Please answer the following questions, each of which is worth 20 points.

1. This question deals with object-oriented design principles.

(a) Define the term *inheritance*.

(b) Suppose you are given a concrete Priority Queue class, MyQueue, which implements the following interface:

```java
public interface PriorityQueue {
    public int size();
    public boolean isEmpty();
    public void insertItem(Object key, Object element)
        throws InvalidKeyException;
    public Object minElement() throws EmptyContainerException;
    public Object minKey() throws EmptyContainerException;
    public Object removeMinElement() throws EmptyContainerException;
}
```

Write an adapter class, StudentPQ, in Java, that uses the MyQueue class to define a priority queue restricted to deal only with keys that are Integer objects and elements that are Student objects.

2. This question deals with stacks and algorithm analysis.

(a) Define what it means for a function \( f(n) \) to be \( \Omega(g(n)) \).

(b) Consider the following algorithm described in pseudo-code, which takes a reference \( S \) to a stack object as input:

```
Let \( n \leftarrow S.size() \)
for \( i \leftarrow 1 \) to \( n - 1 \) do
    Let \( d \leftarrow S.pop() \)
    Let \( T \) be a reference to a new initially-empty stack.
    while \( S \) is not empty do
        Let \( e \leftarrow S.pop() \)
        if \( d = e \) then
            Print “A duplicate has been found.”
        end if
        T.push(e)
    end while
Assign \( S \leftarrow T \)
end for
```

Characterize using the big-Oh notation the running time of the above method in terms of \( n \), the number of elements originally in \( S \).

(c) Consider again the above pseudo code. Suppose \( S \) initially refers to the stack denoted \((1, 2, 3, 4, 5, 6, 7, 8, 9)\), with 1 being at the top and 9 being at the bottom. What is the stack that \( S \) refers to after the pseudo code is complete?
3. This question deals with sequences.
   (a) What is meant by the term rank of an element in a sequence?
   (b) List five (5) methods in the general Sequence ADT.

4. This question deals with heaps and priority queues.
   (a) Define the two properties that makes a binary tree be a heap.
   (b) Draw an example of a heap containing 11 elements with their keys taken uniquely from
       the set of integer objects \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}.
   (c) Draw a sequence of snapshots of the state of this heap that occur when an element with
       key 1 is inserted into the heap.

5. This question deals with AVL trees.
   (a) What is the property that makes a binary search tree be an AVL tree?
   (b) Consider the following AVL tree:

       Show the AVL tree that results from the insertion of an item with key 26.
   (c) Using again the above AVL tree, show the AVL tree that results from the removal of the
       item with key 6.

6. This question deals with hashing.
   (a) Define the term collision in the context of hashing.
   (b) Draw a representation of a 7-celled hash table \(B\) and its contents after we use the
       mapping
       \[ h(x) = 3x + 2 \mod 7 \]
       to insert the elements in the set \{10, 11, 3, 5, 4, 7, 16, 22\} into \(B\), assuming we handle
       collisions with the chaining method.

7. This question deals with sorting.
   (a) Let \(S = \{4, 2, 8, 5, 9, 1, 3, 7\}\) be a sequence of 8 integer objects. Draw the recursion tree
       for a running of Quick-Sort on \(S\), assuming we always use the last element in a sequence
       as the pivot. Show the input sequence for each recursive call.
   (b) What is the worst-case running time of this version of the Quick-Sort algorithm in terms
       of \(n\), the size of \(S\)? Briefly describe a general example of an input sequence that will
       cause this version of quicksort to exhibit its worst-case time bound.
8. This question deals with unweighted graphs.

(a) Define the term degree of a vertex.
(b) Give five (5) methods of the graph ADT.
(c) Draw a picture that represents the adjacency matrix of the following graph:

![Graph Image]

9. Show each snapshot of a running of Dijkstra’s shortest path algorithm on the following weighted graph, starting at the vertex a. Darken in each edge of the (final) shortest path tree from vertex a.

![Graph Image]

10. Suppose you are given a general sequence, S, containing Integer objects as its elements. Briefly describe an efficient method for testing if S contains any duplicate elements. (Hint: don’t use an algorithm as naive as that used in Problem 2 above.) What is the running time of your method?