered to some degree in the answer:

When a program performs an I/O operation the OS will switch to another program – as I/O operations are slow.

iv) What happens when one of the programs finishes? (1 mark)

Example answer:- The following points should be covered to some degree in the answer:

When one of the programs finishes the OS brings in a new program; into the same partition that was vacated by the program that has just finished.

4 marks for a totally correct content, and all issues addressed comprehensively;
3 marks for three-quarters of the issues covered, and issues addressed comprehensively;
2 marks for half issues covered, and issues addressed comprehensively;
1 mark for some basic knowledge (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 10; Memory Management (1).

TOTAL marks (4 marks) [11]

d) Describe in detail how a *segment* is loaded; list the steps taken.

(5 marks)

3.d

Bookwork (5 marks),

d) Describe in detail how a ‘segment’ is loaded. (5 marks)

Example answer:- The following points should be covered to some degree in the answer:

Description of the segmentation process; or how a ‘segment’ is loaded; this works in almost exactly the same way as paged virtual memory:

1. The processor generates a logical address.
2. The segment number field is used by the MMU (memory management unit) to look to see whether the segment is in memory or not, then:
3. If it is in memory, a physical address is computed by adding the base address of the segment to the offset; this is used as a physical address to memory; or
4. If it is not in memory, the transfer is aborted (segment fault); and then the operating system will load the segment from disk to memory.

5 marks for a concise description of complete segmentation loading process,
2 ½ marks for half issues covered,
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 12; Virtual Memory (2) Segmented Virtual Memory.

TOTAL marks (5 marks) [16]

e) Given a physical address size of 2GB and associated 64KB block size, calculate the *number of page frames* in the physical address space. **NOTE**: To gain full marks you must show full working.

(2 marks)

3.e.
Application (2 marks),

e) Given a physical address size of 2G and associated 64K block size. Calculate the number of page frames in the physical address space. NOTE: To gain full marks you must show full working. (2 marks)

Example answer:- The following points should be covered to some degree in the answer:

Virtual address space 2GB and block size 64KB.
If the virtual address space is 2 GB and the page size is 64 KB there are:

$$\text{Number of Page frames} = \frac{\text{Address space}}{\text{Block size}}$$

$$\frac{2G}{64K} = \frac{2{,}147{,}483{,}648}{65{,}536} = \frac{2^{31}}{2^{16}} = 2^{31-16} = 2^{15} = 32{,}768 \approx 32k\, page\ frames$$

2 marks for an answer that calculates the correct answer and is laid out correctly e.g.
2 marks for a correct answer and full working out,
1 mark for a 'right lines' approach. Moderate marks will be awarded in the case of correct application for a wrongly calculated.,
½ **marks** for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 11; Virtual Memory (1).

TOTAL marks (2 marks) [18]

f) Given a 4GB address space and associated 16KB page size; calculate the *number of pages* that result in the virtual address space. **NOTE**: To gain full marks you must show full working.

(2 marks)

3.f.
<u>Application (2 marks),</u>

f) Given a 4G address spaces and associated 16K page sizes; calculate the <u>number of pages</u> that result in the virtual address space. NOTE: To gain full marks you must show full working. (2 marks)

Example answer:- The following points should be covered to some degree in the answer:

Given a block [page] size of 16 KB and a physical address space of 4 GB.
If the virtual address space is 4 GB and the block size is 16 KB there are:

$$\text{Number of Pages} = \frac{\text{Address space}}{\text{Page size}}$$

$$\frac{4G}{16K} = \frac{4{,}294{,}967{,}296}{16{,}384} = \frac{2^{32}}{2^{14}} = 2^{32-14} = 2^{18} = 262{,}144 \approx 262k\ page\ frames$$

2 marks for an answer that calculates the correct answer and is laid out correctly e.g.
2 marks for a correct answer and full working out,
1 mark for a 'right lines' approach. Moderate marks will be awarded in the case of correct application for a wrongly calculated.

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: 11; Virtual Memory (1).

TOTAL marks (2 marks) [20]

4. a) Draw up a table that lists:
1. Page replacement policy name; and
2. Brief description of how the policy works.

In the table, describe three policies: First In First Out; Least Recently Used; and Not Recently Used.

(5 marks)

4.a
<u>Bookwork (5 marks):</u>

a) Draw up a table that lists:
1) Page Replacement policy name; and
2) Brief description of how the policy works.
In the table describe three policies: First in First Out; Least Recently Used; and Not Recently Used. (5 mark)

The following points should be covered to some degree in the answer:

Policy name	**Description of how the policy works**
First in First Out	"Based on the theory that the best page to remove is the one that has been in memory the longest." The First in First Out policy identifies the oldest page in real memory and get rid of that; they can be tagged with a sequence number.
Least Recently Used	"Policy chooses the pages least recently accessed [referenced] to be swapped out." The least recently used works on temporal locality principle. It infers that if something has been used recently then it probably will be again - and the converse. The simplest implementation is to associate a timestamp counter (may need 64 bits) with every page, updated whenever the page is accessed.
Not Recently Used	"Replace the page which is ***not used recently***." Recently used pages kept in memory; this infers not recently used pages replaced. The R and M bits in a page table are used: The R bit set when referenced; and The M bit set when modified (written to).

5 marks for an answer that depicts all the salient facts in a sensible way; and the use of <u>keywords</u> in context; also describing all three in detail;
3 marks for an answer that depicts a proportion of the salient facts in a sensible way; and the use of some <u>keywords</u> in context;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lectures 13: Virtual Memory (3).
TOTAL marks (5 marks) [5]

b) Given that a keyboard sends characters into the computer system, name and describe the two registers normally used to undertake this process.

(5 marks)

4.b

Bookwork & Critique [Differentiate] (5 marks):

b) Given a keyboard sends characters into the computer system. Name and describe the two registers normally used to undertake this process. (5 marks)

The following points should be covered to some degree in the answer:

The name of the first is the Status register.
The Status register indicates the status of the I/O device.
In our simple example, assuming that only one bit in the status register is used (bit 0) and the other 7 bits have the value 0. Bit 0 has two states:
If bit 0 is set to '1', a character has been typed and its value is in the data register; and
If bit 0 is cleared to '0', no character has arrived since the last time the processor read from the data register.

The name of the second is the Data Register.
The Data Register holds the 'value' of the character typed at the keyboard.
Characters are encoded using numeric schemes, an 8-bit scheme is called ASCII, and each of the 256 possible ASCII characters has an 8-bit code.
ASCII is gradually being superseded by the 16-bit Unicode standard [1] – used in Java.
So the word "Hello" would be encoded as the sequence (decimal) 72-101-108-108-111.

5 marks for an answer that depicts all points, and the use of keywords in context;
3 marks for half of the facts;
1 mark for some basic understanding (or attempt).

Reference Learning Resources, Background Reading, and Lecture itself for detailed information; Lecture(s) No(s).: Lectures 14: Controlling Input and Output 1.
TOTAL marks (5 marks)[10]

c) Explain what a *semaphore* is and describe the operations that can be performed on it.

(3 marks)

Bookwork: A semaphore is a shared integer variable that can only be accessed by the special operations P (for procure) and V (for vacate). For semaphore S, P(S) will repeatedly try to find S in state S>0; if it ever succeeds, it will immediately and atomically reduce S by 1 then complete (thereby entering the critical region of code, which immediately follows the P(S) operation). When execution of the critical region is complete, a call to V(S) should be made. This simply increments the value in S (again atomically), thereby potentially making it possible for another thread's P(S) operation to see it in state S>0. For a semaphore initialised to value 1 (a binary semaphore), this will ensure mutually exclusive access to critical regions in multiple threads.

d) In a certain system, the execution of three threads is synchronised using three semaphores, S1, S2 and S3, as shown below. Semaphores S1 and S2 are initialised to zero, while semaphore S3 is initialised to 1. All three semaphores are used only in the sections of code shown below.

```
Thread A      Thread B      Thread C
…             …             …
P(S1)         P(S2)         P(S3)
P(S2)         P(S2)         V(S2)
x=3*x         x=x+7         x=x-1
V(S3)         V(S1)         V(S2)
…             …             V(S2)
…             …             …
```

i) If the variable x is defined as an integer shared variable, initialised to 0, and is not assigned a value in any other sections of the code apart from those shown above, what will be its value when all three threads have finished executing? What will be the values of the three semaphores S1, S2 and S3? Justify your answers.

(5 marks)

Threads A and B block until S1 and S2, respectively, are vacated. This can only be done by Thread C, which vacates S2 first and thereby potentially releases Thread B. Thread B now tries another procure for S2 which can again only be vacated by Thread C. This happens only after thread C has executed its update of x, so this is the first action (x=0-1=-1). Thread C now vacates S2 twice in succession, so up to two procures can now succeed for S2. One of these will release Thread B which can thus do its update of x. Nothing can happen in Thread A until Thread B vacates S1, and Thread C does not touch x again, so this is the second action (x=-1+7=6). Now Thread B vacates S1 and finishes executing its code. The vacate of S1 releases Thread A which now tries to procure S2. This is only possible once Thread C has vacated S2 for the third time and has therefore finished executing its code. Then and only then Thread A can do its update of x, so this is the third action (x=3*6=18). It is also the final update to x since no more are left, so the final value of x is 18. Thread A then vacates S3, so semaphore S3 will be vacated at the end (S3=1). The final vacates for Threads B and C have all been procured as explained above, so semaphores S1 and S2 will have values of zero at the end (S1=S2=0).

ii) Would the same final value for x be computed if the two P operations in Thread A were exchanged? Is there any other way in which the behaviour of the code might change? Justify your answers.

(2 marks)

The key action for release of Thread A to do its update to x is the procurement of semaphore S1. If the two procures at the start of Thread A are exchanged, Thread B still has to finish before Thread A can start its update of x, so the order of updates to x is preserved. The only potential difference in behaviour is the order in which Threads A and B procure S2 when Thread C vacates it three times. Thread A may progress from the first procure to the second earlier than described above.

END OF EXAMINATION