Data Structure, Quiz 1

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%)

(1-1) What is the output of the following Python code?

```python
class Counter:
    count = 0
    def __init__(self):
        Counter.count += 1
        self.id = Counter.count

c1 = Counter()
c2 = Counter()
c3 = Counter()
print(Counter.count, c2.id)
```

(A) `0 2`

(B) `3 2`

(C) `3 3`

(D) `1 2`

(1-2) What is the output of the following C++ code?

```cpp
#include <iostream>
using namespace std;

class Counter {
public:
    static int count;
    int id;
    Counter() {
        count++;
        id = count;
    }
};
int Counter::count = 0;

int main() {
    Counter c1, c2, c3;
    cout << Counter::count << " " << c2.id << endl;
}
```

(A) `0 2`

(B) `3 2`

(C) `3 3`

(D) `1 2`

(2) Four approaches for checking whether two strings are anagrams are listed below. Which approach is BOTH correct AND has the best time complexity?

(A) For each character in `s1`, linearly scan `s2` to find and remove a matching character — $O(n^2)$

(B) Sort both strings and compare them character by character — $O(n \log n)$

(C) Count character frequencies using a fixed-size array of 26 counters (one per letter) for each string, then compare — $O(n)$

(D) Convert both strings to Python `set` objects and compare — $O(n)$

(3) Which of the following best describes the concept of a "time-space tradeoff" in algorithm design?

(A) Algorithms that use more memory always run faster than those using less memory.

(B) Using additional memory (e.g., a lookup table or cache) can sometimes reduce computation time, and vice versa.

(C) The time complexity and space complexity of any algorithm are always equal.

(D) Reducing the number of lines of code always improves both time and space efficiency.

(4) The following four approaches all produce a list/array containing n integers. Which approach has the WORST time performance?

(A) Preallocate an array of size n, then fill each position by index.

(B) Start with an empty dynamic array and repeatedly `append` / `push_back` each element to the end.

(C) Start with an empty dynamic array and, for each new element, create a brand-new array by concatenating the old array with the new element.

(D) Use a built-in function to generate the entire sequence at once (e.g., Python's `list(range(n))` or C++'s `std::iota`).

(5) Which of the following is an advantage of a **compact array** (e.g., a C/C++ integer array or a NumPy integer array) over a **referential array** (e.g., a Python list)?

(A) A compact array can store elements of different data types within the same array.

(B) A compact array stores actual data values contiguously without needing extra references or pointers, making it more memory-efficient.

(C) Appending an element to the end of a compact array is always faster.

(D) A compact array can be dynamically resized with no overhead whatsoever.

(6) Consider a singly linked list that maintains only a `head` reference (no `tail` reference). What is the time complexity of appending a new element to the **end** of a list with n nodes?

(A) $O(1)$, because we can directly access the last node

(B) $O(n)$, because we must traverse the entire list to reach the last node

(C) $O(\log n)$, because we can use a form of binary search on the list

(D) $O(n^2)$, because each traversal step requires checking all previous nodes

(7) In a **circularly linked list**, what does the `next` pointer of the **last node** point to?

(A) `None` / `NULL`, indicating the end of the list

(B) The last node itself

(C) The first node (head) of the list

(D) The second node of the list

(8) When converting a decimal number to binary using repeated division by 2, the remainders are produced from the **least significant bit (LSB)** to the **most significant bit (MSB)**. To output the bits in the correct order (MSB first), which data structure is most appropriate?

(A) Queue — FIFO ensures elements are output in the same order they were added

(B) Stack — LIFO reverses the order, placing the first-computed remainder (LSB) at the bottom and the last-computed remainder (MSB) at the top

(C) Deque — elements can be inserted at either end

(D) A sorted list — automatically arranges elements in ascending order

(9) What is the result of evaluating the postfix expression `3 4 + 2 * 7 -`?

(A) 7

(B) 14

(C) 1

(D) 21

(10) What is the key difference between the `add` operation of an **ordered linked list** and that of an **unordered linked list**?

(A) The ordered linked list's `add` inserts at the head, just like the unordered version.

(B) The ordered linked list's `add` must traverse the list to find the correct position that maintains sorted order.

(C) The ordered linked list's `add` always inserts at the tail.

(D) The ordered linked list's `add` uses binary search to find the insertion point.

(11) Which of the following problems is BEST solved using a Queue?

(A) Implementing the Undo/Redo functionality in a text editor

(B) Managing print jobs in a printer spooler (first-come, first-served)

(C) Evaluating arithmetic expressions using postfix notation

(D) Implementing the "Back" button in a web browser

(12) When performing a **shallow copy** of a list that contains nested mutable objects (e.g., a list of lists), which of the following is TRUE?

(A) Both the outer list and all inner objects are fully and independently duplicated.

(B) Only the outer list is duplicated into a new object; the inner objects are still shared references with the original.

(C) Neither the outer list nor the inner objects are duplicated — it simply creates an alias (another name for the same object).

(D) The inner objects are independently duplicated, but the outer list remains the same object as the original.

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) Method overriding

(b) Polymorphism

(c) Circularly linked list

(d) Dictionary of Keys (DOK) format for sparse matrices

(14-1) Evaluate the time complexity of the following code segment in terms of Big-O notation:

```python
x = 0
for i in range(n):
    for j in range(n):
        for k in range(10):
            x += 1

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

(14-2) Evaluate the time complexity of the following code segment in terms of Big-O notation:

```java
int x = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        for (int k = 0; k < 10; k++) {
            x++;
        }
    }
}

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

(15) Consider the conversion of the infix expression `(A + B) * (C - D) / E` to its postfix form using a stack, as described in class.

(a) What is the resulting postfix expression?

(b) What is the maximum number of symbols that can be present in the stack at any one moment during the conversion?

(16) Assuming that we have a stack `S` that implements the operations defined in the Stack ADT. Trace through the following operations step by step and answer the questions below.

```
S = Stack()
S.push(3)
S.push(7)
S.push(1)
S.pop()
S.push(4)
S.push(9)
S.pop()
S.push(2)
```

(a) What are the final contents of the stack, listed from bottom to top?

(b) What value does `S.peek()` return after all operations are complete?

(17) Suppose that `B[2][1]` is located at address 36, `B[3][4]` is located at address 54, and `B[0][3]` is located at address 16. What is the address of `B[4][2]`? Is the storage order row-major or column-major? What is the size of each element?