Class : 9th

Subject : Computer Science

Chapter : 2 Number Systems

Book Exercise Complete Solved

Punjab Board

Multiple Choice Questions (MCQs)

---

1. What does ASCII stand for?
(a) American Standard Code for Information Interchange
(b) Advanced Standard Code for Information Interchange
(c) American Standard Communication for Information Interchange
(d) Advanced Standard Communication for Information Interchange
✅ **Correct Answer: (a) American Standard Code for Information Interchange**

Explanation:
ASCII is a standardized system for representing characters as numeric codes in computers. It assigns a unique 7-bit binary number to each character (e.g., letters, numbers, symbols) using the **American Standard Code for Information Interchange**.

---

2. Which of the following numbers is a valid binary number?
(a) 1101102
(b) 11011
(c) 110.11
(d) 1101A
✅ **Correct Answer: (b) 11011**

Explanation:
A valid binary number contains **only 0 and 1**. Option (b) `11011` is correct because it contains only binary digits.

- (a) has `2` (invalid in binary),
- (c) contains a decimal point,
- (d) has `A` which is not a binary digit.

---

3. How many bits are used in the standard ASCII encoding?
(a) 7 bits
(b) 8 bits
(c) 16 bits
(d) 32 bits
✅ **Correct Answer: (a) 7 bits**

Explanation:
The original/standard ASCII uses **7 bits**, which allows for **128 unique characters** (0 to 127). Though often stored in 8 bits (1 byte), only 7 bits are actively used in standard ASCII.

---

4. 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
✅ **Correct Answer: (b) It can represent characters from many different languages**

Explanation:
Unicode was developed to overcome the limitation of ASCII. It supports **multiple languages** including symbols, scripts, and emojis. ASCII only supports **English characters**.

---

5. How many bytes are used to store a typical integer?
(a) 1 byte
(b) 2 bytes
(c) 4 bytes
(d) 8 bytes
✅ **Correct Answer: (c) 4 bytes**

Explanation:
A typical integer in modern programming and systems is stored using **4 bytes (32 bits)**, which allows representing a wide range of whole numbers.

6. What is the primary difference between signed and unsigned integers?
(a) Unsigned integers cannot be negative
(b) Signed integers have a larger range
(c) Unsigned integers are stored in floating-point format
(d) Signed integers are only used for positive numbers
✅ **Correct Answer: (a) Unsigned integers cannot be negative**

Explanation:

- **Unsigned integers** only store **positive numbers (including zero)**.
- **Signed integers** store both **positive and negative** values using one bit to indicate the sign.

---

7. In single precision, how many bits are used for the exponent?
(a) 23 bits
(b) 8 bits
(c) 11 bits
(d) 52 bits
✅ **Correct Answer: (b) 8 bits**

Explanation:
In **IEEE 754 single precision format** (32 bits total):

- 1 bit for sign
- **8 bits for exponent**
- 23 bits for mantissa (fraction)

---

8. What is the approximate range of values for single-precision floating-point numbers?
(a) 1.4×10^{-45} to 3.4×10^{38}
(b) 1.4×10^{-38} to 3.4×10^{38}
(c) 4.9×10^{-324} to 1.8×10^{308}
(d) 4.9×10^{-308} to 1.8×10^{324}
✅ **Correct Answer: (a) 1.4×10^{-45} to 3.4×10^{38}**

Explanation:
The range of **IEEE single-precision** floating-point numbers is from **~1.4×10^{-45} to 3.4×10^{38}**, supporting a wide span of small and large real numbers.

---

9. What are the tiny dots that make up an image called?
(a) Pixels
(b) Bits
(c) Bytes
(d) Nodes
✅ **Correct Answer: (a) Pixels**

Explanation:
Pixels (short for "picture elements") are the **smallest units of an image**, arranged in a grid to form pictures on screens.

---

10. 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
✅ **Correct Answer: (a) Red, Green, Blue**

Explanation:
RGB is a color model where colors are created by combining **Red, Green, and Blue** light in different intensities.

✅ Solved Short Questions

---

1. What is the primary purpose of the ASCII encoding scheme?
✅ **Answer:**
The main purpose of the **ASCII (American Standard Code for Information Interchange)** encoding scheme is to represent **text characters (letters, digits, symbols)** in binary form, so that computers can store, process, and transmit them.

---

2. Explain the difference between ASCII and Unicode.
✅ **Answer:**

- **ASCII** uses 7 bits and can represent **128 characters**, mostly English letters and symbols.
- **Unicode** uses up to **32 bits**, and can represent **characters from almost all languages** (e.g., Arabic, Chinese, Urdu), along with symbols and emojis.

---

3. How does Unicode handle characters from different languages?
✅ **Answer:**
Unicode assigns a **unique code point** to every character of every language. It supports **multi-language** text by providing a standard number (code) for each character, ensuring compatibility across platforms.

---

4. What is the range of values for an unsigned 2-byte integer?

✅ Answer:

A **2-byte unsigned integer** means the number is stored using:

- **2 bytes = 16 bits**
- **Unsigned** means **no negative numbers**, only **0 and positive integers**

◈ Formula for Range of Unsigned Integers:

For **n bits**, the range is:
0 to ($2^n - 1$)

Here, n = 16 (because 2 bytes = 16 bits)

→ $2^{16} = 65,536$

So the range is:

✅ **0 to 65,535**

▤ Final Answer:

The range of values for an unsigned 2-byte integer is from 0 to 65,535.

---

5. Explain how a negative integer is represented in binary.

✅ **Answer:**

Negative integers are stored using a method called **Two's Complement**.

- First, the binary of the positive number is written.
- Then, invert all bits and add 1.
  This allows computers to perform arithmetic with both positive and negative numbers easily.

---

6. What is the benefit of using unsigned integers?

✅ **Answer:**

Unsigned integers can represent a **larger range of positive numbers** using the same number of bits because no bit is needed to store the sign.

7. How does the number of bits affect the range of integer values?

✅ **Answer:**

More bits mean a **wider range of values**.

- For **n bits**, an **unsigned** integer can store values from **0 to (2^n - 1)**.
- A **signed** integer can store values from **-2^{n-1} to 2^{n-1} - 1**.

---

8. Why are whole numbers commonly used in computing for quantities that cannot be negative?

✅ **Answer:**

Because negative values don't make sense in certain contexts (e.g., age, quantity, price), **unsigned whole numbers** are used to save memory and avoid errors.

---

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

✅ **Answer:**

The range is calculated using the **IEEE 754 format**:

- 1 bit for sign
- 8 bits for exponent (with bias of 127)
- 23 bits for mantissa
  This gives an **approximate range of 1.4×10^{-45} to 3.4×10^{38}** for single-precision.

---

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

✅ **Answer:**

Floating-point numbers have **limited precision and range**, which can lead to **rounding errors** or **inaccurate calculations** in scientific and financial applications. Understanding these limitations helps avoid bugs and improves accuracy.

✅ Long Questions – Solved (Detailed Answers)

Chapter 2
Punjab Textbook – 9th Class Computer Science

---

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

✅ **Answer:**
Unicode is a **universal character encoding standard** used to represent text in different languages. Each character is assigned a **unique code point**, written as **U+XXXX**, where "XXXX" is a hexadecimal value.

🔷 *Key Features:*

- Supports over **143,000 characters**.
- Includes characters from languages like **English, Arabic, Chinese, Urdu, Hindi**, etc.
- Unicode can be **8, 16, or 32 bits** depending on encoding (UTF-8, UTF-16, UTF-32).

🔷 *Examples:*

Character	**Language**	**Unicode Code Point**
A	English	U+0041
ب	Urdu/Arabic	U+0628
क	Hindi	U+0915
中	Chinese	U+4E2D

Unicode ensures text from different languages can be stored and processed together without conflict.

---

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

✅ Answer:

Computers store **integers (whole numbers)** in **binary form** using **bits** (0s and 1s). Each bit represents a power of 2, and the total number of bits used determines the **range** of values that can be stored.

🔷 1. Types of Integers:

a) Unsigned Integers

- Store **only positive numbers** (including 0)
- All bits are used for value
- No sign bit
- Example (4 bits):

- `0000` = 0
- `1111` = 15
- Range: **0 to (2^n – 1)**

b) Signed Integers

- Store **both positive and negative numbers**
- The **leftmost bit** is used for the **sign**:
  - `0 = Positive`, `1 = Negative`
- Represented using **Two's Complement** method

🔹 2. Two's Complement (for Negative Numbers)

This is the most common method for storing **negative integers**.

➤ Steps to store a negative integer:

1. Write the **binary** of the positive number
2. **Invert all bits** (1's complement)
3. **Add 1** → result is the **Two's complement**

➤ Example: Store -5 in 8 bits

1. +5 in binary: `00000101`
2. Invert: `11111010`
3. Add 1: `11111011` ✅
   So, **-5 = 11111011** in 8-bit two's complement

🔹 3. Bit Length and Storage Capacity

Bit Size	Unsigned Range	Signed Range
8 bits	0 to 255	-128 to 127
16 bits	0 to 65,535	-32,768 to 32,767
32 bits	0 to 4,294,967,295	-2,147,483,648 to 2,147,483,647

---

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

✅ **Answer:**

◆ 1. Converting Decimal to Binary (Positive Integer)

To convert a **positive decimal number** to **binary**, divide the number by 2 repeatedly and write down the remainders. Then reverse the order of remainders.

➤ Example: Convert 13_{10} to binary

```
13 ÷ 2 = 6   remainder = 1
 6 ÷ 2 = 3   remainder = 0
 3 ÷ 2 = 1   remainder = 1
 1 ÷ 2 = 0   remainder = 1
```

Now reverse the remainders:
✓ **$13_{10} = 1101_2$**

◆ 2. Converting Decimal to Binary (Negative Integer)

Negative integers are stored in binary using the **Two's Complement** method.

➤ Steps to convert -5 to binary (8-bit):

1. Write binary of +5 → `00000101`
2. Invert all bits → `11111010`
3. Add 1 → `11111011`

✓ So, **-5 = 11111011_2** (in 8-bit two's complement form)

◆ 3. Converting Binary to Decimal (Positive)

Multiply each binary digit by 2 raised to the power of its position (from right to left), then add.

➤ Example: Convert 1011_2 to decimal

makefile
CopyEdit
$1 \times 2^3 = 8$
$0 \times 2^2 = 0$
$1 \times 2^1 = 2$
$1 \times 2^0 = 1$
Total = 8 + 0 + 2 + 1 = ✓ 11_{10}

◆ 4. Converting Binary to Decimal (Negative Two's Complement)

1. Identify that the binary number is **negative** if the leftmost bit is `1`
2. Take **Two's Complement** to get magnitude:
   - Invert all bits
   - Add 1
3. Convert result to decimal, then add negative sign

➤ *Example: Convert 11111011_2 to decimal*

- It's negative (starts with `1`)
- Invert: `00000100`
- Add 1: `00000101` → 5

✅ Final answer = **-5_{10}**

◈ Decimal to Binary (Negative using 8-bit Two's Complement):

Example: -5

1. Binary of 5 = 00000101
2. Invert = 11111010
3. Add 1 = **11111011**

◈ Binary to Decimal:

Multiply each bit by 2^position and add.

Example: Binary 1011
$= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$
= 8 + 0 + 2 + 1 = **11**

---

4. Perform the following binary arithmetic operations:

(a) Multiply 101 by 11

```
markdown
   101     (5)
×   11     (3)
-------
   101     (5 × 1)
+1010     (5 × 1 with shift)
-------
 1111     ✅ (15 in decimal)
```

(b) Divide 1100 by 10

1100 ÷ 10 →
Binary:

1100 (12) ÷ 10 (2) = **110** ✅ (6 in decimal)

110
____
10 | 1100
10 ← $10_2 \times 1 = 10_2$
----
10 ← bring down next bit
10 ← $10_2 \times 1 = 10_2$
----
0 ← remainder

---

5. Add the following binary numbers:

(a) 101 + 110

```
  101
+ 110
-----
 1011 ✅ (11 in decimal)
```

(b) 1100 + 1011

```
1100
+1011
-----
10111 ✅ (23 in decimal)
```

---

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

(a) 7 + (-4)

7 → 0111
-4 → Two's complement of 0100 = 1100

```
0111
+1100
-----
10011 → Discard carry → **0011 ✅ (3)**
```

(b) -5 + 3

-5 → 1011
3 → 0011

```
1011
+0011
-----
1110 ✅ (which is -2 in Two's complement)
```

---

7. Solve the following binary subtractions:

(a) 1101_2 - 0100_2

1 1 0 1 (13 in binary)

- 0 1 0 0 (4 in binary)

----

1 0 0 1 ✓

= 13 - 4 = **1001** ✓ (9)

(b) 1010_2 - 0011_2

1 0 1 0

- 0 0 1 1

----

0 1 1 1 ✓

= 10 - 3 = **0111** ✓ (7)

(c) 1000_2 - 0110_2

1 0 0 0

- 0 1 1 0

----

0 0 1 0 ✓

= 8 - 6 = **0010** ✓ (2)

(d) 1110_2 - 0100_2

1 1 1 0

- 0 1 0 0

----

1 0 1 0 ✓

= 14 - 4 = **1010** ✓ (10)