Optional. Mark along the line to show your feelings on the spectrum between ☹ and ☺.

Before exam: [☹_________________☺].
After exam: [☹_________________☺].
0. **So it begins (0.5 points)**. Write your name and ID on the front page. Circle the exam room. Write the IDs of your neighbors. Write the given statement. Sign when you’re done with the exam. Write your login in the corner of every page. Enjoy your free half point ☺.

1. **IntList Essentials (3.5 Points).**

   a) Recall the definition of the IntList class:

   ```java
   public class IntList {
       public int val;
       public IntList tail;
       public IntList(int val, IntList tail) {
           this.val = val;
           this.tail = tail;
       }
   }
   ```

   What will be printed by the code below? Write your answer in the three blanks provided.

   ```java
   public static void next1(IntList list) {
       list = list.tail;
   }

   public static IntList next2(IntList list) {
       list = list.tail;
       return list;
   }

   public static void next3(IntList list) {
       IntList temp = list;
       temp = temp.tail;
   }

   public static void main(String[] args) {
       IntList L = new IntList(1, null);
       L = new IntList(2, L);
       L = new IntList(3, L);

       next1(L);
       System.out.println(L.val);  // ___________
       next2(L);
       System.out.println(L.val);  // ___________
       next3(L);
       System.out.println(L.val);  // ___________
   }
   ```
b) Fill in the `strangeInsertFront()` method below such that both print statements at the bottom of this page print out "0 1 2 3 4". You may have only one statement per line (do not add semi-colons, use ours).

```java
public class IntList {
    public int val;
    public IntList tail;
    public IntList(int val, IntList tail) {
        this.val = val;
        this.tail = tail;
    }

    public void print() {
        System.out.print(val + " ");
        if (tail != null) {
            tail.print();
        }
    }

    public IntList strangeInsertFront(int x) {
        IntList temp = tail;
        this.val = x;
        return this;
    }
}

public class P2 {
    public static void main(String[] args) {
        IntList L = new IntList(4, null);
        L = L.strangeInsertFront(3);
        L = L.strangeInsertFront(2);
        L = L.strangeInsertFront(1);

        IntList L2 = L;
        L = L.strangeInsertFront(0);

        L.print(); // should print 0 1 2 3 4
        L2.print(); // should also print 0 1 2 3 4
    }
}
```
2. **SList Debugging and Testing (2.5 Points).** Consider the SList implementation below.

```java
public class SList {
    public class IntNode {
        public int item;
        public IntNode next;
        public IntNode(int i, IntNode n) {
            item = i;
            next = n;
        }
    }
    private static IntNode sentinel;
    public SList() {
        sentinel = new IntNode(982734, null);
    }
    public void insertFront(int x) {
        sentinel.next = new IntNode(x, sentinel.next);
    }
    public int getFront() {
        if (sentinel.next == null) {
            return -1;
        }
        return sentinel.next.item;
    }
}
```

Write a JUnit test that fails on the code above, but would pass on a correct implementation. You may use any JUnit methods like `assertEquals`, `assertNotEquals`, `assertTrue`, `assertFalse`, etc.

```java
@Test
public void test() {
    // Your incorrect code here
    // This would pass on a correct implementation
    // JUnit method calls
}
```
3. ArrayHorse.com (5.5 Points).

a) Consider the code below. Assume N is odd and positive.

```java
public static int[][] genCoolGrid(int N) {
    int[][] grid = new int[N][N];
    for (int i = 0; i <= N / 2; i += 1) {
        for (int j = (N / 2) - i; j <= (N / 2) + i; j += 1) {
            grid[i][j] = 8;
            grid[N - 1 - i][j] = 8;
        }
    }
    return grid; /* sorry, low on space, bad style, but works! */
}
```

Show the return value of `genCoolGrid(5)` in the boxes below. Note: top left is `grid[0][0]`.

```
grid[0][0]  
           
           
           
           
grid[4][0]  
```

b) Suppose we define an interface for functions on integers of a single variable called IUF. An example implementation of this interface that squares integers is shown to the right.

```java
public interface IUF {
    public int apply(int x);
}
public class Squarer implements IUF {
    public int apply(int x) { return x * x; }
}
```

Write a method `mapColumn` that destructively applies the given function to a single column of a 2D array. For example, if we pass in a Squarer object as `f` and 0 as `c`, we'd get back a copy of the grid with column 0 squared.

```java
public static int[][] mapColumn(IUF f, int[][] m, int c) {
    _________________________________
    _________________________________
    _________________________________
    _________________________________
    _________________________________
    return m;
}
```

c) Write code such that `veryCoolGrid` is set equal to 5x5 `coolGrid`, but with the middle column squared.

```java
int[][] veryCoolGrid = _________________________________;
```
4. **All Natural (2.5 points).** Add a new method `allNatural` to the `SList` class. After execution, the `SList` should contain only numbers that are greater than or equal to zero, and in the same order that they were in before the operation began. For example, if the list were `[5, -7, 2, 6, -3, 0]` the list would be `[5, 2, 6, 0]` afterwards. As in class, the sentinel node’s value is a dummy value and is not considered part of the list. We have filled in part of the method for you; each line should be filled in with exactly one statement or variable name (you may not add any semi-colons). **To give you some flexibility, you may fill in EITHER solution below. Do not fill out both.** You should not use new.

```java
public class SList {
    public static class IntNode {
        public int item;
        public IntNode next;
        public IntNode(int item, IntNode next) {
            this.item = item;
            this.next = next;
        }
    }
    IntNode sentinel;
    public SList() {
        sentinel = new IntNode(99999, null);
    }
    public void insertFront(int item) {
        IntNode n = new IntNode(item, sentinel.next);
        sentinel.next = n;
    }

    public void allNatural() {
        IntNode a = sentinel;
        IntNode b = sentinel.next;
        while (b != null) {
            if (b.item >= 0) {
                ____________;
                ____________;
            }
            _____ = _____ .next;
        }
        ____________ = null;
    }
    public void allNatural() {
        IntNode p = ____________;
        IntNode q = p.next;
        while (q != null) {
            if (q.item < 0) {
                ____________;
                ____________;
            }
            q = q.next;
        }
        ____________ = null;
    }
}
```

Circle which of your two solutions you’d like graded. We will ONLY grade one of your solutions. Regrades will not be accepted for mis-circled responses.

LEFT  RIGHT
5. Egg<sub>egg</sub> (5.5 points). You start with only one variable, egg, whose contents are fully described in the following box-and-pointer diagram:

![Box-and-pointer diagram](image)

a) Draw a box-and-pointer diagram if we run the following statements in the order shown:

Statement A: for (int i = 0; i < 2; i += 1) {egg.tail = egg.tail.tail;}
Statement B: egg.tail = new IntList(egg.tail.head - 1, egg.tail);
Statement C: IntList backup = egg;
Statement D: egg = egg.tail.tail;
Statement E: backup.tail = egg;

To make grading easier, redraw your answer below in the box-and-pointer diagram provided. You may not need all of the boxes provided. Work outside these lines will not be graded. Make sure to point the variables egg and backup at the appropriate box. Do not include boxes that cannot be reached from either egg or backup.

![Redrawn diagram](image)

b) Rearrange the 5 statements so that after execution is completed, there exists an IntList with exactly four elements: 1->2->3->4 (not necessarily pointed to by <egg>). Write your response in the blanks below, one letter per blank. Use each letter once.

<table>
<thead>
<tr>
<th>First operation</th>
<th>Second</th>
<th>Third</th>
<th>Fourth</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>egg</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>backup</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6. Array Processing (8 Points). You are a programmer for the people’s glorious state of Oceania. Your citizens have started trying to avoid censorship by rotating their text messages. For example, suppose a citizen wants to use the banned word “light”, for example: “nice light today”. Instead, the citizen might say “ght today nice li”. It is your job to write code to remove banned words. Throughout this problem you may assume that `message` and `banned` are of length greater than zero, and that `banned.length < message.length`. You may also assume that `k` is a valid number¹. Hint: for positive integers, `a % b` returns the remainder of dividing `a` by `b`, e.g. `7 % 3` is `1`.

(a) Fill in the `matches` method below, which returns true if the message starting at `k` matches the banned word, treating the message as circular. For example, if message is `['d', 'd', 'o', 'g', 'b', 'a']` and banned is `['b', 'a', 'd']`, this method will return true for `k = 4`, and false for all other `k`. You may not need all lines. If you don’t use a line, leave it blank.

```java
private static boolean matches(char[] message, char[] banned, int k) {
    for (int i = ____; i < _______________; i += 1) {
        int messageIndex = ________________________________;
        if (_______________________________) {
            ________________________________;
        }
    }
    __________________________________________________________________;
}
```

(b) Complete the helper function `matchStart`, which returns the start index of a banned word in a message. For example, if message is `['d', 'd', 'o', 'g', 'b', 'a']` and banned is `['b', 'a', 'd']`, this method should return `4`, because the match begins at position `4`. If there is no match, return `-1`. You can do this part even if you skipped part a! You may use the method from part a, even if you didn’t actually get it right. You may assume that banned words occur only once (if at all).

```java
public static int matchStart(char[] message, char[] banned) {
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
    __________________________________________________________________
}
```

¹ By valid we mean `0 ≤ k ≤ message.length - 1`
(c) Finally write the filter method, which returns the original message but with the banned word removed. Assume that a banned word, if it appears at all, appears only once. You can do this part even if you skipped parts a and b. You may use your methods from part a or part b in this problem.

Some examples:
- message = ['d', 'd', 'o', 'g', 'b', 'a'], banned = ['b', 'a', 'd'], return ['d', 'o', 'g']
- message = ['d', 'd', 'o', 'g', 'b', 'a'], banned = ['o', 'g'], return ['d', 'd', 'b', 'a']
- message = ['b', 'a', 'b', 'a', 'd', 'd', 'o', 'g'], banned = ['b', 'a', 'd'], return ['b', 'a', 'd', 'd', 'o', 'g']

```java
public static char[] filter(char[] message, char[] banned) {
    ______________________________________;
    if (_______________________________) {
        return __________________________;
    }
    int numToCopy = ____________________________;
    char[] result = new char[_________________________];
    int messageIndex = _____________________________;

    for (int i = 0; ______________ < ___________; i ____________) {
        ______________________________________;
        ______________________________________;
    }

    return ____________________________;
}
```

7. Pip Digs Pep (0 Points). What Bay Area experimental music collective released a 1997 jingle for Pepsi, associating the ubiquitous beverage with “Old outdated software getting thrown into the trash” and a “Medicated ointment being spread on painful rash”?

---

2 By using this trick, citizens can defeat our censorship tool! No doubt our citizens will discover this trick. To fix this, we will simply apply the filter function multiple times.
8. Interfaces (6 points). You may assume that you have a correct implementation of ArrayDeque and LinkedListDeque throughout this problem, and that it implements the Deque interface.

a) Consider the Deque interface from Project 1c. Add a default method reverse that reverses the Deque.

```java
public interface Deque<Item> {
    void addFirst(Item x);
    void addLast(Item x);
    boolean isEmpty();
    int size();
    void printDeque();
    Item get(int index);
    Item removeFirst();
    Item removeLast();

default void reverse() {
    // Designated fun zone. Draw something. Leave a scent on the paper. It is up to you.
}
}
```

Designated fun zone. Draw something. Leave a scent on the paper. It is up to you.
b) Suppose we add another default method to the Deque interface:

A: default void append(Deque<Item> d) { /* code not shown */ }

Suppose that we also add the following four methods to ArrayDeque (not BadArrayDeque):

B: public void append(ArrayDeque<Item> ad) { /* code not shown */ }

@override 
C: public void append(Deque<Item> d) { /* code not shown */ }

D: public void prepend(Deque<Item> d) { /* code not shown */ }

E: public void prepend(ArrayDeque<Item> ad) /* code not shown */

Consider the code below. For each call to append or prepend, tell us what happens by circling exactly one of the provided answers. If you circle a letter (A, B, C, D, or E), you are saying that the method with that label is called. If you circle compile-error you are saying this line will not compile. If you circle runtime-error, you are saying the code will compile, but will crash when this line is reached. Some possibilities may not occur (e.g. there may actually be no compiler errors).

When you’re done, you should have circled exactly 8 items.

```java
public class DMS {
    public static void main(String[] args) {
        Deque<Integer> d = new ArrayDeque<Integer>();
        d.addLast(10);
        d.addLast(20);
        ArrayDeque<Integer> ad = new ArrayDeque<Integer>();
        ad.addLast(30);
        ad.addLast(40);

        d.prepend(d);   Compile-Error    Runtime-Error    A    B    C    D    E
        d.prepend(ad);  Compile-Error    Runtime-Error    A    B    C    D    E
        ad.prepend(d);  Compile-Error    Runtime-Error    A    B    C    D    E
        ad.prepend(ad); Compile-Error    Runtime-Error    A    B    C    D    E

        ad.append(d);   Compile-Error    Runtime-Error    A    B    C    D    E
        ad.append(ad);  Compile-Error    Runtime-Error    A    B    C    D    E
        d.append(d);    Compile-Error    Runtime-Error    A    B    C    D    E
        d.append(ad);   Compile-Error    Runtime-Error    A    B    C    D    E
    }
}
```

/* Did you circle 8 things, one per line? If not, go circle 8 things. */
9. **Dynamic Method Selection and Inheritance (6 points)**. Modify the code below so that the max method works properly. Assume all numbers inserted into a DMSList are positive. You may not change anything in the given code. You may only fill in blanks. You may not need all blanks.

```java
public class DMSList {
    private IntNode sentinel;
    public DMSList() {
        sentinel = new IntNode(-1000, ____________________________);
    }

    public class IntNode {
        public int item;     /* Equivalent of first */
        public IntNode next; /* Equivalent of rest */
        public IntNode(int i, IntNode h) {
            item = i;
            next = h;
        }
        public int max() {
            return Math.max(item, next.max());
        }
    }

    public class ____________________________ {
        ____________________________
        ____________________________
        ____________________________
        ____________________________
        ____________________________
        ____________________________
        ____________________________
        ____________________________
    }

    public void insertFront(int x) {
        sentinel.next = new IntNode(x, sentinel.next);
    }

    public int max() {
        return sentinel.next.max();
    }

    public static void main(String[] args) {
        DMSList dmsl = new DMSList();
        dmsl.insertFront(5);
        dmsl.insertFront(10);
        dmsl.insertFront(20);
        System.out.println(dmsl.max()); /* should print 20 */
    }
} /* end of class */ /* end of test, woo */
```