Q1. Multiple Choice Questions.
1.
Ans. (c) Both a and b
2.
Ans. (c) Base raised to an exponent
3.
Ans. (b) What functions have been wired into hardware
4.
Ans. (b) Space and Time

Q2. Fill in the blanks.
1. Data Type
2. Stack
3. Data Structure
4. Abstract Data Type

Q3. True or False.
1. True
2. False
3. True
4. True

Q4. Answer the Following Questions.

1. Explain data type and why do we need data types?
    - A method of interpreting a bit pattern is called a data type.
    - The term data type refers to the implementation of the mathematical model specified by an ADT(Abstract Data Type).
    - A data type is the abstract concept defined by a set of logical properties.

    We need data types because of following reasons -
    - Even small data can occupy a large volume of memory.
    - Difference data types need different interpretations of the same bit strings.
    - Using one universal data type would make interpretation very hard.

2. Explain Abstract data type with an example.
    - An Abstract data type is a mathematical model that defines:
        - The data objects that make up a data type.
        - The operations that can be performed on those objects.

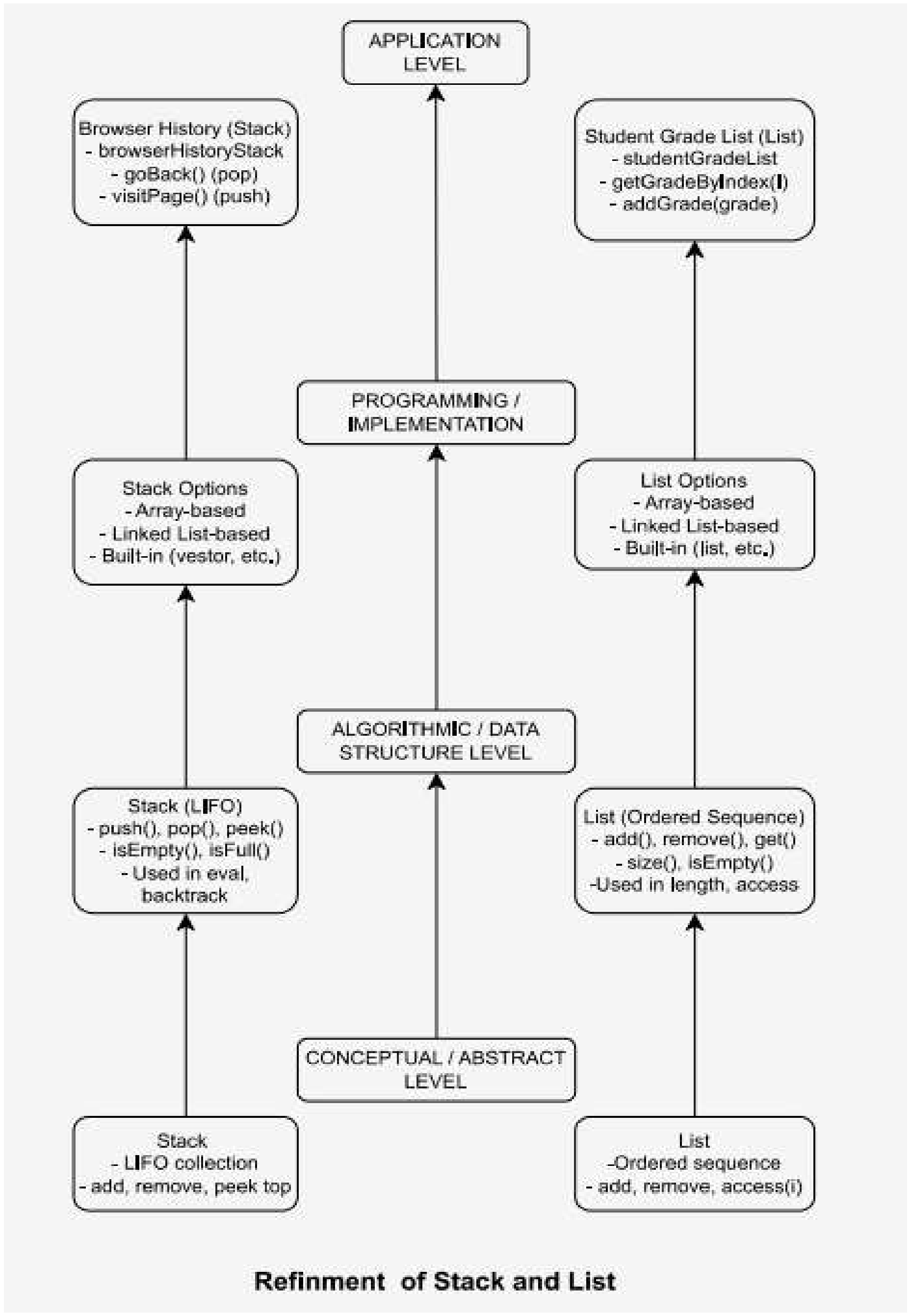
    - An ADT specifies the logical and mathematical properties of the data type, but it does not specify how it is implemented in a programming language.
    - Example -

- STACK-ADT
- A stack is a collection of elements where insertion and deletion happen at one end only (the top).
- Properties of Stack-ADT:
    - PUSH(data): Insert a new element at the top.
    - POP(): Remove the topmost element.
    - Condition: Only the most recently inserted element can be accessed.

3. What is a data structure and what are the differences between data type, abstract data type and data structure?
    - The term data structure refers to a set of computer variables that are connected in some logical or mathematical manner.
    - More precisely, a data structure can be defined as the structural relationship present within the data set and thus should be viewed as 2 tuples. (N, R) where 'N' is the finite set of nodes representing the data structure and 'R' is the set of relationships among those nodes. For ex - in a tree data structure each node is related to each other in a 'parent child' relationship. Thus, a large volume of data can be represented using a tree data structure and the relationship between each data can also be shown.

    Differences between data type, abstract data type and data structure
    - There seems to be an open relationship between the three, a data type has its root in abstract data type and a data structure comprises a set of computer variables of same or different data types.

4. Draw a diagram for showing levels of refinement for a stack and a list for any problem.


APPLICATION LEVEL
Browser History (Stack) - browserHistoryStack - goBack() (pop) - visitPage() (push)
Student Grade List (List) - studentGradeList - getGradeByIndex(I) - addGrade(grade)
PROGRAMMING / IMPLEMENTATION
Stack Options - Array-based - Linked List-based - Built-in (vestor, etc.)
List Options - Array-based - Linked List-based - Built-in (list, etc.)
ALGORITHMIC / DATA STRUCTURE LEVEL
Stack (LIFO) - push(), pop(), peek() - isEmpty(), isFull() - Used in eval, backtrack
List (Ordered Sequence) - add(), remove(), get() - size(), isEmpty() -Used in length, access
CONCEPTUAL / ABSTRACT LEVEL
Stack - LIFO collection - add, remove, peek top
List -Ordered sequence - add, remove, access(i)
Refinment of Stack and List

5. Develop an ADT specification for “Polynomials”. Also include the operations associated with polynomials.
    - abstract typedef < { (c1, e1), (c2, e2), … , (cn, en) } > POLYNOMIAL
      /* where each pair (ci, ei) represents a term: coefficient ci and exponent ei
        with ei ≥ 0 and no two terms having the same exponent */

      Condition: Terms must be ordered by decreasing exponents (ei > ei+1).

Condition: If coefficient ci = 0, the term may be omitted.

- Operations on POLYNOMIAL
    1. CREATE()
       Precondition: None
       Operation: Construct a new empty polynomial P = { }
       Postcondition: P is initialized with no terms
    2. ADD_TERM(P, c, e)
       Precondition: $e \geq 0$
       Operation: Insert term (c, e) into polynomial P.
       If a term with exponent e exists, add coefficients.
       Postcondition: P contains the new/updated term.
    3. REMOVE_TERM(P,e)
       Precondition: Polynomial P contains term with exponent e.
       Operation: Delete the term with exponent e.
       Postcondition: P does not contain (ci, e).
    4. EVALUATE(P,x)
       Precondition: x is a real number.
       Operation: Compute value = Σ (ci * x^ei) for all terms in P.
       Postcondition: Returns a numerical result.
    5. ADD_POLY(P1, P2)
       Precondition: P1, P2 are valid polynomials.
       Operation: Add corresponding terms with same exponents.
       Postcondition: Returns a new polynomial P3 = P1 + P2.
    6. SUBTRACT_POLY(P1, P2)
       Precondition: P1, P2 are valid polynomials.
       Operation: Subtract corresponding terms.
       Postcondition: Returns a new polynomial P3 = P1 - P2.
    7. MULTIPLY_POLY(P1, P2)
       Precondition: P1, P2 are valid polynomials.
       Operation: Multiply each term of P1 with each term of P2.
       Postcondition: Returns a new polynomial P3 = P1 * P2.
    8. DEGREE(P)
       Precondition: $P \neq \emptyset$
       Operation: Return the largest exponent in P.
       Postcondition: Returns degree of polynomial.
    9. DISPLAY(P)
       Precondition: None
       Operation: Output polynomial in readable form (e.g., 5x^3 + 2x - 7).

6. Suggest a suitable data structure for representation of imaginary numbers . An imaginary number is represented by a+ib where i is the iota for the number. Also give specification for the operation associated with them.
   - We can define a structure of imaginary numbers with two fields:
     - real -> to store the real part (a)
     - imag -> to store the imaginary part (b)
   - Example:

     ```
     Struct Complex {
             float real;
             float imag;
     }
     ```

   - Operations on Complex Numbers.
     1. Addition
        If $x = a + ib$ and $y = c + id$,
        then $x + y = (a + c) + (b + d)i$
     2. Subtraction
        $x - y = (a - c) + (b - d)i$
     3. Multiplication
        $x * y = (ac - bd) + (ad + bc)i$
     4. Division
        $x / y = [(ac + bd) / (c^2 + d^2)] + [(bc - ad) / (c^2 + d^2)]i$
     5. Conjugate
        If $x = a + ib$, then conjugate $= a - ib$
     6. Modulus (Magnitude)
        $|x| = \sqrt{(a^2 + b^2)}$