


True or False

For each question, simply answer T (True) or F (False). No explanation is required. (2 points for each question)

1. An important improvement of C++ from C is protecting data with encapsulation. The "private" and "protected" instance variables can only be accessed by instance methods of the same class. *inheritance*
2. In C, dynamic memory allocation mainly relies on "malloc" or "calloc". The allocated memory has to be manually released using "free()", otherwise memory leaks happen. C++ improves the memory operations by using the keyword "new," which makes it easier to allocate memory and memory is released when the pointer is set to "nullptr".
3. When programming a class in C++, it is recommended to implement a constructor and a destructor by yourself. However, it is not mandatory.
4. Operator overloading is a technique to simplify your code by defining how operators behave. By doing so, the program is easier to read and maintain. 
5. A sequential array is a type of data structure that has $O(1)$ time complexity of getting the i -th element, amortized $O(1)$ time complexity for inserting an element *at the end* and $O(1)$ time complexity for deleting the i -th element. However, its downside is the need to allocate continuous memory space in C++.
6. To improve the space efficiency of storing a matrix M , we may always use an array of $[r, c, v]$ to represent M , where $M[r][c] = v$.
7. A stack is a data structure that follows LIFO (Last-in, First-out) order, whereas a queue is a data structure that follows FIFO (First-in, First-out) order.
8. Assume Q is a circular queue implemented in C++ using an integer array. The two variables "front" and "rear" representing the front and rear element indexes are integers. If we have unlimited memory to allocate, then we could increase the capacity of Q indefinitely.
9. By using postfix notation, parentheses can be omitted.
10. Inserting a new node into an arbitrary position of a singly linked list has $O(N)$ time complexity.

Basic Questions

For each question, please give your answer with a short but concise explanation. *Answers without explanations will get 0 points.* (6 points for each question)

11. Explain the core concepts of encapsulation, abstraction, inheritance, and polymorphism. For each concept, give an example of how and why it can be helpful in programming.
12. Please explain what a sparse matrix is. How would you design a method to store a sparse matrix with the least storage space used? Under what circumstances does your design use less storage space?
13. When designing queues, we often implement queues as "circular queues" to avoid wasting memory for popped elements. In our slides, we provided a pseudo code of the queue **Push** operation, as shown below:

```

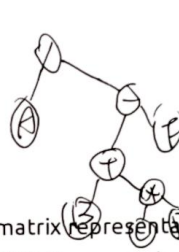
template < class T >
void Queue< T >::Push (const T& x)
{
    // Add x at rear of queue
    if((rear+1)%capacity == front)
    {
        // queue is going to full, double the capacity!
    }
    rear = (rear+1)%capacity;
    queue [rear] = x;
}

```

Why do we **double the capacity** when the queue is full? Why is it more efficient than increasing the capacity by 1?

14. Given an $M \times N$ sparse matrix

$\begin{bmatrix} 1 & 0 & 0 & 9 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 8 \\ 0 & 6 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 5 & 11 & 0 & 0 & 0 & 13 \end{bmatrix}$



ABCD*+*-/

const + n cycle
const + (n+1) cycle

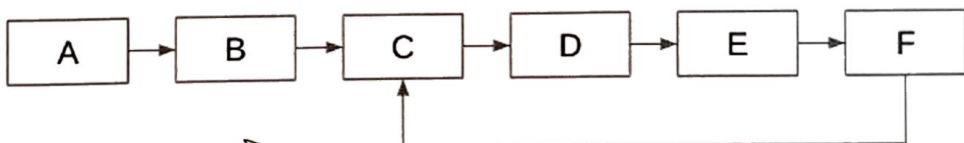
Please represent it using a linked-list sparse matrix representation. The representation should be able to support the iteration of a column in $O(M)$ time complexity and the iteration of a row in $O(N)$ time complexity.

15. Given an infix representation $(A / ((B + (C * D)) - F))$, transfer it into a postfix representation using a stack. Please show your work step-by-step.

Application Questions

For each question, please give your answer with a short but concise explanation. Answers without explanations will get 0 points. (10 points for each question)

16. Given a postfix representation, assume it contains M operands and N binary operators. Prove that the relationship $M = N + 1$ holds.
17. How to find if a list has a cycle? Can you do it in $O(N)$ time complexity and less than or equal to $O(N)$ extra space? Note that a cycle does not necessarily connect the tail to the head, the cycle can happen at any node in the list. An example of a list with a cycle is shown in the following figure:



Handwritten notes: $O(N)$ and $O(1)$ with arrows pointing to the cycle and the list respectively.

18. Circular queues are considered more efficient since popped elements don't occupy storage space. How to implement a circular queue without the use of arrays?
19. Assume you only have queues. How could you utilize these queues to design a data structure that performs like a stack (i.e., supports insertion and deletion following LIFO order)?
20. Now assume you only have stacks. How could you utilize these stacks to design a data structure that performs like a queue (i.e., supports insertion and deletion following FIFO order)?

Follow-Up Questions for Bonus Points

prefix \rightarrow
AB



2023-10-24
void Print(Node* root) {
if (root) Print(root->left);
}

For each question, please give your answer with a short but concise explanation. Answers without explanations will get 0 points. (10 bonus points for each question)

21. Following question 17 in the Application Questions, how to check if a linked list contains a cycle using only $O(1)$ extra space and $O(N)$ time complexity?
22. We have introduced infix and postfix representations in our course. There is another representation method called "prefix," where all operators are placed in front of the operands. For example, $-/+ABCD$ in the prefix is equivalent to $AB+C/D$ in the postfix. Please design a method to transform a prefix representation into its corresponding postfix representation. For simplicity, assume the prefix representation only contains binary operators and operands. Your explanation should include the data structures you use.
23. Assume you can only have one queue, how could you utilize this queue and $O(1)$ extra space to design a data structure that performs like a stack (i.e., supports insertion and deletion following LIFO order)?

Challenging Bonus Question

For each question, please give your answer with a short but concise explanation. Answers without explanations will get 0 points. (20 bonus points)

24. In real-world applications, we often need to know if an element **is** or **is not** in a data list. For example, immigration agencies might want to know efficiently if an applicant **is not** on the wanted list. Another example is checking if an event or log **is** recorded on a blockchain block. Suppose we have N elements; each element is an array of M 64-bit integers. We want to design a data structure that could store the elements. This data structure must meet the following requirements:
 1. Stores all the N elements in less than or equal to $O(M+N)$ storage space
 2. Supports inserting new elements in $O(M)$ time complexity
 3. Given an array of M 64-bit integers, check if this array (element) **is** probably included in the N elements in $O(M)$ time complexity. We call this method `.isProbablyIncluded()`
 - Note that we are not required to support data deletion, nor checking if an element **is not** included.
 - Note that `.isProbablyIncluded(element)` should give true / false Boolean result. If **true** is returned, then element must have a non-zero probability of being included in the data structure. When $N \ll 2^{64}$, the probability should be near 1. If **false** is returned, then *element* should have zero probability of being included in the data structure.