CS61B, Fall 1997
Midterm #1
Professor K. Yelick

Problem #1

(2 points) What is y after the following code executes?

```
static void addOne (int x){
   x += 1;
}
int y = 3;
addOne(y);
```

Answer:

Problem #2

(8 points) Answer questions about the following classes. For parts b-e, choose one of the following:

CE: The code will result in a compiler error from javac.
RT: The code will compile without errors, but will cause an error of some kind run time.
OK: The code will compile and run without errors. Show what the program will print.

```
abstract class A {
   abstract public void foo ();
}
class B extends A {
   public void foo () { System.out.println("Calling B.foo");}
   protected int value = 0;
}
class C extends B {
   public void foo () { System.out.println("Calling C.foo," +
value);}
}
class D extends C {
   public void foo () { System.out.println("Calling D.foo()," +
value);}
```

```
}
```

a. (2 points)

```
A a1 = new A();
a1.foo();
```

b. (2 points)

```
A a2 = new B();
a2.foo();
```

c. (2 points)

```
A a3 = new C();
a3.foo();
```

d. (2 points)

```
B b4 = new D();
((C) b4).foo();
```

Problem #3

(12 points) Consider the following ListNode class definition.

```
class ListNode {
   int item;
   ListNode next;
   /** Postcondition: Constructs a new listnode containing i and n
*/
   ListNode (int i, ListNode n) { item = i; next = n; }
}
```

a. (4 Points) Complete the following code to copy a list.

```
/** Postcondition; returns a copy of l. (Copies all the nodes).
*/
private static ListNode copy(ListNode l) {
   if (l == null) return l;
   else {
      return (new ListNode (_____________ , _____________));
   }
}
```

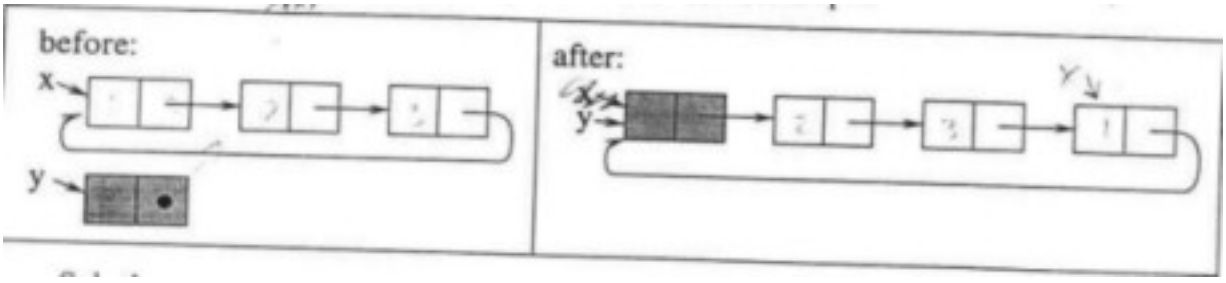
b. (4 Points) Complete the following code to merge 2 sorted lists.

```
/** Precondition: ln1 and ln2 are sorted
 *  Postcondition: returns a new sorted list with all the
   elements of ln1 and ln2, modifying ln1 and ln2 in the process. */
private static ListNode merge(ListNode ln1, ListNode ln2) {
   if (ln1 == null) return (ln2);
   if (ln2 == null) return (ln1);
   if (ln1.item < ln2.item) {

      _____________________________ ;
      return ln1;
   }
   else {

      _____________________________ ;
      return ln2;
   }
}
```

c. (4 Points) Given a non-empty cyclic list x and a single node y, write 2 lines of code to insert y into x after the position x points. Here is an example:



Solution:_____

_____

Problem #4

(6 Points) The following function will sort a stack, placing the smaller elements toward the bottom of the stack. Fill in the missing 2 lines.

```
public static void stackSort (IntStack s1){
   if (s1.isEmpty()) return;
   IntStack s2 = new IntStack();
   int tmp;
   int count = s1.size();
   while (count > 0) {
      int min = s1.pop()
      for (int j = 1; j < count; j++) {
     tmp = s1.pop();
     if (tmp < min) {

         _____

            _____
         }
         else {
            s2.push(tmp);
         }
      }
      s1.push(min);
      while (!s2.isEmpty()) {
         s1.push(s2.pop());
      }
      count--;
   }
}
```

```
/**
 * file : IntStack.java
 * desc : Implements the class Stack
 */
```

```
public class IntStack
{
   /**
    * post : constructs an empty stack
    */
   public IntStack()
   {
      max = 10;
      elems = new int [max];
      top = 0;
   }

   /**
    * post : returns true <==> stack is empty
    */
   public boolean isEmpty()
   {
      return (top == 0);
   }

   /**
    * post : returns the number of elements in the stack
    */
   public int size()
   {
      return (top);
   }

   /**
    * pre : isEmpty() == false
    * post: removes and returns element at the top
    */
   public int pop()
   {
      return elems[--top];
   }

   /**
    * post : put ‘elem’ at the top
    */
   public void push(int elem)
   {
```

```
      checkSize();
      elems[top++] = elem;
   }

   public String toString ()
   {
      String result = "[ ";
      for (int i = 0; i < top; i++) {
         result += elems[i] + " ";
      }
      result += "]";
      return result;
   }

   // private fields
   private int max;        // Current capacity of stack
   private int top;        // Current number of stack elements.
   private int[] elems;    // Data in stack (elems[t-1] is top).

   /**
    * post : If the stack was full, the capcity is expanded
    *        (doubled)
    */
   private void checkSize()
   {
      if (top == max)
      {
         // double the capacity
         int newmax = max << 2;
         int[] newelems = new int [newmax];

         // copy the data
         int i;
         for ( i = 0; i < max; i++ )
            newlems[i] = elems[i];

         // point to the new data
         max = newmax;
         elems = newelems;
      }
   }
}
```
