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) 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

B. (2 Points) A(n) _____ is a Java construct that enables a programmer to define a new data type.

a. class
b. method
c. data field
d. object

C. (2 Points) If you attempt to use a reference variable before it is instantiated, a(n) _____ will be thrown.

a. IndexOutOfBoundsException
b. InstantiationException
c. IllegalAccessException
d. NullPointerException

D. (2 Points) A linked list contains components, called _____, which are linked to one another.

a. nodes
b. arrays
c. vectors
d. references

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) 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 _____ method of the ADT stack retrieves and then removes the top of the stack.

a. createStack
b. push
c. pop
d. peek

H. A superclass method can be accessed by a subclass, even though it has been overridden by the subclass, by using the _____ reference.

a. super
b. final
c. static
d. new

2. (20 Points) Corrected generic MyArray class:

```
public class MyArray<I> {
   private Object[] array = new Object[100];
   private int currentLocation = 0;

   public int size() {
      return currentLocation;
   }

   public boolean isEmpty() {
      return (currentLocation == 0);
   }

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

   public I getElement(int index) {
      I element = null;
      if ((index < currentLocation) && (index >= 0)) {
         element = (I) array[index];
      }
      return element;
   }

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

   public void removeElement(int index) {
      if ((index >= currentLocation) || (index < 0)) {
         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;
   }
}
```

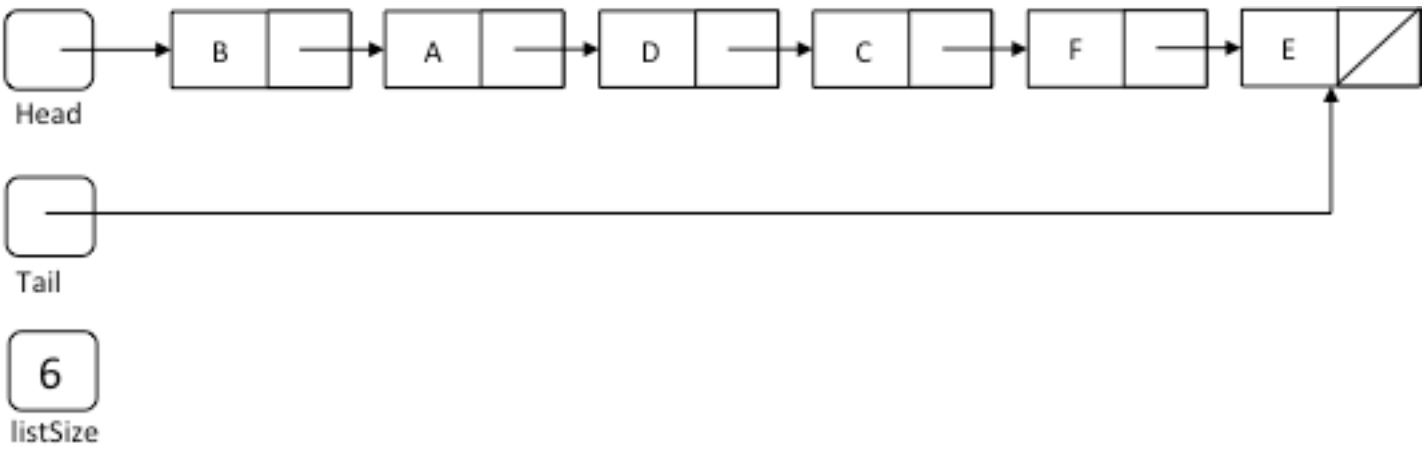
Version 1

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

```
import java.util.Vector;

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

    private Object[] queue = new Object[100];
    private int front = -1;
    private int back = -1;
    private int size = 0;

    @Override
    public boolean isEmpty() {
        return (this.size == 0);
    }

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

    @Override
    public void enqueue(I element) {
        if (this.size == this.queue.length) {
            return;
        } else if (this.isEmpty()) {
            this.front = 0;
            this.back = 0;
            this.queue[front] = element;
        } else {
            this.back = this.increment(this.back);
            this.queue[back] = element;
        }
        this.size++;
    }

    @Override
    public I dequeue() {
        if (this.isEmpty()) {
            return null;
        } else {
            I element = (I) queue[this.front];
            this.front = this.increment(this.front);
            this.size--;
            return element;
        }
    }

    @Override
    public Vector<I> peekAll() {
        Vector<I> v = new Vector<I>();
        int count = 0;
        for ( int i = this.front ;
              count < this.size ;
              i = this.increment(i), count++) {
            v.add((I) this.queue[i]);
        }
        return v;
    }

    @Override
    public boolean equals(Object obj) {
        boolean answer = false;
        ArrayBasedQueue<I> otherQ;

        if (obj instanceof ArrayBasedQueue) {
            otherQ = (ArrayBasedQueue<I>) obj;
        } else {
            return answer;
        }

        Vector<I> myPV = this.peekAll();
        answer = myPV.equals(otherQ.peekAll());
        return answer;
    }

    private int increment(int index) {
        index++;
        if (index >= this.queue.length) {
            index = 0;
        }
        return index;
    }
}
```

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) 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

B. (2 Points) A(n) _____ is an instance of a class.

a. method
b. data field
c. interface
d. object

C. (2 Points) When you declare a variable that refers to an object of a given class, you are creating a(n) _____ to the object.

a. interface
b. reference
c. method
d. ADT

D. (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

E. (2 Points) I 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 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) The _____ method of the ADT stack retrieves the top of the stack, but does not change the stack.

a. createStack
b. push
c. pop
d. peek

H. (2 Points) The constructor of a subclass can call the constructor of the superclass by using the _____ reference.

a. extends
b. new
c. super
d. import

2. (20 Points) Corrected generic MyArray class:

```
public class MyArray<I> {
   private Object[] array = new Object[100];
   private int currentLocation = 0;

   public int size() {
      return currentLocation;
   }

   public boolean isEmpty() {
      return (currentLocation == 0);
   }

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

   public I getElement(int index) {
      I element = null;
      if ((index < currentLocation) && (index >= 0)) {
         element = (I) array[index];
      }
      return element;
   }

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

   public void removeElement(int index) {
      if ((index >= currentLocation) || (index < 0)) {
         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;
   }
}
```

Version 2

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

```
import java.util.Vector;

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

   private Object[] stack = new Object[100];
   private int top = -1;
   private int size = 0;

   @Override
   public boolean isEmpty() {
      return (this.size == 0);
   }

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

   @Override
   public void push(I element) {
      if (this.size == this.stack.length) {
         return;
      } else {
         this.stack[++this.top] = element;
      }
      this.size++;
   }

   @Override
   public I pop() {
      I element = null;
      if (!this.isEmpty()) {
        element = (I) this.stack[this.top--];
      }
      this.size--;
      return element;
   }

   @Override
   public Vector<I> peekAll() {
      Vector<I> v = new Vector<I>();
     for ( int i = this.top ; i >= 0 ; i-- ) {
         v.add((I) this.stack[i]);
      }
      return v;
   }

   @Override
   public boolean equals(Object obj) {
      boolean answer = false;
      ArrayBasedStack<I> otherS;

      if (obj instanceof ArrayBasedStack) {
         otherS = (ArrayBasedStack<I>) obj;
      } else {
         return answer;
      }

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

      return answer;
   }
}
```

Version 2

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