Version 1

1. (16 Points) Multiple Choice:

A. (2 Points) Which of the following loop headers will arrange for the loop body to execute exactly 10 times?
```
a. for (int i = 1  ; i < 10  ; ++i)
b. for (int i = 0  ; i <= 10 ; ++i)
c. for (int i = -5 ; i < 5   ; ++i)
d. for (int i = 2  ; i < 20  ; ++i)
```

B. (2 Points) An instance of a class is known as a(n) _____.
```
a. module
b. object
c. abstract data type
d. data structure
```

C. (2 Points) A class method is defined as _____.
```
a. static
b. abstract
c. private
d. protected
```

D. (2 Points) The insertion operation of the ADT list can insert new items _____.
```
a. only at the front of the list
b. only at the end of the list
c. only in the middle of the list
d. into any position of the list
```

E. (2 Points) Which of the following will be true when the reference variable curr references the last node in a linear linked list?
```
a. curr == null
b. head == null
c. curr.getNext() == null
d. head.getNext() == null
```

F. (2 Points) In a grammar, the symbol x | y means _____.
```
a. x or y
b. x followed by y
c. x out of y
d. x divided by y
```

G. (2 Points) If the array: {6, 2, 7, 13, 5, 4}is added to a stack, in the order given, which number will be the first number to be removed from the stack?
```
a. 6
b. 2
c. 5
d. 4
```

H. (2 Points) Which of the following is the code to insert a new node, referenced by `newNode`, into an empty queue represented by a circular linked list?
```
a. newNode.setNext(lastNode);
b. lastNode.setNext(lastNode);
   lastNode = newNode;
c. newNode.setNext(lastNode);
   newNode = lastNode;
d. newNode.setNext(newNode);
   lastNode = newNode;
```

Version 1

2. (20 Points) Given the following StackInterface:

```
public interface StackInterface {
   public void push(Object obj);
   public Object pop();
   public Object peek();
}
```

The correct array-based implementation is:

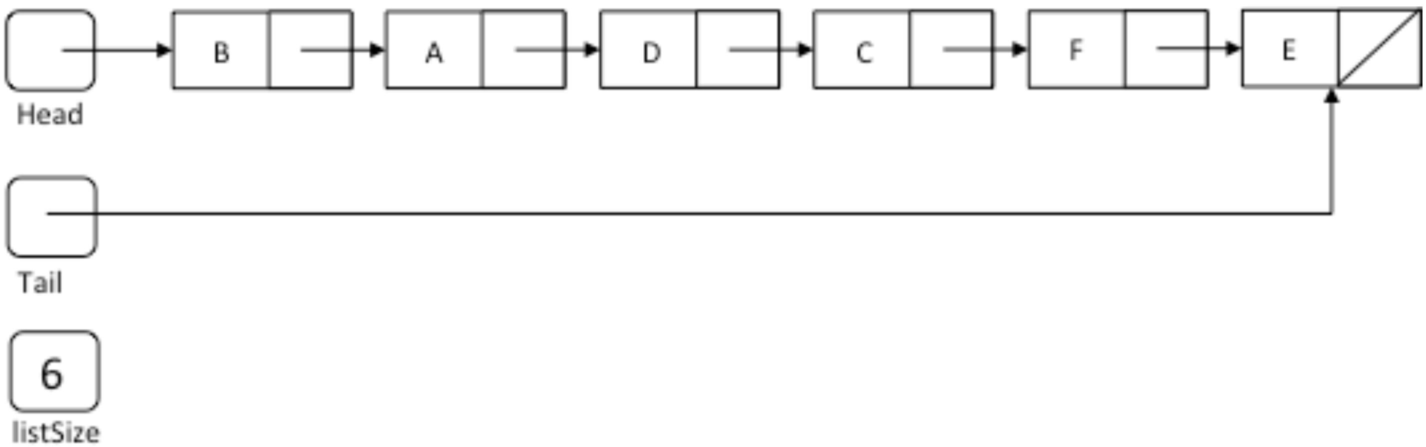
```
import java.util.Vector;

public class ArrayStack implements StackInterface {

   private Vector<Object> stackVector = new Vector<Object>();

   private final int INVALID_STACK_POINTER = -1;
   private int stackPointer = INVALID_STACK_POINTER;

   @Override
   public void push(Object obj) {
      stackVector.add(++stackPointer, obj);
   }

   @Override
   public Object pop() {
      Object obj = null;
      if (stackPointer != INVALID_STACK_POINTER) {
         obj = stackVector.elementAt(stackPointer);
         stackVector.removeElementAt(stackPointer--);
      }
      return obj;
   }

   @Override
   public Object peek() {
      Object obj = null;
      if (stackPointer != INVALID_STACK_POINTER) {
         obj = stackVector.elementAt(stackPointer);
      }
      return obj;
   }
}
```

Version 1

3. (50 Points) The correct LinkedQueue implementation is:

```java
import java.util.Vector;

public class LinkedQueue implements QueueInterface {

    private Node front = null, back = null;
    private int size = 0;

    @Override
    public boolean isEmpty() {
       return (front == null);
    }

    @Override
    public int size() {
       return this.size;
    }

    @Override
    public void add(Object obj) {
       Node newNode = new Node(obj);
       if (back == null) {
          front = newNode;
       } else {
          back.setNext(newNode);
       }
       back = newNode;
       this.size++;
    }

    @Override
    public Object remove() {
       Object obj = null;
       if (front != null) {
          obj = front.getObject();
          front = front.getNext();
          this.size--;
       }
       if (front == null) {
          back = null;
       }
       return obj;
    }

    @Override
    public boolean equals(Object oQueue) {
       boolean answer = false;
       LinkedQueue otherQueue;

       if (oQueue instanceof LinkedQueue) {
          otherQueue = (LinkedQueue) oQueue;
       } else {
          return answer;
       }

       Vector<Object> myPV = this.peekAll();
       answer = myPV.equals(otherQueue.peekAll());

       return answer;
    }

    @Override
    public Vector<Object> peekAll() {
       Vector<Object> pv = new Vector<Object>();
       Node curNode = this.front;

       while (curNode != null) {
          pv.add(curNode.getObject());
          curNode = curNode.getNext();
       }

       return pv;
    }
}
```

4. (20 Points) The list after `doStuff1()` has finished executing:



Version 2

1. (16 Points) Multiple Choice:

A. (2 Points) Which of these expressions is illegal in Java?
   a. x++ 5
   b. x =+ 5
   c. x + = 5
   d. x == 5

B. (2 Points) Which of the following is an example of a syntax error?
   a. a program encounters an instruction to divide by zero
   b. an array subscript in a program goes out of range
   c. the beginning of a while loop is written as "whille" instead of "while"
   d. an algorithm that calculates the monthly payment of a loan displays incorrect results

C. (2 Points) The midpoint of a sorted array can be found by _____, where first is the index of the first item in the array and last is the index of the last item in the array.
   a. first / 2 + last / 2
   b. first / 2 – last / 2
   c. (first + last) / 2
   d. (first – last) / 2

D. (2 Points) In the ADT list, when an item is deleted from position i of the list, _____.
   a. the position of all items is decreased by 1
   b. the position of each item that was at a position smaller than i is decreased by 1
   c. the position of each item that was at a position greater than i is decreased by 1
   d. the position of each item that was at a position smaller than i is increased by 1 while the position of each item that was at a position greater than i is decreased by 1

E. (2 Points) Which of the following statements deletes the node that curr references?
   a. prev.setNext(curr);
   b. curr.setNext(prev);
   c. curr.setNext(curr.getNext());
   d. prev.setNext(curr.getNext());

F. (2 Points) In a grammar, the symbol x y means _____.
   a. x or y
   b. x followed by y
   c. x or y or both
   d. x multiplied by y

G. (2 Points) If the array: {6, 21, 35, 3, 6, 2, 13} is added to a stack, in the order given, which of the following is the top of the stack?
   a. 2
   b. 6
   c. 3
   d. 13
   e. 35

H. (2 Points) The _____ operation retrieves the item that was added earliest to a queue, but does not remove that item.
   a. enqueue
   b. dequeue
   c. dequeueAll
   d. peek

2. (20 Points) Given the following StackInterface:

```
public interface StackInterface {
    public void push(Object obj);
    public Object pop();
    public Object peek();
}
```

The correct referenced-based implementation is:

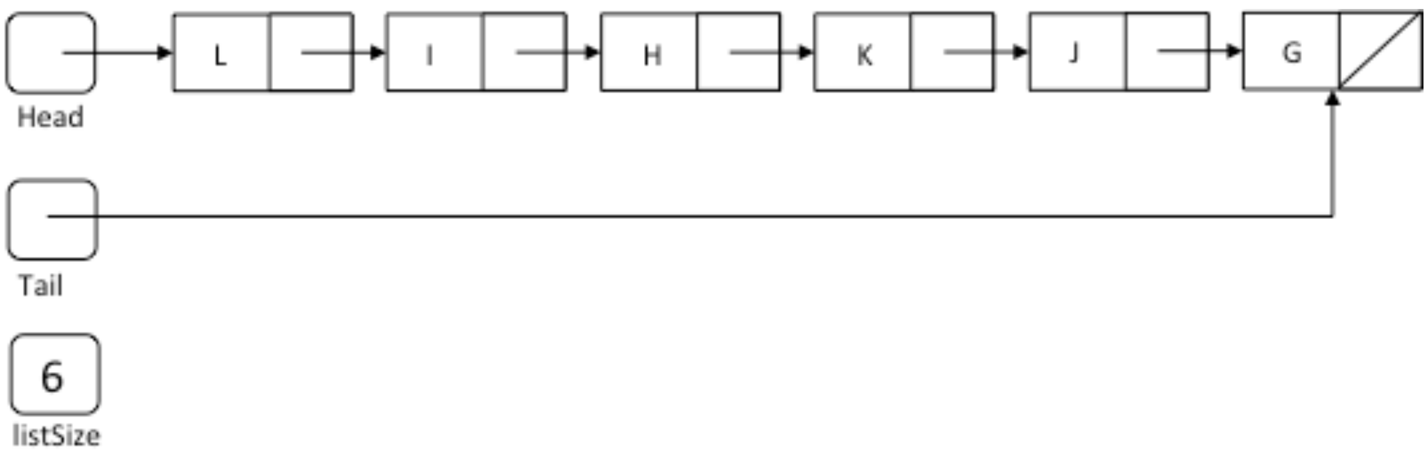
```
public class Node {
    private Object object;
    private Node next;

    public Node() {
        this.object = null;
        this.next = null;
    }

    public Node(Object object) {
        this.object = object;
        this.next = null;
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
    }

    public Object getObject() {
        return object;
    }
}
```

```
public class ReferenceStack implements StackInterface {

    private Node stackPointer = null;

    @Override
    public void push(Object obj) {
        Node newNode = new Node(obj);
        if (stackPointer == null) {
            stackPointer = newNode;
        } else {
            newNode.setNext(stackPointer);
            stackPointer = newNode;
        }
    }

    @Override
    public Object pop() {
        Object obj = null;
        if (stackPointer != null) {
            obj = stackPointer.getObject();
            stackPointer = stackPointer.getNext();
        }
        return obj;
    }

    @Override
    public Object peek() {
        Object obj = null;
        if (stackPointer != null) {
            obj = stackPointer.getObject();
        }
        return obj;
    }
}
```

3. (50 Points) The correct ArrayQueue implementation is:

```
import java.util.Vector;
public class ArrayQueue implements QueueInterface {

	private Vector<Object> queueVector = new Vector<Object>();

	@Override
	public boolean isEmpty() {
		return queueVector.isEmpty();
	}

	@Override
	public int size() {
		return queueVector.size();
	}

	@Override
	public void add(Object obj) {
		queueVector.addElement(obj);
	}

	@Override
	public Object remove() {
		Object obj = null;
		if (queueVector.size() > 0) {
			obj = queueVector.elementAt(0);
			queueVector.remove(0);
		}
		return obj;
	}

	@Override
	public boolean equals(Object oQueue) {
		boolean answer = false;
		ArrayQueue otherQueue;

		if (oQueue instanceof ArrayQueue) {
			otherQueue = (ArrayQueue) oQueue;
		} else {
			return answer;
		}

		answer = queueVector.equals(otherQueue.peekAll());

		return answer;
	}

	@Override
	public Vector<Object> peekAll() {
		return (Vector<Object>) queueVector.clone();
	}
}
```

4. (20 Points) The list after `doStuff2()` has finished executing:



Version 3

1. (16 Points) Multiple Choice:

A. (2 Points) If we wanted to write an if-statement that executes whenever the real number x is between 10.0 and 20.0, how should the test condition be written?
   - a. `10.0 < x || x > 20.0`
   - b. `10.0 < x && x > 20.0`
   - **c. `10.0 < x && x < 20.0`**
   - d. `10.0 < x || x < 20.0`

B. (2 Points) The communication mechanisms among modules are called _____.
   - a. algorithms
   - b. solutions
   - c. prototypes
   - **d. interfaces**

C. (2 Points) In a sorted array, the k^{th} smallest item is given by _____.
   - **a. `anArray[k-1]`**
   - b. `anArray[k]`
   - c. `anArray[SIZE-k]`
   - d. `anArray[SIZE+k]`

D. (2 Points) In the ADT list, when an item is inserted into position i of the list, _____.
   - a. the position of all items is increased by 1
   - b. the position of each item that was at a position smaller than i is increased by 1
   - **c. the position of each item that was at a position greater than i is increased by 1**
   - d. the position of each item that was at a position smaller than i is decreased by 1 while the position of each item that was at a position greater than i is increased by 1

E. (2 Points) Which of the following statements deletes the first node of a linear linked list that has 10 nodes?
   - a. `head.setNext(curr.getNext());`
   - b. `prev.setNext(curr.getNext());`
   - **c. `head = head.getNext();`**
   - d. `head = null;`

F. (2 Points) If the string w is a palindrome, which of the following is true?
   - a. w minus its first character is a palindrome
   - b. w minus its last character is a palindrome
   - **c. w minus its first and last characters is a palindrome**
   - d. the first half of w is a palindrome
   - e. the second half of w is a palindrome

G. (2 Points) If the array: {6, 2, 7, 13, 5, 4} is added to a queue, in the order given, which number will be the first number to be removed from the queue?
   - **a. 6**
   - b. 2
   - c. 5
   - d. 4

H. (2 Points) Operations on a queue can be carried out at _____.
   - a. its front only
   - b. its back only
   - **c. both its front and back**
   - d. any position in the queue

Version 3

2. (20 Points) Given the following QueueInterface:

```
public interface QueueInterface {
   public void add(Object obj);
   public Object remove();
   public Object peek();
}
```

The correct array-based implementation is:

```
import java.util.Vector;

public class ArrayQueue implements QueueInterface {

   private Vector<Object> queueVector = new Vector<Object>();

   @Override
   public void add(Object obj) {
      queueVector.addElement(obj);
   }

   @Override
   public Object remove() {
      Object obj = null;
      if (queueVector.size() ≥ 0) {
         obj = queueVector.elementAt(0);
         queueVector.remove(0);
      }
      return obj;
   }

   @Override
   public Object peek() {
      Object obj = null;
      if (queueVector.size() > 0) {
         obj = queueVector.elementAt(0);
      }
      return obj;
   }
}
```

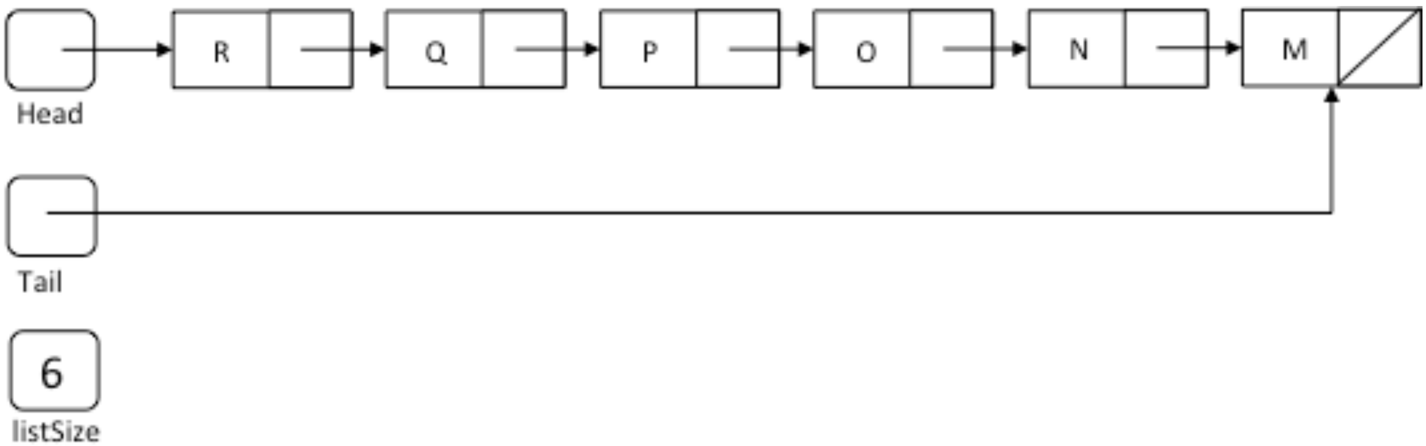
Version 3

3. (50 Points) The correct LinkedStack implementation is:

```
import java.util.Vector;

public class LinkedStack implements StackInterface {

    private Node stackPtr = null;
    int size = 0;

    @Override
    public boolean isEmpty() {
        return (stackPtr == null);
    }

    @Override
    public int size() {
        return this.size;
    }

    @Override
    public void push(Object obj) {
        Node newNode = new Node(obj);
        if (stackPtr == null) {
            stackPtr = newNode;
        } else {
            newNode.setNext(stackPtr);
            stackPtr = newNode;
        }
        this.size++;
    }

    @Override
    public Object pop() {
        Object obj = null;
        if (stackPtr != null) {
            obj = stackPtr.getObject();
            stackPtr = stackPtr.getNext();
        }
        this.size--;
        return obj;
    }

    @Override
    public boolean equals(Object oStack) {
        boolean answer = false;
        LinkedStack otherStack;

        if (oStack instanceof LinkedStack) {
            otherStack = (LinkedStack) oStack;
        } else {
            return answer;
        }

        Vector<Object> myPV = this.peekAll();
        answer = myPV.equals(otherStack.peekAll());

        return answer;
    }

    @Override
    public Vector<Object> peekAll() {
        Vector<Object> pv = new Vector<Object>();
        Node curNode = this.stackPtr;

        while (curNode != null) {
            pv.add(curNode.getObject());
            curNode = curNode.getNext();
        }

        return pv;
    }
}
```

Version 3

4. (20 Points) The list after `doStuff3()` has finished executing:



Version 4

1. (16 Points) Multiple Choice:

A. (2 Points) If s1 is of type String, what does s1.compareTo(s1) return?
   - **a. zero**
   - b. true
   - c. false
   - d. Cannot be determined without knowing the value of s1.

B. (2 Points) Which of the following is an example of a logical error?
   - **a. an algorithm that calculates the monthly payment of a loan displays incorrect results**
   - b. an array subscript in a program goes out of range
   - c. a program expects a nonnegative number but reads –23
   - d. the beginning of a while loop is written as "whille" instead of "while"

C. (2 Points) The factorial of n is equal to _____.
   - a. n – 1
   - b. n – factorial (n–1)
   - c. factorial (n–1)
   - **d. n * factorial (n–1)**

D. (2 Points) In the following list {John, Kate, Fred, Mark, Jon, Adam, Drew} which element does not have a predecessor?
   - **a. John**
   - b. Mark
   - c. Drew
   - d. Kate

E. (2 Points) Which of the following statements is used to insert a new node, referenced by newNode, at the end of a linear linked list?
   - a. newNode.setNext(curr); prev.setNext(newNode);
   - b. newNode.setNext(head); head = newNode;
   - **c. prev.setNext(newNode);**
   - d. prev.setNext(curr); newNode.setNext(curr);

F. (2 Points) The symbol A^nB^n is standard notation for the string that consists of _____.
   - a. an A, followed by an n, followed by a B, followed by an n
   - b. an equal number of A's and B's, arranged in a random order
   - **c. n consecutive A's, followed by n consecutive B's**
   - d. a pair of an A and a B, followed another pair of an A and a B

G. (2 Points) The last-in, first-out (LIFO) property is found in the ADT _____.
   - a. list
   - **b. stack**
   - c. queue
   - d. tree

H. (2 Points) In a queue, items can be added _____.
   - a. only at the front of the queue
   - **b. only at the back of the queue**
   - c. either at the front or at the back of the queue
   - d. at any position in the queue

Version 4

2. (20 Points) Given the following QueueInterface:

```
public interface QueueInterface {
    public void add(Object obj);
    public Object remove();
}
```

The correct referenced-based implementation is:

```
public class Node {
    private Object object;
    private Node next;

    public Node(Object object) {
            this.object = object;
            this.next = null;
    }

    public Node getNext() {
            return next;
    }

    public void setNext(Node next) {
            this.next = next;
    }

    public Object getObject() {
            return object;
    }
}
```

```
public class ReferenceQueue implements QueueInterface {

    private Node front = null, back = null;

    @Override
    public void add(Object obj) {
            Node newNode = new Node(obj);
            if (back == null) {
                    front = newNode;
                    back = front;
            } else {
                    back.setNext(newNode);
                    back = newNode;
            }
    }

    @Override
    public Object remove() {
            Object obj = null;
            if (front != null) {
                    obj = front.getObject();
                    front = front.getNext();
            }
            if (front == null) {
                    back = null;
            }
            return obj;
    }
}
```

Version 4

3. (50 Points) The correct ArrayStack Implementation is:

```
import java.util.Vector;

public class ArrayStack implements StackInterface {

   private Vector<Object> stackVector = new Vector<Object>();

   private final int INVALID_STACK_POINTER = -1;
   private int stackPointer = INVALID_STACK_POINTER;

   @Override
   public boolean isEmpty() {
      return stackVector.isEmpty();
   }

   @Override
   public int size() {
      return stackVector.size();
   }

   @Override
   public void push(Object obj) {
      stackVector.add(++stackPointer, obj);
   }

   @Override
   public Object pop() {
      Object obj = null;
      if (stackPointer != INVALID_STACK_POINTER) {
         obj = stackVector.elementAt(stackPointer);
         stackVector.removeElementAt(stackPointer--);
      }
      return obj;
   }

   @Override
   public boolean equals(Object oStack) {
      boolean answer = false;
      ArrayStack otherStack;

      if (oStack instanceof ArrayStack) {
         otherStack = (ArrayStack) oStack;
      } else {
         return answer;
      }

      answer = stackVector.equals(otherStack.peekAll());
      return answer;
   }

   @Override
   public Vector<Object> peekAll() {
      return (Vector<Object>)stackVector.clone();
   }
}
```

Version 4

4. (20 Points) The list after `doStuff4()` has finished executing: