ECE551
PRACTICE Final

This is a full length practice midterm exam. If you want to take it at exam pace, give yourself 175 minutes to take the entire test. Just like the real exam, each question has a point value. There are 115 points from 8 question, so pace yourself accordingly.

Questions:

1. Multiple Choice: 10 pts
2. Concurrency: 10 pts
3. Data Structure Concepts: 12 pts
4. OO Implementation: 15 pts
5. Coding Pictionary: 12 pts
6. Coding 1: 12 pts
7. Coding 2: 22 pts
8. Coding 3: 22 pts

Question 1 Multiple Choice [10 pts]

1. Which data structure is most efficient to use for implementing a priority queue?
   (a) Linked List
   (b) Array
   (c) Skip List
   (d) Heap
   (e) Hash Table
2. What principle is key to building large pieces of software?
   (a) Locality
   (b) Reciprocation
   (c) Abstraction
   (d) Redundancy
   (e) None of the Above
3. Which of the following accurately describes abstraction?
   (a) Pointers to sub-classes can be treated as pointers to their super-class.
   (b) Separation of interface from implementation.
   (c) Method calls are always dispatched to the implementation in the dynamic type of the object.
   (d) Data is often re-referenced again soon after it has been referenced.
   (e) None of the Above

4. What is the main dis-advantage of heap sort?
   (a) Its worst case running time is $O(N^2)$
   (b) Its average case running time is $O(N^2)$
   (c) It requires allocating extra space for temporary arrays
   (d) It requires very complex operations at each step which take a long time.
   (e) None of the Above
5. Which algorithmic category best describes Dijkstra's shortest path algorithm?
   (a) Dynamic Programming
   (b) Brute Force
   (c) Genetic
   (d) Greedy
   (e) None of the Above

Question 2 Concurrency [10 pts]

1. Briefly explain the concept of a "critical section."

2. Briefly explain why a critical section cannot be protected by simply doing something like this:

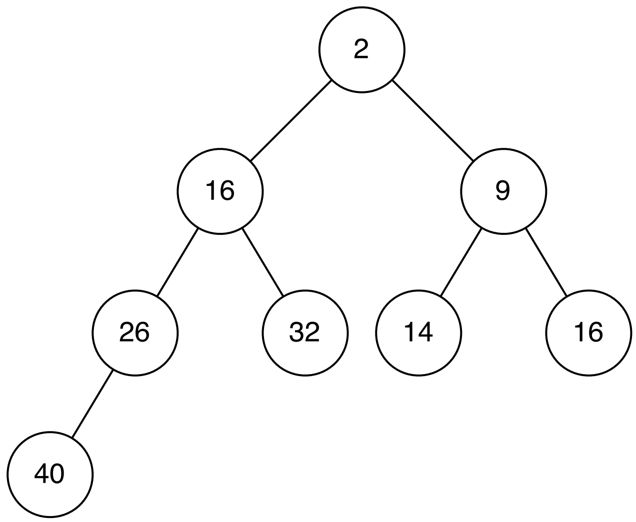
   ```
   while(locked) { ; }
   locked = 1;
   //critical section
   locked = 0;
   ```

3. Give an example of one of the primitives that can be used to build a lock which correctly protects a critical section.

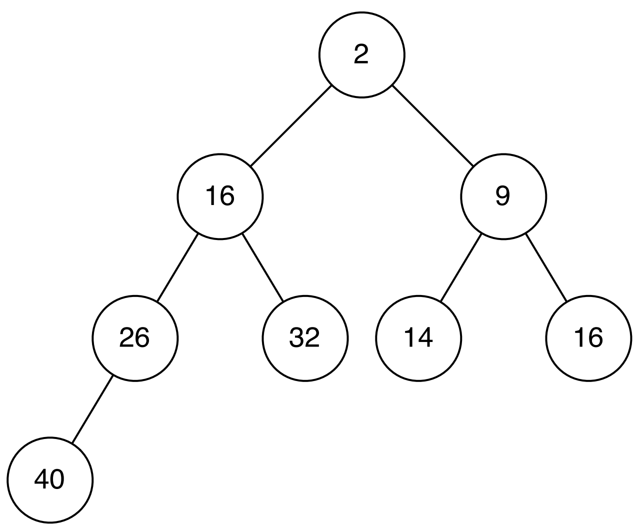
Question 3 Data Structure Concepts [12 pts]

Show the results of performing each of the following operations on the shown data structures:

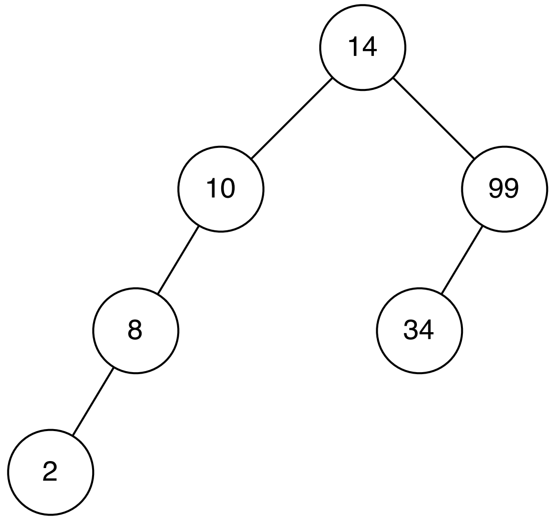
1. Add 12 to the following min-heap



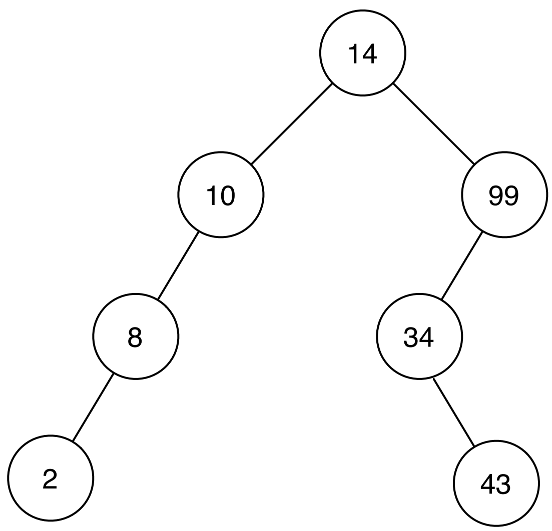
2. Remove the minimum element from the following min-heap:



3. Add 43 to the following (regular, un-balanced) BST



4. Remove 14 from the BST in the previous part (before you added 43).



Question 4 OO Implementation [15 pts]

For parts 1–5 use the following class declarations:

```
class A {
  int a;
  virtual void foo();
};
class B : public A {
  int b;
  virtual void bar();
}
class C: public A {
 int c;
 virtual void foo();
};
class D: public B, public C{
  int d;
  virtual void xyz();
}
```

1. Draw the layout of objects of type `A`

2. Draw the layout of objects of type `B`

3. Draw the layout of objects of type `D`

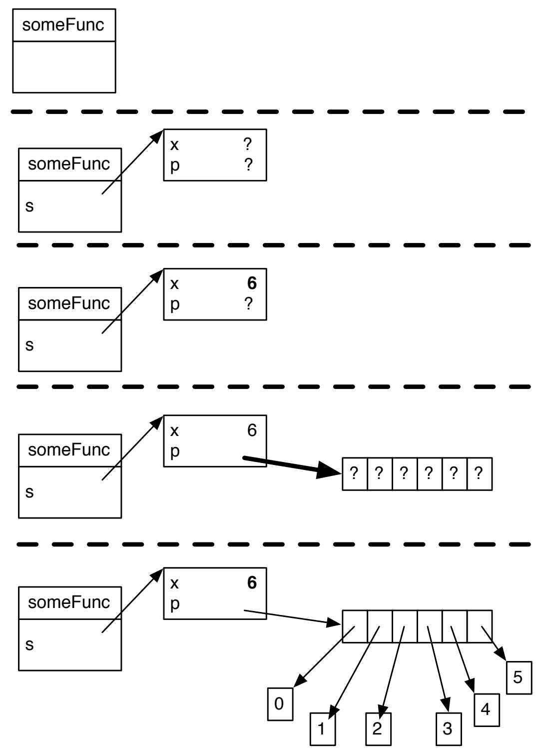
4. Suppose you wanted objects of type `D` to have only one `A` instead of two. Show how you would change the above declarations to do this.

5. Draw the layout of objects of type `D` with the change you made in part 4.

Question 5 Coding Pictionary [12 pts]

The figure below on the right depicts the state of `someFunc` at five points in time. Each dotted horizontal line separates one time from the next. You should write `someFunc` (on the left) by writing the code that goes with the four dotted lines—that is, your `someFunc`'s execution should look like the pictures on the right:

```
struct _astruct {
   int x;
   int ** p;
};
typedef struct _astruct astruct;

astruct * someFunc(void) {
//code for the first dotted line



//code for the second dotted line




//code for the third dotted line




//code for the fourth dotted line









 return s;
}
```



Question 6 Coding 1 [12 pts]

Write the `reverse` method in the `LinkedList` class below. This method should reverse the order of the linked list:

```
template<typename T>
class LinkedList {
  private:
    class Node{
       public:
        Node * next;
        T data;
        Node(T _data): next(NULL), data(_data) {}
        Node(T _data, Node * _next): next(_next), data(_data) {}
    };
  Node * head;
public:
  void reverse() {









  }
};
```

Question 7 Coding 2 [22 pts]

Suppose you have already written a templated `Set` class, with the following interface:

```
template<class T>
class Set {
public:
  Set();
  void add(const T& item);
  bool contains(const T& item) const;
  void remove(const T& item);
  class iterator {
    iterator & operator++();
    T& operator*();
    bool operator==(const iterator & rhs);
    bool operator!=(const iterator & rhs);
  };
  iterator begin() const;
  iterator end() const;
};
```

and you also have the following abstract `Function` class:

```
template<class R, class A>
class Function {
public:
  virtual R invoke(A arg) =0;
};
```

Write a function which filters a `Set`, creating a new `Set` which is a subset of the first set, containing only those items for which the `Function` (`f`) it is passed returns `true`:

(answer on the next page)

```
template<class T>
Set<T> * filterSet(Set<T> * inSet, Function<bool,const T&> * f){

}
```

Question 8 Coding 3 [22 pts]

Suppose you have the following `BinaryTree` class:

```
template<typename T>
class BinaryTree {
  private:
    class Node {
    public:
      T data;
      Node * left;
      Node * right;
    };
    Node * root;
  const T& minInTree() {
    //already implemented, not shown
  }
  const T& maxInTree() {
    //already implemented, not shown
  }
 public:
  //constructors, destructors, other methods not shown
  bool isBSTOrdered() {
    //you will write this
  }
};
```

You must write the `isBSTOrdered` method which determines if the `BinaryTree` obeys the BST ordering. You may find the `minInTree` and `maxInTree` methods useful to do this, which determine the smallest and largest value in the tree respectively. These methods may only be called if the tree is non-empty (else they will throw an exception). You may write any helper methods you wish to (and are encouraged to write at least one).

(answer on the next page)

```
template<typename T>
class BinaryTree {
 //everything else ommitted to give you space to write









   bool isBSTOrdered() {












   }
};
```