

scholareye.xyz

CLASS 9 • COMPUTER SCIENCE & ENTREPRENEURSHIP

CHAPTER 2

Number Systems

Complete Exercise Solutions

10

MCQs

10

Short Questions

7

Long Questions

www.scholareye.xyz

See Clearly. Learn Deeply. Succeed Confidently.

A**Multiple-Choice Questions**

10 MCQs • correct option ticked & explained

Q1**What does ASCII stand for?****(a) American Standard Code for Information Interchange****b**

(b) Advanced Standard Code for Information Interchange

c

(c) American Standard Communication for Information Interchange

d

(d) Advanced Standard Communication for Information Interchange

Explanation**Correct: (a)**

ASCII is the standard character encoding used in computers to represent text. It stands for **American Standard Code for Information Interchange**. It assigns a unique numeric value to letters, digits, and punctuation marks.

Q2**Which of the following numbers is a valid binary number?****a**

(a) 1101102

**(b) 11011****c**

(c) 110.11

d

(d) 1101A

Explanation**Correct: (b)**

Binary numbers consist of only two digits: 0 and 1. Option (a) contains the digit '2', option (c) contains a decimal point (a fractional representation, not a standard binary integer), and option (d) contains the letter 'A' (a hexadecimal digit). Therefore, **11011** is the only valid binary integer.

Q3 How many bits are used in the standard ASCII encoding?

☒ (a) 7 bits

☐ (b) 8 bits

☐ (c) 16 bits

☐ (d) 32 bits

Explanation

Correct: (a)

Standard ASCII uses **7 bits** to represent 128 characters ($2^7 = 128$). Extended ASCII uses 8 bits (allowing 256 characters), but the standard ASCII scheme is fundamentally 7-bit.

Q4 Which of the following is a key advantage of Unicode over ASCII?

☐ (a) It uses fewer bits per character

☒ (b) It can represent characters from many different languages

☐ (c) It is backward compatible with binary

☐ (d) It is specific to the English language

Explanation

Correct: (b)

Standard ASCII is limited to 128 characters, mostly catering to the English language. Unicode was developed to solve this limitation by providing a unique code point for every character in every language worldwide, including symbols, emojis, and historical scripts.

Q5 How many bytes are used to store a typical integer?

a (a) 1 byte

b (b) 2 bytes

☒ **(c) 4 bytes**

d (d) 8 bytes

Explanation

Correct: (c)

In modern computer architectures (32-bit and 64-bit systems), a standard integer (`int` in languages like C++ or Java) typically occupies **4 bytes** (32 bits) of memory.

Q6 What is the primary difference between signed and unsigned integers?

☒ **(a) Unsigned integers cannot be negative**

b (b) Signed integers have a larger range

c (c) Unsigned integers are stored in floating-point format

d (d) Signed integers are only used for positive numbers

Explanation

Correct: (a)

A signed integer reserves one bit (the most significant bit) to indicate whether the number is positive or negative. An unsigned integer uses all its bits for the magnitude, so it can represent only zero and positive numbers.

Q7 In the single precision, how many bits are used for the exponent?

a (a) 23 bits

☒ **(b) 8 bits**

c (c) 11 bits

d (d) 52 bits

Explanation

Correct: (b)

The IEEE 754 single-precision format uses 32 bits in total, divided as: **1 bit** for the sign, **8 bits** for the exponent, and **23 bits** for the mantissa (fraction/significand).

Q8 What is the approximate range of values for single-precision floating-point numbers?

☒ **(a) 1.4×10^{-45} to 3.4×10^{38}**

b (b) 1.4×10^{-38} to 3.4×10^{45}

c (c) 4.9×10^{-324} to 1.8×10^{308}

d (d) 4.9×10^{-308} to 1.8×10^{324}

Explanation

Correct: (a)

Because of the 8-bit exponent in single precision, the representable magnitudes range from about 1.4×10^{-45} (very close to zero) up to 3.4×10^{38} . Option (c) gives the approximate range for double-precision numbers.

Q9 What are the tiny dots that make up an image called?

☒ (a) Pixels

☐ (b) Bits

☐ (c) Bytes

☐ (d) Nodes

Explanation

Correct: (a)

“Pixel” is short for **picture element**. A digital image is a grid made of thousands or millions of these tiny dots, each holding colour information.

Q10 In an RGB color model, what does RGB stand for?

☒ (a) Red, Green, Blue

☐ (b) Red, Gray, Black

☐ (c) Right, Green, Blue

☐ (d) Red, Green, Brown

Explanation

Correct: (a)

The RGB colour model is an additive model in which **Red, Green and Blue** light are combined in various intensities to reproduce a wide spectrum of colours on electronic displays.

B

Short Questions

10 questions • concise model answers

Q1

What is the primary purpose of the ASCII encoding scheme?

ANSWER

The primary purpose of ASCII (American Standard Code for Information Interchange) is to provide a **standard numerical representation for text characters**. This allows different computer systems and software to exchange textual data reliably by agreeing on a specific number (code) for each letter, digit, and punctuation mark.

Q2

Explain the difference between ASCII and Unicode.

ANSWER

ASCII uses 7 or 8 bits to represent 128 or 256 characters, primarily covering the English language, digits, and basic punctuation. Unicode, on the other hand, is a much larger character encoding standard that uses variable-length encoding (8, 16, or 32 bits) to assign a unique code point to over 140,000 characters from virtually every language, symbol set, and emoji in the world.

Q3 How does Unicode handle characters from different languages?**ANSWER**

Unicode assigns a unique hexadecimal number (called a **code point**) to every single character, regardless of the platform, program, or language. For example, the English letter 'A' is `U+0041`, the Hindi letter 'अ' is `U+0905`, and the Chinese character '中' is `U+4E2D`. Software then uses encoding formats like UTF-8 or UTF-16 to store these code points in binary.

Q4 What is the range of values for an unsigned 2-byte integer?**ANSWER**

An unsigned 2-byte integer uses 16 bits ($2 \text{ bytes} \times 8 \text{ bits/byte}$). Since there is no sign bit, all 16 bits are used for the magnitude. The range is 0 to $2^{16} - 1$. Therefore, the range is **0 to 65,535**.

Q5 Explain how a negative integer is represented in binary.**ANSWER**

In modern computing, negative integers are represented using the **Two's Complement** method. To find the two's complement of a number, first write the positive binary equivalent, invert all the bits (0s become 1s, 1s become 0s), and then add 1 to the result.

Example: +5 in 8-bit is `00000101`; inverted it is `11111010`; adding 1 gives `11111011`, which is -5.

Q6 What is the benefit of using unsigned integers?**ANSWER**

The main benefit of unsigned integers is that they use the **full range of bits to represent positive values**. This effectively doubles the maximum positive value that can be stored compared to a signed integer of the same size, and it avoids the complexity of handling a sign bit for quantities that inherently cannot be negative (like memory addresses or object counts).

Q7 How does the number of bits affect the range of integer values?**ANSWER**

The number of bits directly determines how many different values can be represented: with n bits there are 2^n possible values. For example, an 8-bit unsigned integer can represent $2^8 = 256$ values (0-255), while a 16-bit unsigned integer can represent $2^{16} = 65,536$ values (0-65,535). More bits mean a larger range.

Q8 Why are whole numbers commonly used in computing for quantities that cannot be negative?**ANSWER**

Whole numbers (unsigned integers) are used for quantities like counting items, indexing arrays, or representing memory addresses because these quantities logically cannot be negative. Using unsigned integers for these tasks is more efficient, as it uses all available bits for the positive range and prevents logical errors that could arise from negative counts.

Q9

How is the range of floating-point numbers calculated for single precision?

ANSWER

For single precision (32-bit), the range is determined by the **8-bit exponent** field, which uses a biased representation (bias = 127). Normalised numbers use exponents from -126 to $+127$, so the largest value is about $2^{128} \approx 3.4 \times 10^{38}$. The smallest normalised value is $2^{-126} \approx 1.2 \times 10^{-38}$, and denormalised numbers reach down to $2^{-149} \approx 1.4 \times 10^{-45}$. Hence the approximate range is 1.4×10^{-45} to 3.4×10^{38} .

Q10

Why is it important to understand the limitations of floating-point representation in scientific computing?

ANSWER

Floating-point numbers cannot represent all real numbers exactly because of finite precision (they rely on binary fractions). This leads to tiny rounding errors. In scientific computing, these small errors can compound over millions of iterations, potentially leading to drastically incorrect results (e.g., in weather forecasting, orbital mechanics, or nuclear simulations). Understanding this limitation allows scientists to choose appropriate data types (like double precision) and error-mitigation algorithms.

C

Long Questions

Detailed, step-by-step solutions

Q1

Explain how characters are encoded using Unicode. Provide examples of characters from different languages and their corresponding Unicode code points.

ANSWER

Unicode encoding works by maintaining a standardized database called the **Unicode Standard**. Every character is assigned a unique hexadecimal number called a **code point**, written as `U+XXXX`.

- **Process:** The computer maps the character to its code point, then uses an encoding format (UTF-8, UTF-16 or UTF-32) to translate that code point into binary for storage and transmission.

Character	Script	Code point
A	English letter	U+0041
ﺃ	Arabic letter	U+0628
क	Hindi letter	U+0915
😊	Emoji	U+1F60A

This universal mapping ensures that no matter which operating system or programming language is used, a character always resolves to the correct symbol.

Q2 Describe in detail how integers are stored in computer memory.

ANSWER

Integers are stored as **fixed-length binary sequences** (usually 8, 16, 32, or 64 bits). The method depends on whether the integer is signed or unsigned:

Unsigned integers

All bits represent the magnitude. Example: the 8-bit unsigned integer `00000101` equals 5.

Q2 (continued)

Signed integers (Two's Complement)

The Most Significant Bit (MSB) is the sign (0 = positive, 1 = negative). To store -5 in 8 bits: write 5 as `00000101`, invert the bits to get `11111010`, then add 1 to get `11111011`. This lets the CPU perform addition and subtraction with the same ALU circuitry.

Q3 Explain the process of converting a decimal integer to its binary representation and vice versa. Include examples of both positive and negative integers.

Decimal to Binary (positive)

Use **repeated division by 2**: divide by 2, record the remainder (0 or 1), repeat with the quotient until it is 0, then read the remainders from bottom to top.

$$\begin{array}{l} 10 \div 2 = 5 \quad \text{R } 0 \\ 5 \div 2 = 2 \quad \text{R } 1 \\ 2 \div 2 = 1 \quad \text{R } 0 \\ 1 \div 2 = 0 \quad \text{R } 1 \end{array}$$

✓ Read upward: $10 = 1010$

Q3 (continued)

Binary to Decimal (positive)

Multiply each bit by 2 raised to its positional power (starting from 0 on the right) and add the results.

$$\begin{aligned} 1010 &= (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) \\ &= 8 + 0 + 2 + 0 \end{aligned}$$

✓ $1010 = 10$

Q3 (continued)**Negative integers**

Convert the positive number to binary, then take the Two's Complement (invert the bits and add 1).

```
+10 = 00001010
invert = 11110101
+1     = 11110110
```

✓ **-10 (8-bit) = 11110110**

Q4 Perform the following binary arithmetic operations:**(a) Multiplication of 101 by 11**

```
   101
×   11
-----
   101   (101 × 1)
  101    (101 × 1, shifted left by 1)
-----
  1111
```

✓ **Result: 1111 (which is 15 in decimal)**

Q4 (continued)**(b) Division of 1100 by 10**

$$\begin{array}{r} 110 \quad (\text{Quotient}) \\ 10 \overline{) 1100} \\ \underline{10} \\ 010 \\ \underline{10} \\ 00 \end{array}$$

✓ Result: 110 (which is 6 in decimal)

Q6 Add the following binary numbers:**(a) 101 + 110**

$$\begin{array}{r} 101 \\ + 110 \\ \hline 1011 \end{array}$$

Step by step: $1+0 = 1$; $0+1 = 1$; $1+1 = 0$ carry 1, and the carry is written as the leftmost digit.

✓ Result: 1011

Q6 (continued)

(b) $1100 + 1011$

```

  1100
+ 1011
-----
 10111

```

Step by step: $0+1 = 1$; $0+1 = 1$; $1+0 = 1$; $1+1 = 0$ carry 1, and the carry is written as the leftmost digit.

✓ Result: 10111

Q7 Convert the following numbers to 4-bit binary and add them:

(a) $7 + (-4)$

- 7 in 4-bit binary: 0111
- -4: 4 is 0100 → invert 1011 → add 1 → 1100

```

  0111
+ 1100
-----
 10011

```

Restricted to 4 bits, the overflow (leftmost 1) is discarded, leaving 0011.

✓ Result: $0011 = 3 \rightarrow 7 + (-4) = 3$

Q7 (continued)**(b) $-5 + 3$**

- -5 : 5 is $0101 \rightarrow$ invert $1010 \rightarrow$ add 1 $\rightarrow 1011$
- 3 in 4-bit binary: 0011

$$\begin{array}{r} 1011 \\ + 0011 \\ \hline 1110 \end{array}$$

The MSB is 1, so the result is negative. Two's complement of 1110 :
invert $\rightarrow 0001$, add 1 $\rightarrow 0010 = 2$.

✓ **Result: $1110 = -2 \rightarrow -5 + 3 = -2$**

Q8 Solve the following:**(a) $1101_2 - 0100_2$**

Decimal check: $13 - 4 = 9$.

$$1101 - 0100 = 1001$$

✓ **Result: 1001_2**

Q8 (continued)**(b) $1010_2 - 0011_2$**

Decimal check: $10 - 3 = 7$.

$$1010 - 0011 = 0111$$

✓ **Result: 0111_2**

Q8 (continued)**(c) $1000_2 - 0110_2$**

Decimal check: $8 - 6 = 2$.

$$1000 - 0110 = 0010$$

✓ **Result: 0010_2**

Q8 (continued)**(d) $1110_2 - 100_2$**

Align the bits first: 100_2 is treated as 0100_2 . Decimal check: $14 - 4 = 10$.

$$1110 - 0100 = 1010$$

✓ **Result: 1010_2**

scholareye.xyz



Chapter 2 Complete!

You have worked through every exercise of **Number Systems**.
Revise the solutions, practise the binary arithmetic again, and
keep exploring.

Binary & Two's Complement

ASCII & Unicode

Floating Point

Pixels & RGB

Keep Learning, Keep Growing

www.scholareye.xyz

More solved exercises, MCQs & CSS preparation material