# CSC 205 — All-In-One Revision

*Built from your lecture notes plus the 2022, 2023 and 2024 past papers. Ordered by how often each topic actually appears.*

**What the papers actually test, in order of marks**

1. **Number base conversions** — every paper, usually 5+ marks
2. **Signed representations** (sign magnitude, 1's, 2's, 9's, 10's complement, excess-N) — every paper
3. **Bit-width formulas and 2's complement arithmetic** — every paper
4. **Computer ethics** — scope areas and theories
5. **Operating systems** — services and types
6. **Windows and GUI** — features, window parts, GUI terms
7. **History and generations** — a question or two

*Parts 1 to 3 are worth more than everything else combined and they are procedural, so you can learn them tonight. Parts 4 to 7 are pure recall, cheap to pick up in the morning.*

## PART 1 · NUMBER SYSTEMS AND CONVERSIONS

### 1.1 The four bases

| Base | Name | Symbols used | Suffix | Why it exists |
|---|---|---|---|---|
| 10 | Decimal | 0 to 9 | D | What humans count in |
| 2 | Binary | 0, 1 | B | Computers are built from switches with two states, on and off |
| 8 | Octal | 0 to 7 | O / Q | Short form for binary, 1 octal digit = 3 bits |
| 16 | Hexadecimal | 0 to 9, A to F | H | Short form for binary, 1 hex digit = 4 bits |

A binary digit is a **bit**. Eight bits is a **byte**. Hex and octal exist only because long binary strings are painful to read and easy to get wrong.

### 1.2 The table you must know cold

| Hex | Bin | Dec |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |

| Hex | Bin | Dec |
|---|---|---|
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| A | 1010 | 10 |
| B | 1011 | 11 |
| C | 1100 | 12 |
| D | 1101 | 13 |
| E | 1110 | 14 |
| F | 1111 | 15 |

**Powers of two.** Write this at the top of your answer sheet the moment the exam starts.

| 2^n | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

### 1.3 Hex to binary, and binary to hex

**Hex to binary:** replace each hex digit with its 4 bits. Nothing else.

```
A3C5H = 1010 0011 1100 0101 B
102AH = 0001 0000 0010 1010 B
```

**Binary to hex:** group into 4s **starting from the right**, pad the left with zeros, then replace each group.

```
1001001010B
  group from right:  10 0100 1010
  pad left:        0010 0100 1010
  convert:            2    4    A   =  24AH
```

**Octal works the same way but in groups of 3**, because 8 = 2^3.

```
10101101B  ->  010 101 101  ->  2 5 5  =  255 octal
```

### 1.4 Any base to decimal

Multiply each digit by its base raised to the position, counting from 0 at the right, then add.

```
A1C2H = 10x16^3 + 1x16^2 + 12x16^1 + 2x16^0
      = 40960  +  256   +  192     +  2    = 41410

10110B = 1x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0
       = 16 + 0 + 4 + 2 + 0 = 22
```

## 1.5 Decimal to any base

Divide repeatedly by the target base. Keep the remainders. Stop when the quotient hits 0. **Read the remainders bottom to top.**

```
261 to hexadecimal:
  261 / 16  ->  quotient 16, remainder 5
   16 / 16  ->  quotient  1, remainder 0
    1 / 16  ->  quotient  0, remainder 1   (stop)
  read upward:  1 0 5   ->  105H
```

```
120 to binary:
  120/2 -> 60 r 0
   60/2 -> 30 r 0
   30/2 -> 15 r 0
   15/2 ->  7 r 1
    7/2 ->  3 r 1
    3/2 ->  1 r 1
    1/2 ->  0 r 1   (stop)
  read upward:  1111000
```

## 1.6 One base straight to another

Same repeated division, you just do the arithmetic in the starting base. Easier route in an exam: go through decimal, or through binary if both bases are powers of 2.

```
1023 (base 4) to base 3:
  1023(4)/3 -> quotient 25D, remainder 0
     25D /3 -> quotient  8D, remainder 1
      8D /3 -> quotient  2D, remainder 2
      2D /3 -> quotient   0, remainder 2   (stop)
  read upward: 2210 (base 3)
```

## 1.7 Numbers with a fractional part

Split the number at the point and treat the two halves differently.

- **Whole part:** divide repeatedly, read remainders **upward** (as above)
- **Fraction part: multiply** repeatedly by the base, read the whole numbers **downward**, in the order they appear

```
18.6875 to binary

whole part 18:
  18/2 -> 9 r 0 ;  9/2 -> 4 r 1 ;  4/2 -> 2 r 0 ;  2/2 -> 1 r 0 ;  1/2 -> 0 r 1
  read upward:  10010

fraction .6875:
  .6875 x 2 = 1.375  ->  1
  .375  x 2 = 0.75   ->  0
  .75   x 2 = 1.5    ->  1
  .5    x 2 = 1.0    ->  1   (fraction is now 0, stop)
  read downward:  .1011

combine:  18.6875 = 10010.1011 B
```

```
18.6875 to hexadecimal
  whole 18:    18/16 -> 1 r 2 ;  1/16 -> 0 r 1   =>  12H
  fraction:    .6875 x 16 = 11.0  ->  11 = B
  combine:     12.BH
```

**Catch:** some fractions never terminate in binary, for example 0.456. Stop after the number of places the question asks for, or about 6, and say you rounded.

# PART 2 · SIGNED NUMBERS (not in your notes, in every paper)

## 2.1 The five representations

All five answer one question: how do you store a negative number when you only have 0s and 1s? **Fix the width first** — every answer depends on how many bits or digits you are using.

| Name | Rule | Example |
|---|---|---|
| Sign magnitude | Leftmost bit is the sign (0 = +, 1 = −), rest is the plain value | −5 in 4 bits = 1101 |
| 1's complement | Write the positive in binary, then flip every bit | 25 = 11001, flip = 00110 |
| 2's complement | 1's complement, then add 1. This is what real computers use | 00110 + 1 = 00111 |
| 9's complement | Decimal. Subtract each digit from 9 | 99 − 25 = 74 |
| 10's complement | Decimal. 9's complement plus 1 | 74 + 1 = 75 |
| Excess-N (biased) | Add N, store the result as a plain positive binary number. Subtract N to read back | Excess-8: −3 + 8 = 5 = 0101 |

**The 2022 paper asked all four for the number 25:** 9's = **74** | 10's = **75** | 1's = **00110** | 2's = **00111**

## 2.2 Reading a 2's complement number back

Look at the leftmost bit. If it is 0 the number is positive, read it normally. If it is 1 the number is negative, and you find its size by taking the 2's complement again.

```
1111 as 4-bit signed:
  leftmost = 1, so negative
  flip -> 0000, add 1 -> 0001 = 1
  so 1111 = -1,  magnitude 1

0111 as 4-bit signed:
  leftmost = 0, so positive
  0111 = 4+2+1 = 7,  magnitude 7
```

## 2.3 The three formulas

| Question | Formula | Worked |
|---|---|---|
| Bits needed to store N different things | Smallest n with $2^n \geq N$ | 356 instructions: 2^8 = 256 too small, 2^9 = 512 enough → **9 bits** |
| Largest value in n bits, unsigned | $2^n - 1$ (range 0 to that) | 5 bits: 2^5 − 1 = **31** |
| Range in n bits, signed 2's complement | $-2^{n-1}$ to $+2^{n-1} - 1$ | 4 bits: **−8 to +7** \| 8 bits: −128 to +127 |

*The signed range is not symmetric. There is always one more negative than positive.*

## 2.4 2's complement arithmetic

The point of 2's complement is that **subtraction becomes addition**. To do A − B, write B as a negative and add.

**1.** Fix the width **2.** Write both numbers in 2's complement at that width **3.** Add as ordinary binary **4.** Throw away any carry out of the leftmost column **5.** Read the answer back using 2.2

```
-2 - 6  in 4 bits
  -2:  +2 = 0010 -> flip 1101 -> +1 = 1110
  -6:  +6 = 0110 -> flip 1001 -> +1 = 1010

     1110
   + 1010
   -------
    11000     carry out of the left, drop it
     1000

  read 1000: negative, flip 0111, +1 = 1000 = 8
  answer: -8

5 - 6  in 4 bits
  +5 = 0101 ,  -6 = 1010

     0101
   + 1010
   -------
```

```
    1111

read 1111: negative, flip 0000, +1 = 0001 = 1
answer: -1
```

**Overflow.** If you add two positives and get a negative, or add two negatives and get a positive, the answer did not fit in the width. That is overflow. Say so if asked.

## 2.5 IEEE 754 floating point

| Precision | Total bits | Sign | Exponent | Mantissa | Bias |
|---|---|---|---|---|---|
| Single | 32 | 1 | 8 | 23 | 127 |
| Double | 64 | 1 | 11 | 52 | 1023 |

**Steps:** 1. sign bit  2. convert the number to binary including the fraction  3. normalise to $1.xxxx \times 2^{e}$  4. exponent field = e + bias  5. mantissa = the digits after the point, padded with zeros. The leading 1 is not stored.

```
Store -6.5 in single precision
  sign     = 1
  6.5      = 110.1 binary
  normalise= 1.101 x 2^2,  e = 2
  exponent = 2 + 127 = 129 = 10000001
  mantissa = 101 then zeros to 23 bits
  answer   = 1 10000001 10100000000000000000000
```

# PART 3 · COMPUTER ETHICS

## 3.1 Definition

Computer ethics is a branch of **applied ethics** dealing with the ethical principles, values and moral dilemmas arising from the use of computers and related technologies. It studies the ethical issues that arise during the design, development, implementation and use of computer systems, and aims to set guidelines so technology serves the best interests of individuals and society.

## 3.2 The seven areas in its scope — memorise this list

| Area | One line each |
|---|---|
| Privacy | Data privacy, surveillance, and protecting personal information in the digital age |
| Cybersecurity | Preventing cyberattacks and using security measures responsibly, including responsible disclosure of vulnerabilities |
| Intellectual property | Ethical use and protection of copyrights, patents and trademarks; piracy, plagiarism, fair use |
| Digital divide | The unequal access to technology and the internet between socio-economic groups |
| Artificial intelligence | Algorithmic bias and AI autonomy, especially in healthcare and criminal justice |
| Social media and online behaviour | Cyberbullying, harassment, and digital citizenship |
| Digital manipulation and misinformation | Deepfakes, spreading false information, and media literacy as the response |

*Memory hook:* **P C I D A S D**. *If a question asks you to "discuss the scope of computer ethics", name the areas and give one sentence each. That structure alone earns most of the marks.*

## 3.3 Historical development — the timeline

| Period | What happened |
|---|---|
| 1940s to 1950s | First electronic computers, ENIAC and UNIVAC. Ethical focus was military and defence use, responsibility and accountability in life-and-death contexts |
| 1960s to 1970s | Computers reach universities and research. Data privacy, intellectual property and responsible use of computing resources gain attention |
| 1970s to 1980s | Personal computers spread. Software piracy, unauthorised access and digital privacy become prominent |
| 1980s | Hacker culture emerges, debates on curiosity versus malice. **ACM** and **IEEE** establish professional codes of conduct |
| 1990s | The internet arrives. Censorship, digital citizenship, cyberbullying, global information flow |
| 21st century | Information warfare and AI. Algorithmic bias, impact on employment and autonomy. IoT spreads |

Two organisations to name: the **Association for Computing Machinery (ACM)** and the **Institute of Electrical and Electronics Engineers (IEEE)**, both of which wrote codes of conduct in the 1980s.

## 3.4 The nine ethical theories

| Theory | Core idea | Applied to computing |
|---|---|---|
| Utilitarianism | Consequentialist. Maximise overall happiness or well-being | Does deploying this AI benefit the greatest number of people? |
| Deontology | Non-consequentialist. Actions are right or wrong in themselves, whatever the outcome | You have a duty to protect personal data regardless of the benefits of collecting it |
| Virtue ethics | Cultivate good moral character traits | Develop honesty, integrity and responsibility in technology professionals |
| Rights-based ethics | People have fundamental rights that must be respected | Protects digital rights: privacy, freedom of expression, access to information |
| Social contract theory | Ethical rules come from an agreement among people to promote cooperation | Guides rules on responsible use of technology and data, and developers' obligations |
| Feminist ethics | Values of care, empathy and relationality | Consider all stakeholders, especially those marginalised or disproportionately affected |
| Ethical pluralism | No single theory answers everything | Use several ethical perspectives together on complex decisions |
| Ethic of prudence | Precaution and risk assessment | Weigh long-term consequences and risks of new technologies |
| Precautionary principle | Act to prevent harm even without scientific certainty | Applied to nanotechnology, biotechnology and AI where risks are not fully understood |

*The two most likely to be asked to contrast:* **utilitarianism** *(judge by consequences) versus* **deontology** *(judge by the action itself). Learn that pair properly.*

# PART 4 · OPERATING SYSTEMS

## 4.1 Definition

An **operating system** is an interface between the computer user and the computer hardware. It is software that performs all the basic tasks: file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. Examples: Linux, Windows.

The layered picture to draw if asked: **Users** at the top → **Software** (system software and application software) → **Operating System** → **Hardware** (CPU, RAM, I/O) at the bottom.

## 4.2 The seven OS services

| Service | What it does |
|---|---|
| Program execution | Handles activities from user programs to system programs. Each activity is a **process**: the complete execution context, meaning code to execute, data to manipulate, registers and OS resources in use |
| I/O operations | An I/O subsystem is the devices plus their driver software. Drivers hide the peculiarities of specific hardware from users. The OS manages communication between user and device drivers |
| File system manipulation | Grants permission to read or write files (read-only, read-write, denied), provides interfaces to create and delete files and directories, and to back up the file system |
| Communication | In distributed systems, manages communication between processes. Handles routing, connection strategies, contention and security. Done by **shared memory** or **message passing** |
| Error detection and handling | Errors can occur in the CPU, in I/O devices or in memory hardware. The OS constantly checks for errors and takes appropriate action to ensure correct and consistent computing |
| Resource management | In multi-user or multi-tasking environments, allocates main memory, CPU cycles and file storage. Uses **schedulers** and CPU scheduling algorithms |
| Protection | Controls access of programs, processes and users to system resources. Protects external I/O devices from invalid access and provides authentication by passwords |

## 4.3 Types of operating system

### a) Batch processing

A technique in which the OS collects the programs and data together in a batch before processing starts. A **job** is a predefined sequence of commands, programs and data treated as a single unit. The OS keeps a number of jobs in memory and executes them without manual intervention, in first come first served order. When a job finishes its memory is released and its output is copied to an output spool for later printing.

| Advantages | Disadvantages |
|---|---|
| Takes much of the work of the operator onto the computer | Difficult to debug a program |
| Increased performance, since a new job starts as soon as the previous one finishes, with no manual intervention | A job could enter an infinite loop |
| | Due to lack of a protection scheme, one batch job can affect pending jobs |

### b) Multitasking / time-shared

Multitasking is when multiple jobs are executed by the CPU simultaneously by **switching** between them. Switches happen so frequently that users can interact with each program while it is running. The user gives instructions directly and receives an immediate response. A time-shared OS uses CPU scheduling and multiprogramming to give each user a small portion of a time-shared CPU, and each user has at least one separate program in memory. These were developed to provide interactive use of a computer system at a reasonable cost.

*Multitasking operating systems are also known as* **time-sharing systems**. *That equivalence is a likely one-mark question.*

# PART 5 · WINDOWS AND GRAPHICAL USER INTERFACES

## 5.1 Windows and GUI

Windows is an operating system developed and marketed by Microsoft. It is a **Graphical User Interface (GUI)** operating system, meaning a platform where items are represented by an icon (a picture) on screen, which is easier to remember than the lengthy commands you needed in DOS.

A **graphical user interface** is a system that allows creation and manipulation of user interfaces employing windows, menus, icons, dialog boxes, mouse and keyboard. Examples: Macintosh toolbox, Microsoft Windows, X-windows.

Four tasks Windows performs: input and output management, resource management, data management, process management, plus the user interface.

## 5.2 Windows features

- **Graphical User Interface** — items represented by pictures
- **User friendly** — easier to learn and use
- **Multitasking** — run more than one application at a time, e.g. type a letter while printing another
- **Multithreading** — transfer data between applications, e.g. insert a cartoon into a letter
- **Common menus** — the same commands for printing across all software
- **Runs in protected mode** — the failure of one programme does not halt the computer
- **Faster processing** — 32-bit processing
- **Support for LAN, multimedia and internet**
- **On-line help facility**
- **Windowing capability**

## 5.3 The Windows screen

Four parts: the **Desktop**, the **Desktop Icons**, the **Task Bar**, the **Start Button**, and the **System Tray**.

**Taskbar** — the long horizontal bar at the bottom of the desktop. Three elements: **Start Button** on the extreme left (access to all applications), **System Tray** on the extreme right (status and time indicator), and **Task status** in the middle (buttons for all open windows).

**Start menu options:** Programs (start an application), Documents (the last 15 documents opened), Settings (adjust the system through control panels), Find (locate files and folders), Help, Run (a command line to type a path and program), Shut Down.

**Desktop icons:** My Computer, Recycle Bin, Internet Explorer, Network Neighborhood.

## 5.4 Components of a window — likely to be asked

| Part | Function |
|---|---|
| Title bar | Displays the window's title and indicates by its colour when the window has focus. Also how you move the window, by dragging it |
| Frame | Surrounds the window. Resize by pointing at a frame edge or corner |
| Control menu | Contains commands for manipulating the window. Opened by clicking the icon in the upper-left corner |
| Menu bar | Contains the titles of pull-down menus, for example File, Edit and Help |
| Work space | The portion inside the frame and under the title bar and menu bar. Also called the **client area** |
| Minimize and Maximize buttons | A window is in one of three states: **maximized** (fills the desktop), **minimized** (a button on the taskbar), or **restored** (occupies a section of the desktop) |
| Restore button | When maximized, the Maximize button is replaced by Restore. When minimized, the Minimize button is replaced by Restore |
| Close button | Closes the window. An alternative to the Close command on the control menu |
| Help button | In the button group in the upper-right corner, shown with a question mark icon |
| Scroll bars | When data extends beyond the area the window covers, horizontal and vertical scroll bars let you view the hidden data |

## 5.5 Common GUI terms

| Term | Definition |
|---|---|
| Pointing devices | Allow users to point at parts of the screen, invoke a command from a menu, and manipulate objects by selecting, moving or merging them. Since the 1960s: light pen, joystick, touch sensitive screen, and the mouse, which won out because of hand coordination and easier cursor tracking |

| Term | Definition |
|---|---|
| Bit-mapped displays | Made up of tiny dots (**pixels**) that are independently addressable, giving much finer resolution than character displays. Main advantage is graphic manipulation for vector and raster graphics, presenting information in its final form, also called **WYSIWYG** — What You See Is What You Get |
| Windows | When a screen is split into several independent regions, each one is a window. Several applications can display results simultaneously. Can be **tiled** or **overlapped** |
| Menus | Display a list of commands available within an application, so the user need not remember commands. Each item is a word or an icon |
| Dialog boxes | Collect information from the user or present information to them, for example number of copies when printing. Also used for error messages, as alert boxes |
| Icons | A symbolic representation of a system or user-defined object such as a file, folder, address, book or application |
| Desktop metaphor | Users are not aware of applications. They deal with files, folders, drawers, a clipboard and an out-box. Opening a report document implicitly invokes the word processor |

## 5.6 The four components of a GUI development environment

1. A **windowing system** — allows programs to display multiple applications at once, with tools for building movable and resizable windows, menus and dialog boxes
2. An **imaging model** — defines how fonts and graphics are created on screen, for example typeface and size in a word processor, curves and lines in a drawing program
3. An **application program interface (API)** — functions that let the programmer specify how the application controls menus, scroll bars and icons
4. A set of **tools and frameworks** for creating interfaces, mostly object-oriented

Three dominant camps of GUIs: IBM's **System Application Architecture (SAA)** which includes Microsoft Windows and PM (Presentation Manager); **UNIX** systems built around X Window; and the **Macintosh**.

## 5.7 Windows version timeline

| 1.0 | 2.0 | 3.0 | 95 | 98 | ME | XP | Vista | 7 | 8.0 | 8.1 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nov 1985 | Dec 1987 | May 1990 | Aug 1995 | Jun 1998 | Sep 2000 | Oct 2001 | Nov 2006 | Jul 2009 | Oct 2012 | Oct 2013 | Jul 2015 | Oct 2021 |

Microsoft was named “Traf-O-Data” in 1972, renamed “Micro-soft” in November 1975, then “Microsoft” on 26 November 1976. It entered the marketplace in August 1981 with MS-DOS, a 16-bit command-line operating system. Founded by Bill Gates and Paul Allen.

# PART 6 · FOUNDATIONS AND HISTORY

## 6.1 Computer science and its neighbours

| Discipline | Definition |
|---|---|
| Computer Science (CS) | The study of algorithms, computational systems and information. Both a theoretical science and a practical engineering discipline, concerned with what can be computed, how to compute it efficiently, and how to design and build systems to do so |
| Information Technology (IT) | Focuses on the implementation, support and management of computer systems, networks and hardware so they function effectively for users |
| Software Engineering (SE) | A subfield of computer science that applies engineering principles to the design, development, testing and maintenance of software systems |

## 6.2 Key figures

| Person | Contribution |
|---|---|
| Charles Babbage | Designed the Difference Engine (1820s) and the Analytical Engine (1840s), the first general-purpose mechanical computer |
| Ada Lovelace | Wrote the first algorithm intended for Babbage's Analytical Engine |
| Herman Hollerith | Developed punch-card systems for the 1890 US Census, leading to the company that became IBM |
| Alan Turing | 1930s and 40s theoretical foundations, the Turing machine |
| John von Neumann | The von Neumann architecture |

## 6.3 Generations of computers

| Generation | Years | Key technology | Key developments |
|---|---|---|---|
| Mechanical era | — | Mechanical calculators and analytical engines | Babbage, Lovelace, Hollerith's punch cards |
| First | 1940–1956 | **Vacuum tubes** for circuitry, magnetic drums for memory | ENIAC (1945), the first general-purpose electronic computer. UNIVAC I (1951), the first commercial computer. Programs written in machine language. Large, slow, unreliable |
| Second | 1956–1963 | **Transistors** replaced vacuum tubes | Smaller, faster, cheaper, more energy-efficient. High-level languages FORTRAN and COBOL. Early operating systems |
| Third | 1964–1971 | **Integrated circuits (ICs)** | Further miniaturisation, more power. Operating systems and time-sharing become widespread. CDC 6600, an early supercomputer using functional parallelism |
| Fourth | 1971–present | **Microprocessors**, an entire CPU on a single chip | Intel 4004 (1971), the first single-chip microprocessor. Personal computers (Apple II, IBM PC), networking, the internet, GUIs, and the mobile and AI revolutions |

*Fast recall chain:* **tubes, transistors, ICs, microprocessors**. *If you remember only the technology per generation you can reconstruct the rest.*

# PART 7 · MORNING DRILL

*Cover the answers. Write each one out before you look. If you can do all of these, you are ready.*

## Conversions

| Question | Answer |
|---|---|
| $10100101_2$ to octal | 245 |
| $10100101_2$ to hexadecimal | A5 |
| $120_{10}$ to binary | 1111000 |
| $745_8$ to binary | 111100101 |
| $110110.11_2$ to decimal | 54.75 |
| $3AB_{16}$ to binary | 1110101011 |
| $261_{10}$ to hexadecimal | 105H |
| $18.6875_{10}$ to binary | 10010.1011 |
| $19.25_{10}$ to binary | 10011.01 |

## Signed numbers and formulas

| Question | Answer |
|---|---|
| 9's complement of 25 | 74 |
| 10's complement of 25 | 75 |
| 1's complement of 25 (5 bits) | 00110 |
| 2's complement of 25 (5 bits) | 00111 |
| Bits needed to encode 356 instructions | 9 bits |
| Largest unsigned value in 5 bits | 31 |
| Range of 4-bit signed 2's complement | −8 to +7 |
| 0111 as signed 2's complement, magnitude | +7, magnitude 7 |
| 1111 as signed 2's complement, magnitude | −1, magnitude 1 |
| −2 − 6 in 4-bit 2's complement | 1000, that is −8 |
| 5 − 6 in 4-bit 2's complement | 1111, that is −1 |
| Single precision IEEE 754 field sizes | 1 sign, 8 exponent, 23 mantissa, bias 127 |

## Recall

| Question | Answer |
|---|---|
| Seven areas in the scope of computer ethics | Privacy, cybersecurity, intellectual property, digital divide, AI, social media and online behaviour, digital manipulation and misinformation |
| Two bodies that wrote codes of conduct in the 1980s | ACM and IEEE |
| Utilitarianism versus deontology | Judge by consequences versus judge by the action itself |
| Seven OS services | Program execution, I/O operations, file system manipulation, communication, error detection, resource management, protection |
| Two ways processes communicate | Shared memory or message passing |
| Another name for a multitasking OS | Time-sharing system |
| Three elements of the taskbar | Start button (left), system tray (right), task status (middle) |
| What WYSIWYG stands for | What You See Is What You Get |
| Technology of each generation | Vacuum tubes, transistors, integrated circuits, microprocessors |
| First general-purpose electronic computer | ENIAC, 1945 |
| First commercial computer | UNIVAC I, 1951 |
| First single-chip microprocessor | Intel 4004, 1971 |

**In the hall**

1. First thing, write the powers of two and the hex table in the margin. Thirty seconds, saves you ten minutes.
2. Always fix the bit width before answering any complement question.
3. On a paper that deducts for wrong answers, a blank beats a guess.
4. Show your division and multiplication steps. Method earns marks even when the final digit is wrong.