Exam 2 Review Questions.
Created by Kyle Parker, Spring 2025

1. Member functions can access a pointer to themselves called what? What is the purpose of this pointer, and can you provide a use case for it?

2. What data structure would be best suited for implementing a reverse operation easily?

3. We want to create an application like MS Word or Google Docs. What data structure should we utilize for an undo/redo operation?

4. Suppose we have a attribute defined as: `std::fstream myStream`. Write code for the following prompts (1-4 lines).
    a. Open a file named "myData.dat" for reading.
    b. Open a file named "server-dump.log" for writing.
    c. Close the file.
    d. Verify the file was successfully opened.

5. Define abstraction, one of the four pillars in object-oriented programming.

6. Define encapsulation, one of the four pillars in object-oriented programming.

7. Define the following data structures: queue, stack, binary search tree (BST), and singly linked list. For each data structure, provide specific examples of real-world applications.

8. Write a recursive `insert(const std::string& newData)` member function for the BST class. **The function should return true if the new data was successfully inserted and false otherwise.** Assume that the BST uses dynamically linked nodes, where each node contains a string data member and pointers to its left and right children. You may define a private helper function with different parameters to facilitate the recursion; however, the public version must remain as specified above.

    Important note: All variables preceded with an underscore (_) denote a private variable; assume getters exist for Node.
    Class BST has `Node * _root`.

Class Node has `Node* _left`, `Node* _right`, `string _data`.

9. Write a recursive `destroyTree()` member function for the BST class which reclusively destroys the current tree. This function should *recursively* deallocate all nodes in the current tree to free up memory. You may define a private helper function with different parameters to facilitate the recursion.

   Important note: All variables preceded with an underscore (_) denote a private variable; assume getters exist for Node.
   Class BST has `Node * _root`.
   Class Node has `Node* _left`, `Node* _right`, `string _data`.

10. Create a structure which will hold a GPS point, title, an annotation, and a unique id. Here are the data types:
    GPS point: CLLocation
    Title: std::string
    Annotation: std::string ***CONSTRAINT: Must be less than 20 chars***
    Unique ID: UUID

    Create declarations of an explicit constructor and required attributes. We want all attributes to be non-mutable.
    Do not use default values. Assume required include statements are present.

```
struct PinPoint {
        public: // All operations should be public.




        private: // All attributes should be private.




}
```

// Place your declaration for an output stream operator. Display the data in JSON format. Assume the output stream operator is already overloaded for UUID and CLLocation. Sample:

```
{
  uuid: 5B3D1A2E-3F4C-4B5D-8A6E-7B8C9D0E1F2A,
  title: "My pinpoint",
  annotation: "My pinpoint's annotation",
  point: { lat: 38.8409, long: - 105.0422 }
}
```

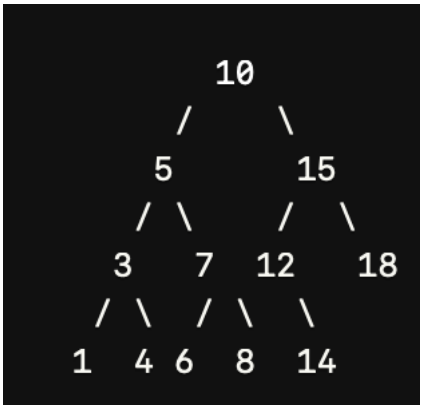
a. Define the constructor declared for PinPoint.
b. Define the output stream operator. **You CANNOT use the friend operator. Write down any assumptions made.**

11. Write a function which accepts a queue and sum all values within. You **only** have access to `enqueue`, `dequeue`, and `isEmpty`. Ensure you restore the queue to its original state (that is it holds the same values in the same order).

12. Write a function which accepts a std::string and converts it to an integer. This should work *similar* to stoi. However, only 10 digits are permitted (assume max value is 9,999,999,999.) It is valid to have a preceding + to indicate a positive value. Otherwise, only digits are permitted. Further, negative values are considered illegal. Return -1 for invalid inputs.

    Samples:
    Input: "09876543567897654356789" → Result: -1
    Input: "12452" → Result: 12452
    Input: "+24" → Result: 24
    Input: "2i9" → Result: -1
    Input: "-24" → Result: -1

13. Answer the following questions about this BST.
    a. Is this a proper BST?
    b. What is a valid sequence of insertions that would result in the same binary search tree (BST) structure?
    c. What is the height of this tree?
    d. What is the depth of this tree?
    e. What is the height of node 12?
    f. What is the depth of node 5?



14. **True/False**: It is possible to change the precedence of the addition operator (+) to be higher than that of the multiplication operator (*).

15. **True/False**: A template function always has identical behavior to overloaded functions.

16. Write a template function which accepts two parameters to mock the behavior of an overloaded function, add. Assume the first parameter type is the return type as shown below with overloaded operators.
    double add(double a, int b)
    int add(int a, int b)
    float add(float a, int b)