1. Using fsu::List L, insert the characters [letters in alpha order] so that a standard traversal of L spells [word]. For example insert a,a,b,c,g,k,m,m,n,o [in this order] to spell "backgammon". [Practice with flist.x]

2. What is a minimal set of operations in the API of X that is not found in Y or Z? [X,Y,Z = permutation of List, Vector, Deque] [Read notes and source code API]

3. Declare an X-based Stack/Queue of T objects. [X=List, Vector, or Deque; T = some type] Note: Be sure you know what the default cases are. [Notes, source code]

4. Assume that d is an fsu::Deque and that the following is a result of d.Dump():

```plaintext
content_[i]:
i mod 10:
   b   e
```

What is the result of the call d.Display() ? [Practice with fdeque.x]

5. Assume that d is an fsu::Deque and that the following is a result of d.Dump():

```plaintext
content_[i]:
i mod 10:
   b   e
```

What is the result of d.Dump() after the operation d.[some operation]? [Practice with fdeque.x]

6. Given the tree (show picture of tree), show the contents of the control stack S during a depth-first search: [Notes; practice with more complicated trees]

Example:
```
       (R)
        /    ""
       (A)   (B)
```

```
S  -->
--|--|
R  RA
R   R
R   -
```
7. Given the tree (show picture of tree), show the contents of the control queue Q during a breadth-first search: [Notes; practice with more complicated trees]

Example:

```
          (R)
         / \
        (A)  (B)
```

Q <-
------
R
AB
A
B

8. What operators are required for a (xxx) iterator? [xxx = forward, bidirectional, random-access] [Notes]

9. What is the worst-case/average-case asymptotic runtime of: [Notes, Assignments]
   - List, Vector, Deque operations [PushBack, etc]
   - OList, OVector operations [Insert, Includes, Retrieve, Remove]
   - BST operations [Insert, Includes, Retrieve, Erase]
   - BST::Iterator operations [++]
   - LLRBT operations [Insert, Includes, Retrieve, Erase]
   - HashTable operations [Insert, Includes, Retrieve, Remove]
   - HashTable::Iterator operations [++]
   - g_lower_bound, g_upper_bound
   - g_min_element, g_max_element
   - g_push_heap, g_pop_heap

10. Analysis of BST::Iterator::operator++ [using edge moves] [Notes]

11. Order-properties of traversals (Pre, In, Post, Level) [Notes]

12. Is this tree (BST) (AVL) (RBT) (LLRBT) [picture] [Notes]

Example:

```
2
/ \
1 3
```

Answer: yes to all

13. Given a tree: Draw the mutated tree after LLRBT::Insert(t) [Notes, frbltt.x]

14. Properties of fsu::Pair, fsu::Entry [Notes]

15. What is the return value of g_lower_bound/g_upper_bound with certain arguments? [Notes; Practice with binary_search_trace.x or fgss.x]

Example: ABCDEFG <- value
          01234567 <- index

```
g_lower_bound(0,7,D) returns 3
```

16. Given a recursive function, what does it compute? [Assignments, Textbook]

17. Properties of ADT Stack, Queue, PriorityQueue [Notes]

18. Apply ga to array, container. [Notes, source code]

Example: copy array A to vector v
          g_copy (A, A+size, v.Begin())

19. Given a vector v, show it after g_push_heap or g_pop_heap is called [Notes]
20. This question is about internal operation/implementation of fsu::HashTable. Suppose you have a hash
table myTable as follows: KeyType is K = fsu::String, DataType is D = int, and HashType is H =
hashclass::Simple. The implementation of BucketType is fsu::List < fsu::Entry >, it uses
BucketType::PushBack(entry) for table insert, and there are five (5) buckets. Recall that the Simple hash
function just adds the character values of the string, starting with a = 0.

Assume that myTable is the hash table object defined above and that the following is the result of the
call myTable.Dump():

[a depiction of Dump]

What is the result when the following traversal code is executed?

```cpp
for (Iterator i = myTable.Begin(); i != myTable.End(); ++i)
    std::cout << *i << ' ';
std::cout << '\n';
```

[Practice with fhtbl_Simple.x]

21. This question is about internal operation/implementation of fsu::HashTable. Suppose you have a hash
table myTable as follows: KeyType is K = fsu::String, DataType is D = int, and HashType is H =
hashclass::Simple. The implementation of BucketType is fsu::List < fsu::Entry >, it uses the
BucketType::PushBack(entry) for table insert, and there are five (5) buckets. Recall that the Simple hash
function just adds the character values of the string, starting with a = 0.

What is the hash value and bucket number for the key [xxxx]? [Practice with hashcalc.x]

22. This question is about internal operation/implementation of fsu::HashTable. Suppose you have a hash
table myTable as follows: KeyType is K = fsu::String, DataType is D = int, and HashType is H =
hashclass::Simple. The implementation of BucketType is fsu::List < fsu::Entry >, it uses
BucketType::PushBack(entry) for table insert, and there are five (5) buckets. Recall that the Simple hash
function just adds the character values of the string, starting with a = 0.

Assume that myTable is the hash table object defined above and that the following code is executed:

```cpp
myTable.Clear();
... (some table operations)
myTable.Dump();
```

What is the result of the call myTable.Dump()? [Practice with fhtbl_Simple.x]