Version 1

Instructions

- **Write your name and version number on the top of the yellow paper.**
- **Answer all questions on the yellow paper.**
- **One question per page.**
- **Use only one side of the yellow paper.**

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) Which access modifier, used when defining a method, indicates that only one such method is available for all instances of the class?

a. final
b. private
c. protected
d. static

C. (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

D. (2 Points) Data structures are part of an ADT's _____.

a. definition
b. implementation
c. specifications
d. usage

E. (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. Adam

F. (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

G. (2 Points) The last node of a linear linked list _____.

a. has the value null
b. has a next reference whose value is null
c. has a next reference which references the first node of the list
d. cannot store any data

H. (2 Points) Which of the following statements deletes the node that curr references?

I.

a. `prev.setNext(curr);`
b. `curr.setNext(prev);`
c. `curr.setNext(curr.getNext());`
d. `prev.setNext(curr.getNext());`

Version 1

2. (20 Points) Given the following generic MyArray class:

```
public class MyArray<I> {

    private I[] array = (I[]) new Object[100];
    private int currentLocation = 0;

    public void addElement(I element) {
       array[currentLocation++] = element;
    }

    public void replaceElement(I newElement, int index) {
       if ((index >= currentLocation) ||
            ((index < 0) || (index > array.length))) {
          System.out.println("Error");
       }

       array[index] = newElement;
    }

    public void removeElement(int index) {
       if (((index >= currentLocation) && (index < array.length)) ||
            ((index < 0) || (index > array.length))) {
          System.out.println("Error");
       }

       for ( int i = index + 1 ; i < currentLocation ; i++ ) {
          array[i-1] = array[i];
       }
       array[--currentLocation] = null;
    }

    public void clear() {
       for ( int i = 0 ; i < array.length ; i++ ) {
          array[i] = null;
       }
       currentLocation = 0;
    }

    public int numberOfElements() {
       return currentLocation;
    }
}
```

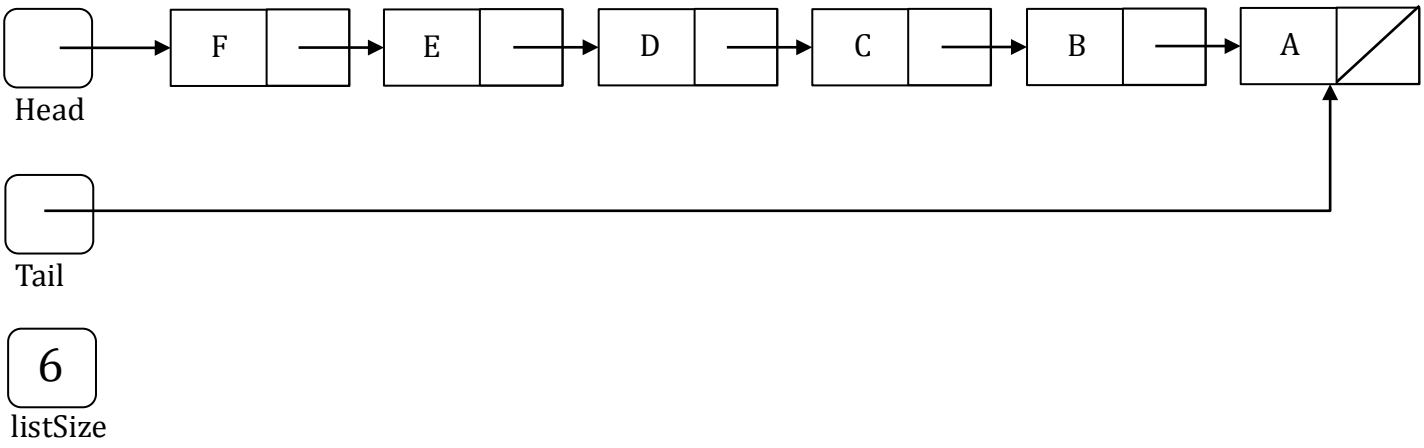
Version 1

3. (40 Points) The correct generic LinkedQueue Implementation is:

```
import java.util.Vector;

public class LinkedQueue<I> implements
                                QueueInterface<I> {

    private Node<I> 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 enqueue(I element) {
       Node<I> newNode = new Node<I>(element);
       if (back == null) {
          front = newNode;
       } else {
          back.setNext(newNode);
       }
       back = newNode;
       this.size++;
    }

    @Override
    public I dequeue() {
       I element = null;
       if (front != null) {x
          element = front.getElement();
          front = front.getNext();
          this.size--;
       }
       if (front == null) {
          back = null;
       }
       return element;
    }

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

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

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

       return answer;
    }

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

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

       return pv;
    }
}
```

Version 1

4. (30 Points) The list after doStuff1() has finished executing:



Version 2

Instructions

- **Write your name and version number on the top of the yellow paper.**
- **Answer all questions on the yellow paper.**
- **One question per page.**
- **Use only one side of the yellow paper.**

1. (16 Points) Multiple Choice:

A. (2 Points) The Java expression 9 / 5 + 9 % 5 equals _____.

a. 0
b. 1
c. 3
d. 5
e. 6

B. (2 Points) Consider the following code that appears in a test class.

```
A a = new A();
int c = a.b;
```

In order for this code to work, which statement must be true?

a. a must be declared public inside class A
b. b must be declared public inside class A
c. c must be declared public inside class A
d. Method b( ) must return int

C. (2 Points) The syntax errors of a program are removed during the _____ phase of the program's life cycle

a. verification
b. coding
c. testing
d. refining
e. maintenance

D. (2 Points) An ADT's _____ govern(s) what its operations are and what they do.

a. specifications
b. implementation
c. documentation
d. data structure

E. (2 Points) In the following list:
`John,Kate, Fred, Mark, Jon, Adam, Drew`
which element does not have a successor?

a. John
b. Mark
c. Drew
d. Adam

F. (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

G. (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

H. (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;

Version 2

2. (20 Points) Given the following generic MyArray class:

```
public class MyArray<I> {

    private I[] array = (I[]) new Object[100];
    private int currentLocation = 0;

    public void addElement(I element) {
       array[currentLocation++] = element;
    }

    public void replaceElement(I newElement, int index) {
       if ((index >= currentLocation) ||
            ((index < 0) || (index > array.length))) {
          System.out.println("Error");
       }

       array[index] = newElement;
    }

    public void removeElement(int index) {
       if (((index >= currentLocation) && (index < array.length)) ||
            ((index < 0) || (index > array.length))) {
          System.out.println("Error");
       }

       for ( int i = index + 1 ; i < currentLocation ; i++ ) {
          array[i-1] = array[i];
       }
       array[--currentLocation] = null;
    }

    public void clear() {
       for ( int i = 0 ; i < array.length ; i++ ) {
          array[i] = null;
       }
       currentLocation = 0;
    }

    public int numberOfElements() {
       return currentLocation;
    }
}
```

Version 2

3. (40 Points) The correct generic LinkedStack implementation is:

```
import java.util.Vector;

public class LinkedStack<I> implements
StackInterface<I> {

    private Node<I> stackPtr = null;
    int size = 0;

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

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

    @Override
    public void push(I element) {
       Node<I> newNode = new Node<I>(element);
       if (stackPtr == null) {
          stackPtr = newNode;
       } else {
          newNode.setNext(stackPtr);
          stackPtr = newNode;
       }
       this.size++;
    }

    @Override
    public I pop() {
       I element = null;
       if (stackPtr != null) {
          element = stackPtr.getElement();
          stackPtr = stackPtr.getNext();
       }
       this.size--;
       return element;
    }

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

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

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

       return answer;
    }

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

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

       return pv;
    }
}
```

Version 2

4. (30 Points) The list after doStuff2() has finished executing: