CIS 1100 Exam 2 Solutions

2.1

The issue is that foo is a null reference. Since we know that the error occurs at this line and that it's a null pointer exception, it must be foo, since foo is the only reference on this line, and it's the only reference being dereferenced.

2.2

```
public static void printGrades(List<Grade> grades)
```

2.3

You could use the setter to prevent users from setting age to unreasonable values, like negative numbers. Other possibilities are changing the setter to only allow increasing the age, or incrementing the age by one.

2.4

```
int[][]
```

3

The corrected code with some comments:

```
   1. public static void insertionSort(List<String> l) {
   2.   for (int i = 1; i < l.size(); i++) {
// each iteration of the loop moves (inserts) the element at index i into the correct
spot in the sorted portion [0, i)

// finds the index to insert at
   3.     int index = i;
   4.     for (int j = i; j > 0; j--) {
   5.       if (l.get(i).compareTo(l.get(j - 1)) < 0) {
   6.         index = j - 1;
   7.       }
   8.     }
```

```
// move the string from the previous spot i to the new spot index
   9.      String value = l.remove(i);
   10.     l.add(index, value);
   11.   }
   12. }
```

The errors were:
line 4: > should be >=
line 5: the first j should be i
line 6: j shold be j - 1
line 9: index should be i

4

```
public static Node combine(Node head1, Node head2) {
  // discard the first element of both, if there is one
  if (head1 != null) {
    head1 = head1.next;
  }
  if (head2 != null) {
    head2 = head2.next;
  }

  // if the first linked list is empty, just return the second
  if (head1 == null) {
    return head2;
  }

  // find the last element of the first linked list
  Node last1 = head1;
  while (last1.next != null) {
    last1 = last1.next;
  }
  // put the second linked list after the end of the first
  last1.next = head2;

  return head1;
}
```

5.1

```
public void setAt(int row, int col, char symbol) {
  if (row < 0 || row >= symbols.length || col < 0 || col >= symbols[0].length) {
    return;
  }
```

```
  symbols[row][col] = symbol;
}
```

5.2

```
public void flip() {
  for (int i = 0; i < symbols.length / 2; i++) {
    char[] temp = symbols[i];
    symbols[i] = symbols[symbols.length - i - 1];
    symbols[symbols.length - i - 1] = temp;
  }
}
```

Alternatively, you could swap each char individually using nested for loops.

5.3

```
public void resize(int rows, int cols) {
  char[][] temp = new char[rows][cols];
  for (int i = 0; i < temp.length; i++) {
    for (int j = 0; j < temp[i].length; j++) {
      temp[i][j] = ' ';
    }
  }

  for (int i = 0; i < symbols.length && i < rows; i++) {
    for (int j = 0; j < symbols[i].length && j < cols; j++) {
      temp[i][j] = symbols[i][j];
    }
  }

  symbols = temp;
}
```