DATASCI 350 - Data Science Computing

Assignment 01 - Computational Literacy

Instructions

This assignment evaluates your understanding of topics covered in the first two weeks of class, including binary and hexadecimal number systems, ASCII encoding, and programming language fundamentals.

You must complete this assignment individually. While you may use available resources such as notes, books, and AI tools, you are expected to submit original work. Please acknowledge all resources used, including input from classmates and AI. If you are unsure about permissible resources or proper acknowledgement, please consult the instructor.

Present your solutions clearly and systematically, showing your problem-solving process. Please ensure that any code is well-commented.

Submission

Please submit your solutions as either a single Jupyter notebook or a PDF file. Follow the instructions provided in each section carefully, and please submit your completed assignment to Canvas.

Question 01

Convert the decimal number 53 to binary. Show your work.

Answer:

53/2 = 26...1 26/2 = 13...0 13/2 = 6...1 6/2 = 3...0 3/2 = 1...1 1/2 = 0...1

Therefore, 53 to binary is 110101

Question 02

Convert the binary number 1011001 to decimal. Show your work.

Answer:

2^0+2^3+2^4+2^6 = 89

Question 03

What is the hexadecimal representation of the RGB colour (128, 64, 255)? Explain your answer.

Answer:

128/16 = 8...0 64/16 = 4...0 255/16 = 15...15

Therefore, the hexadecimal representation is 8040FF

Question 04

Convert the hexadecimal colour #2A9F3B to its RGB components. Show your steps.

Answer:

2A = 2*16 + A = 42 9F = 9*16 + F = 159 3B = 3*16 + B = 59 RGB (42, 159, 59)

Question 05

Using the coin representation system described in the lecture (c[quarters][dimes][nickels][pennies]), convert $1.37 to coin representation. Explain your reasoning.

Answer:

1.37/0.25 = 5...0.48 (you need 5
$0.25)1.37 - 5 * 0.25 = 0.120.12/0.10 = 1.2(youneed1$0.10) 0.12-0.10 = 0.02
0.02/0.05 = 0.4 (you need 0 $0.05)youneed2$0.01

$1.37 converts to coin representation is c5102

Question 06

What is the Unicode representation of the word "Emory"? Use the Unicode table provided in the lecture to find out.

Answer:

"Emory" in Unicode: \u0045\u006D\u006F\u0072\u0079

Question 07

Explain the difference between ASCII and Unicode. Why was Unicode developed, and what advantages does it offer over ASCII?

Answer:

ASCII is an early character encoding system that uses 7 bits with 128 possible characters and only works for basic English. Unicode is a modern character encoding standard that aims to represent every character in every language, such that it uses variable length encoding (can be 8, 16, or 32 bits) depending on the encoding (UTF-8, UTF-16, UTF-32).

Unicode was developed because ACII was too limited, such that the latter fails to represent accented letters and non-Latin languages. ASCII was designed primarily for English and cannot represent characters from languages such as Chinese, Arabic, Hindi, or Russian. Unicode provides a unified standard that allows all of these languages, and more, to be represented in the same system. This universality ensures that text can be shared and displayed correctly across different computers, software, and platforms worldwide.

Unicode is also backward compatible with ASCII. The first 128 Unicode characters are identical to the ASCII set, which means that older ASCII text files remain valid under Unicode. Additionally, Unicode's use of variable-length encodings (such as UTF-8) provides flexibility, allowing it to efficiently store English text while still being capable of encoding more complex characters when needed. Overall, Unicode solves the limitations of ASCII, enables global communication, and supports modern text and symbols, making it essential for today's digital world.

Question 08

Describe the Von Neumann architecture and its significance in modern computing. What is the Von Neumann bottleneck, and how does it affect computer performance?

Answer:

Von Neumann architecture is a classic computer design in which both program instructions and data are stored in the same memory and accessed by a CPU. The CPU includes a control unit and an arithmetic logic unit, and it communicates with memory and input/output devices through a shared bus. Its significance lies in the stored-

program concept, which allows computers to be easily reprogrammed and has shaped the design of most modern general-purpose computers.

The Von Neumann bottleneck is the performance limitation caused by the CPU and memory sharing a single communication path. Since instructions and data must travel over the same bus, the CPU can be forced to wait for memory access, slowing execution. Although techniques like caching and pipelining reduce this problem, the bottleneck remains a fundamental challenge in computer performance.

Question 09

Compare and contrast low-level and high-level programming languages. Give two examples of each and explain when you might choose to use one over the other.

Answer:

Low-level and high-level programming languages differ mainly in how close they are to machine hardware vs. human language. Low-level languages, such as machine code (binary instructions executed directly by the CPU) and assembly language (mnemonics closely representing machine instructions), provide minimal abstraction and give programmers fine-grained control over hardware resources like memory and registers. These languages are useful when you need high performance, precise timing, or direct hardware interaction—for example in operating system kernels, embedded systems, or device drivers—but they are harder to write, read, and maintain because the programmer must manage many details manually.

High-level languages like Python and Java sit much further from hardware, offering strong abstraction with features such as automatic memory management, clear syntax, and rich libraries that simplify complex tasks for developers. These languages are typically chosen for application development, rapid prototyping, data analysis, and web programming because they make code easier to write and port across different systems. The trade-off is that high-level languages may introduce some performance overhead compared with low-level code, but they greatly speed up development and reduce bugs.

Question 10

Discuss the concept of abstraction in computer science, using the representation of images in computers as an example. How does this abstraction process impact data analysis and predictive modelling in image-related tasks?

Answer:

Abstraction in computer science refers to hiding unnecessary details to focus on higher-level concepts, allowing programmers and systems to work without knowing all the low-level mechanics. For example, computers represent images not as pictures humans see but as grids of pixels stored as numerical values (e.g., color components stored as bits). At the lowest level, each pixel's data is just bits in memory; higher levels of abstraction let us think of images as objects we can display, edit, or analyze without worrying about underlying storage. This process of abstraction is crucial in data analysis and predictive modeling involving images: models need to operate on numerical representations of pixels or features derived from them rather than raw pictures. By abstracting images into arrays of numbers, algorithms can detect patterns, learn from data, and make predictions (e.g., classifying objects in images) without human intervention in how pixels are stored. Effective abstraction thus bridges human concepts (like "cat" or "edge") and machine representations, making complex tasks tractable and enabling powerful image-based analysis.