Multiple Choices Questions (60 Points)

After answering all knowledge questions, transfer your solution letter to the table below. Only one solution is correct for each answer option. Please mark your solution clearly:

Questions 1.1 to 1.10

	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	1.10
Answer	C	B	C	B	D	D	B	B	A	B
Points										

Questions 1.11 to 1.20

	1.11	1.12	1.13	1.14	1.15	1.16	1.17	1.18	1.19	1.20
Answer	D	A	B	C	D	B	B	C	B	A
Points										

Question 1.1 | Access Speed (3 Points)

Which of the following computer memory circuits has the highest access speed?

- ☐ `A: SRAM`
- ☐ `B: DRAM`
- ☐ `C: Register`
- ☐ `D: SSD`

Question 1.2 | Cache Prediction (3 Points)

Which principle suggests that recently accessed data will likely be used again soon?

- ☐ `A: Spatial Locality`
- ☐ `B: Temporal Locality`
- ☐ `C: Data Locality`
- ☐ `D: Cache Locality`

Question 1.3 | Byte (3 Points)

In the context of binary number measurement, what does a byte represent?

- ☐ `A: A single bit representing two possible values: 0 and 1.`
- ☐ `B: A group of four bits that can represent 2^4 possibilities.`
- ☐ `C: A group of eight bits that can represent 2^8 possibilities.`
- ☐ `D: A group of sixteen bits that can represent 2^16 possibilities.`

Question 1.4 | Clock Signal (3 Points)

What does a clock signal in computer systems define?

- ☐ `A: The data transfer rate.`
- ☐ `B: The frequency of processor operations.`
- ☐ `C: The sampling rate of analog-to-digital conversion.`
- ☐ `D: The speed of internet connections.`

Question 1.5 | RodRego (3 Points)

Which of the following is NOT one of the three instructions available in RodRego?

- ☐ A: INC
- ☐ B: DEB
- ☐ C: END
- ☐ D: MUL

Question 1.6 | Preloading Data (3 Points)

Consider the following Python code:

```
matrix = [
   [ 1,  2,  3,  4], [ 5,  6,  7,  8],
   [ 9, 10, 11, 12], [13, 14, 15, 16]
]

print("Accessed element:", matrix[2][1])
```

Which element is likely loaded into cache immediately after accessing matrix[2][1]?

- ☐ A: matrix[3][1]
- ☐ B: matrix[3][0]
- ☐ C: matrix[1][1]
- ☐ D: matrix[2][2]

Question 1.7 | Memory Hierarchy (3 Points)

Which of the following computer memory circuits is usually used to build cache?

- ☐ A: SSD
- ☐ B: SRAM
- ☐ C: DRAM
- ☐ D: Register

Question 1.8 | Boolean Expression (3 Points)

Identify the option below that is logically equivalent to the following Python expression:

not (x <= 5 and y > 10 or z == 0)

- ☐ `A: x > 5 and y <= 10 or z != 0`
- ☐ `B: x > 5 or y <= 10 and z != 0`
- ☐ `C: x > 5 or y <= 10 or z != 0`
- ☐ `D: x <= 5 or y > 10 and z != 0`

Question 1.9 | Debugging Output (3 Points)

Consider the following Python code:

```
def modify_lst(lst, value):
    lst[0].append(value)

my_lst = [[1],2,3]
modify_lst(my_lst, 4)
modify_lst(my_lst, 5)
modify_lst(my_lst, 6)
print(my_lst)
```

What is the output of this code?

- ☐ `A: [[1, 4, 5, 6], 2, 3]`
- ☐ `B: [[1], 2, 3]`
- ☐ `C: [[1, 6], 2, 3]`
- ☐ `D: The program can not be executed.`

Question 1.10 | Exception Propagation (3 Points)

What happens if an exception is raised in a try block with no matching except clause?

- ☐ `A: The exception is ignored, and the program continues.`
- ☐ `B: The exception propagates up to the caller.`
- ☐ `C: The else block is executed instead.`
- ☐ `D: Python automatically adds a default except block.`

Question 1.11 | Truth Table (3 Points)

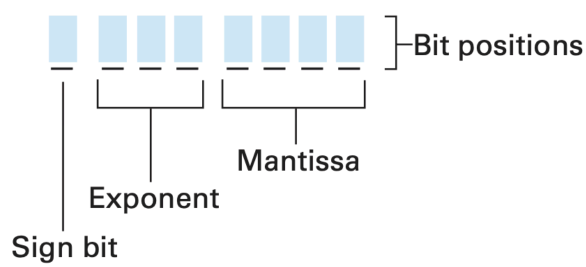
What is a truth table, and which of the following problems can it solve?

- ☐ A: A diagram of logical gates; it can solve geometry and probability problems.
- ☐ B: A chart displaying binary numbers; it can solve numerical and statistical problems.
- ☐ C: A table listing variables and values; it can solve algebraic equations and calculus problems.
- ☐ D: A table showing all possible logical values for a Boolean expression; it can solve logical equivalence and circuit design problems.

Question 1.12 | Floating-point Notation (3 Points)

What does 11011101 denote in floating-point notation (illustrated below)?

- ☐ A: -1 5/8
- ☐ B: 3 1/4
- ☐ C: -3 1/4
- ☐ D: 6 1/2



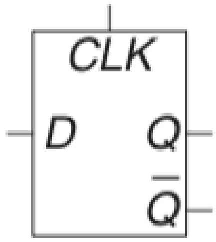
Question 1.13 | Von Neumann Architecture (3 Points)

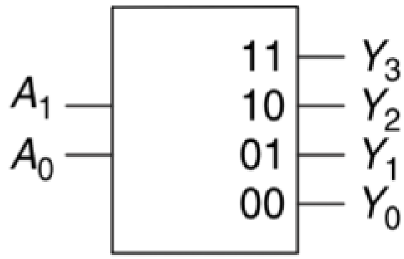
Von Neumann Architecture is the blueprint for all of the modern computers. Which part of it is the implementation of the tape of a Turing machine?

- ☐ A: CPU
- ☐ B: Memory
- ☐ C: Program
- ☐ D: None of the above

Question 1.14 | Memory Blocks (3 Points)

What are the names of the following two memory blocks?

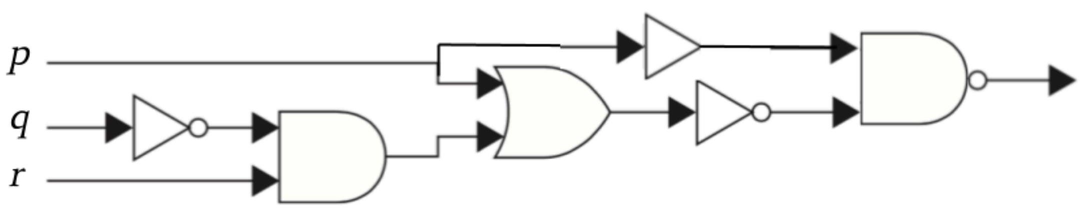




- ☐ A: Decoder; Mux
- ☐ B: Flip-flop; Decoder
- ☐ C: D latch; Decoder
- ☐ D: Mux; Flip-flop

Question 1.15 | Circuit Gates (3 Points)

Someone designed a circuit block like below. What is the boolean equivalent of it?



- ☐ A: ¬(p × (q + r))
- ☐ B: p × (¬q + r)
- ☐ C: ¬(p × (¬q + r))
- ☐ D: Always true

Question 1.16 | Computer Memory (3 Points)

Which of the following statements about computer memory is FALSE?

- ☐ A: Data in RAM is volatile and will be erased once power is off.
- ☐ B: Data stored in hard drive (HDD) can be directly accessible by CPU depending on where the data is stored.
- ☐ C: Once a computer is turned on, program stored in ROM will be executed immediately to load the operating system into RAM.
- ☐ D: Although SSD runs faster than HDD, SSD is not as reliable as HDD for long-term archiving of data.

Question 1.17 | Machine Language (3 Points)

Which of the following statements about machine language and high-level programming languages is TRUE?

- ☐ A: Machine language code requires fewer resources to write and is easier to debug.
- ☐ B: High-level programming languages can not be understood by machines directly and require compiling.
- ☐ C: High-level programming language code runs faster as it hides unnecessary details for programmers.
- ☐ D: Machine language code is portable across different computers.

Question 1.18 | Python Data Types (3 Points)

Which of the following data types in Python are mutable?

```
int, list, dictionary, str, tuple
```

- ☐ A: str, tuple
- ☐ B: list, dictionary, str
- ☐ C: list, dictionary
- ☐ D: list

Question 1.19 | Turing Machine (3 Points)

Which of the following statements is TRUE regarding the Turing machine?

- ☐ `A: Turing machines can only execute basic arithmetic operations.`
- ☐ `B: A programming language (e.g., Python) can be a Turing machine.`
- ☐ `C: Every Turing machine can determine in advance whether it will halt or run forever.`
- ☐ `D: The Halting Problem refers to a Turing machine running out of tape while performing computations.`

Question 1.20 | Python Syntax (3 Points)

Which of the following Python codes is <u>NOT</u> correct for calculating the square root of the variable “x”? (Suppose the math package is already imported.)

- ☐ `A: y = x ^ 0.5`
- ☐ `B: y = x ** 0.5`
- ☐ `C: y = math.sqrt(x)`
- ☐ `D: y = math.pow(x, 0.5)`

Programming Questions (40 Points)

Please write your programming solution as clearly as possible.
Try to add comments to explain each code segment.

Question 2.1 | Number Conversion (10 Points)

In the lecture, we learned how to convert between binary and decimal numbers using the following two examples. In this task, you will generalize this to any inputs - **pay attention to input type**. Note you can not call Python built-in functions int/bin().

Task 1: Binary to Decimal (5 Points)

Converting binary to decimal

Base 2 to base 10: $10110_2 = ?_{10}$

- In a binary number, each column has twice the weight of the previous column, so for i-th column, adding $D_i \times 2^{i-1}$.

  $10110_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22_{10}$

```
def bin2dec(bin_str: str):
      # Please write your code here
```

Task 2: Decimal to Binary (5 Points)

Converting decimal to binary

Base 10 to base 2: $84_{10} = ?_2$

- Repeatedly divide the number by 2; the remainder goes in each column (from right to left).
  - 84/2 = 42, so 0 goes in the most right column
  - 42/2 = 21, so 0 goes in the 2nd right column
  - 21/2 = 10, with a remainder of 1 going in the 3rd right column
  - 10/2 = 5, so, 0 goes in the 4th right column
  - 5/2 = 2, with a remainder of 1 going in the 5th right column
  - 2/2 = 1, so, 0 goes in the 6th right column
  - 1/2 = 0, with a remainder of 1 going in the 7th right column

  So, $84_{10} = 1010100_2$.

```
def dec2bin(dec_num: int):
      # Please write your code here
```

Question 2.2 | Register Simulation (30 Points)

In this task, you will write a Python program to simulate a digital circuit that performs binary subtraction of two binary strings. The simulation will incorporate a **half-subtractor** and a **full-subtractor** to achieve the desired functionality.

Note that the parameters bit, `b_in`, bit1, and bit2 are strings. You are not allowed to use the Python built-in functions and, or, not, and xor. Your code implementation must be successive, meaning task 2 must rely on the code you wrote in task 1, et cetera.

Task 1: Implement Auxiliary Functions (12 Points)

Implement the following auxiliary functions, ensuring that each function adheres to the specified requirements:

- **NOT:** Simulates the NOT gate. It takes a single bit as input and returns the corresponding bit after negation.
- **AND:** This function simulates the AND gate. It accepts two bits as input and returns the result of their logical AND operation.
- **OR:** Simulates the OR gate. It accepts two bits as input and returns the result of their logical OR operation.
- **XOR:** Simulates the XOR gate. It takes two bits as input and returns the result of their exclusive OR operation.

```
def NOT(bit):
      # Please write your code here
```

```python
def AND(bit1, bit2):
     # Please write your code here
```

```python
def OR(bit1, bit2):
     # Please write your code here
```

```python
def XOR(bit1, bit2):
     # Please write your code here
```

Task 2: Implement Subtractor Functions (12 Points)

Utilizing the auxiliary functions from Task 1, implement the following:

1. **half_subtractor:** This function simulates a half-subtractor. It takes two bits as input and returns two values: the difference and the borrow out.

Circuits for subtracting numbers

- $D = A$ **XOR** B
- B_{out} = (**NOT** A) **AND** B

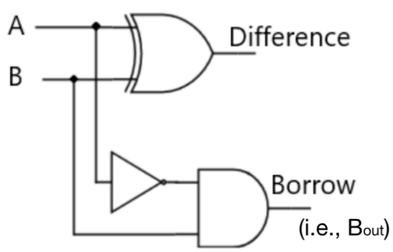


Figure-3:Circuit Diagram of Half subtractor

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Figure-2:Truth Table of Half subtractor

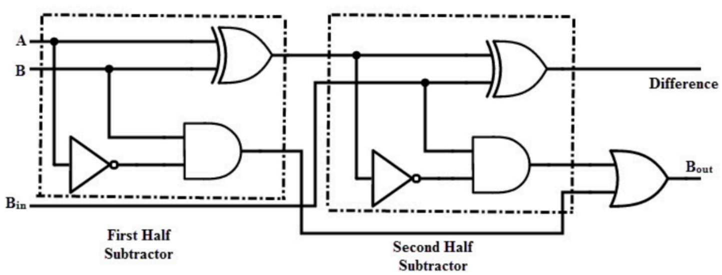
Half Subtractor

```
def half_subtractor(bit1, bit2):
      # Please write your code here
```

2. **full_subtractor:** This function simulates a full-subtractor. It takes three inputs: two bits for the current step and one bit representing the borrow-in from the previous step. The function returns the difference and the borrow out.

Circuits for subtracting numbers

- D = A **XOR** B **XOR** B_{in}
- B_{out} = ((**NOT** A) **AND** B)) **OR** (**NOT**(A **XOR** B) **AND** B_{in})



Full Subtractor

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Figure-5:Truth Table of Full subtractor

```
def full_subtractor(bit1, bit2, b_in):
      # Please write your code here
```

Task 3: Implement the Subtraction Function (6 Points)

Write a subtract function that inputs two binary strings: the subtrahend and the minuend. The function should return a binary string representing the difference between the two binary numbers. The subtrahend is assumed to be greater than or equal to the minuend.

If the binary strings have different lengths, prepend zeros to the shorter string until both have the same length. For example, if the subtrahend is '100' and the minuend is '1', represent '1' as '001' before subtracting.

The functions developed in Tasks 1 and 2 must be utilized to complete this task.

```
def subtractor(subtrahend, minuend):
      # Please write your code here
```