Data Structure, Midterm

Student ID: ***Double click here to fill the Student ID***

Name: ***Double click here to fill the name***

A. Single Choice Questions (5% each, 60%)

(1) Which of the following is TRUE about constructors in OOP?

(A) Constructors are used to destroy an object of a class.

(B) Constructors are special methods invoked when an object of the class is created.

(C) Constructors are class methods that cannot access an object's variables.

(D) Constructors must always return a value.

(B)

(2) Which concept in OOP best suggests that only necessary information is displayed to the user while the implementation details are hidden?

(A) Encapsulation

(B) Abstraction

(C) Polymorphism

(D) Inheritance

(B)

(3) What does Big O notation primarily describe?

(A) The minimum time a function takes to run

(B) The minimum space a function uses

(C) The exact time a function takes to run in the worst case

(D) The upper bound of the runtime complexity in the worst case

(D)

(4-1) What is the time complexity of `pop()` and `pop(0)` of a list in Python?

(A) $O(n)/O(n)$

(B) $O(1)/O(1)$

(C) $O(n)/O(1)$

(D) $O(1)/O(n)$

(D)

(4-2) What is the time complexity of `pop_back()` and `erase(0)` of a vector in C++?

(A) $O(n)/O(n)$

(B) $O(1)/O(1)$

(C) $O(n)/O(1)$

(D) $O(1)/O(n)$

(D)

(5) Which statement best describes the advantage of an ordered linked list over an unordered linked list when performing search operations?

(A) Ordered linked lists can perform search operations in constant time.

(B) The search operation can stop as soon as it finds an element greater than the one being searched for, potentially reducing the search time.

(C) Unordered linked lists guarantee a faster search operation due to the lack of order.

(D) Both types of lists perform search operations at the same speed and take the same amount of time.

(B)

(6) Considering performance and storage efficiency, in what scenario would an unordered linked list be preferred over a compact array?

(A) When frequent insertions and deletions of elements are needed

(B) When accessing elements by index is the most common operation

(C) When memory space is limited and must be minimized

(D) When the dataset contains a large number of duplicate values

(A)

(7) What mechanism does a sparse array commonly use to efficiently store non-zero values and their indices?

(A) A dictionary/hash table with keys representing indices and values representing non-zero elements

(B) A contiguous block of memory allocated for all possible indices and values

(C) A linked list where each node contains an array of elements

(D) A dynamic array that increases storage space as elements are added

(A)

(8) Which data structure would you choose to implement an undo functionality in a text editor where operations need to be reversed in the exact opposite order of their execution?

(A) Queue

(B) Stack

(C) Dequeue

(D) Sparse matrix

(B)

(9) Consider a pair of corresponding infix and postfix expressions. Which of the following is TRUE?

(A) In a postfix expression, the order of operators is from low to high precedence.

(B) The infix and postfix expressions always have the same number of parentheses.

(C) The infix and postfix expressions always have the same order of operands.

(D) A stack for the operands is required when converting from infix to postfix expressions.

(C)

(10) Consider a sequence of push and pop operations on an initially empty stack. Which of the following sequences cannot occur without checking the stack's contents?

(A) Push, Push, Pop, Push, Pop, Pop

(B) Push, Pop, Push, Push, Pop, Pop

(C) Push, Push, Pop, Pop, Push, Pop

(D) Push, Push, Pop, Pop, Pop, Push

(D)

(11) Which of the following statements is TRUE?

(A) Queue and stack can not be implemented by a linked list.

(B) A stack is a First In, First Out (FIFO) structure.

(C) Using prefix representation eliminates the need for parentheses.

(D) `+-8*237` is not a valid prefix representation.

(C)

(12) Which of the following statements is TRUE?

(A) Recursive functions use a queue implicitly when our computer executes them.

(B) Recursion always uses less memory than the iterative approach.

(C) Recursion is unsuitable for problems that can be broken down into smaller, similar problems.

(D) The minimum number of moves required to transfer n disks from the source to the target in the Tower of Hanoi problem is $2^n - 1$

(D)

B. Short-answer questions, please provide the derivation for each question along with your answer (8% each, 40%).

(13) Briefly explain each of the following terms.

(a) Encapsulation

(b) Compact array

(c) Three laws of recursion

(d) Abstract data type

(a) **Encapsulation**: A principle in object-oriented programming where data and the methods that manipulate the data are bundled together within a class, restricting access to the data directly and ensuring it can only be modified through these methods. This protects the integrity of the data.

(b) **Compact Array**: A compact array is used in programming to directly storing the objects themselves. This approach is beneficial when we want to optimize the storage and it is use in language like C++ or C.

(c) **Three Laws of Recursion**:

1. **Base case**: Every recursive function must have a stopping condition to prevent infinite loops.
2. **Change state**: It must change its state and move toward the base case in each call.
3. **Recursive call**: The function must call itself with the updated state until it reaches the base case.

(d) **Abstract Data Type (ADT)**: A concept in computer science that defines a data structure purely by the operations that can be performed on it and the mathematical properties of those operations, without any regard for how the operations are implemented. This abstraction allows for the implementation details to vary, as long as the interface and behavior from the user's perspective remain consistent.

(14-1) Evaluate the time complexity of the following code segment in terms of Big-O notation (You can consider n as the power of 2 for approximation):

```
i=1
while i <= n:
    for j in range(1, i + 1):
        for k in range(1, n + 1):
            x += 1
    i *= 2
```

```
for i in range(1, n + 1):
    for j in range(1, i + 1):
        for k in range(1, j + 1):
            x -= 1
```

When only considered i and j:

i	1	2	4	8	...	n
j	1	1,2	1,2,3,4	1~8	...	1~n
#	1	2	4	8	...	n
#	2^0	2^1	2^2	2^3	...	$2^{\log}_{n}$

$2^0 + 2^1 + 2^2 + \ldots + 2^{\log n} = \frac{2^0(2^{\log n}-1)}{2-1} = 2^{\log n+1} - 1 = 2n - 1$

When consider k -> $(2n-1) * n = O(n^2)$

==========

$\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j} 1 = \sum_{i=1}^{n}\sum_{j=1}^{i} j = \sum_{i=1}^{n} \frac{(i+1)(i)}{2} = \frac{1}{2}\sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} \frac{i}{2} = \frac{1}{2} * \frac{(n)(n+1)(2n+1)}{6} + \frac{(n)(}{}$

So the total Big-O is $O(n^3)$

(14-2) Evaluate the time complexity of the following code segment in terms of Big-O notation (You can consider n as the power of 2 for approximation):

```
for (i = 1; i <= n; i *= 2){
    for (j = 1; j <= i; j++){
        for (k = 1; k <= n; k++){
            x++;
        }
    }
}

for (i = 1; i <= n; i++){
    for (j = 1; j <= i; j++){
        for (k = 1; k <= j; k++){
            x--;
        }
    }
}
```

When only considered i and j:

i	1	2	4	8	...	n
j	1	1,2	1,2,3,4	1~8	...	1~n
#	1	2	4	8	...	n
#	2^0	2^1	2^2	2^3	...	$2^{\log}_{n}$

$2^0 + 2^1 + 2^2 + \ldots + 2^{\log n} = \frac{2^0(2^{\log n}-1)}{2-1} = 2^{\log n+1} - 1 = 2n - 1$

When consider k -> $(2n-1)*n = O(n^2)$

==========

$\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j}1 = \sum_{i=1}^{n}\sum_{j=1}^{i}j = \sum_{i=1}^{n}\frac{(i+1)(i)}{2} = \frac{1}{2}\sum_{i=1}^{n}i^2 + \sum_{i=1}^{n}\frac{i}{2} = \frac{1}{2} * \frac{(n)(n+1)(2n+1)}{6} + \frac{(n)(}{}$

So the total Big-O is $O(n^3)$

(15) Considering the evaluation of the postfix expression `26*4/39-*`:

(a) What is the result of the expression?

(b) When using a stack for evaluation, as described in our class, what is the maximum number of elements that can be present in the stack at any moment during the evaluation?

(a) The result is $((2*6)/4)*(3-9) = -18$

(b) 3 and the stack is as follows where the right hand side is the top of the stack.

`[2]`

`[2,6]`

`[12]`

`[12,4]`

`[3]`

`[3,3]`

`[3,3,9]`

`[3,-6]`

`[-18]`

(16) Assuming that we have a deque `DQ` that implements the operations defined in deque ADT. What will be the content of the deque after the following operation and why? Be sure to point out the front and rear of the deque in your answer.

```
DQ = deque()
DQ.add_front(6)
DQ.remove_rear()
DQ.add_rear(20)
DQ.add_rear(15)
DQ.remove_front()
if (DQ.is_empty()) DQ.remove_rear()
DQ.add_rear(2)
DQ.add_front(4)
```

After the first step: [6]

After the second step: []

After the third step: [20]

After the fourth step: [20,15]

After the fifth step: [15]

After the sixth step: [15]

After the seventh step: [15,2]

After the eightth step: [4,15,2]

The content will be [4,15,2] where 4 is in the front and 2 is in the rear.

(17) Suppose that `A[3][2]` is located at address 67, `A[4][5]` is located at address 143, `A[1][4]` is located at address 107. What is the base address of the array element `A[0][0]`? Is it column-major or row-major? What is the total number of rows/columns?

Assume `A[i][j]` = $x + \alpha * i + \beta * j$

We have

$$67 = \alpha * 3 + \beta * 2 + x$$

$$143 = \alpha * 4 + \beta * 5 + x$$

$$107 = \alpha * 1 + \beta * 4 + x$$

$\alpha = 4, \beta = 24, x = 7$

The base address is 7

It is column major layout since $\alpha < \beta$ and the size of element is 4.

The total number of rows is 6 and the total number of columns is unkown.