NAME: **login ID:**
precept:

COS 126 Written Exam 2, Fall 2009

This test is 11 questions, weighted as indicated. The exam is closed book, except that you are allowed to use a one page cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. *Put your name, login ID, and precept number on this page* (*now*), and write out and sign the Honor Code pledge before turning in the test. You have 50 minutes to complete the test.

"I pledge my honor that I have not violated the Honor Code during this examination."

Signature

1	Binary Search Trees	/8
2	DFAs	/12
3	Linked Lists	/15
4	ADTs and APIs	/10
5	Mergesort	/10
6	Architecture	/6
7	Circuit	/8
8	Theory	/12
9	Church-Turing	/6
10	TSP performance	/5
11	Debugging	/5
TOTAL		/97

December 17, 2009

1. **Binary Search Trees** (8 points).

   Draw the binary search tree that results when you insert the keys

   ```
   33 67 14 26 43 71 70
   ```

   in that order into an initially empty tree.

   Circle the statements below that explain why using binary search trees makes sense (more than one might apply).

   A. Search in BSTs is always much faster than in linked lists.

   B. Everything in a computer is encoded in binary.

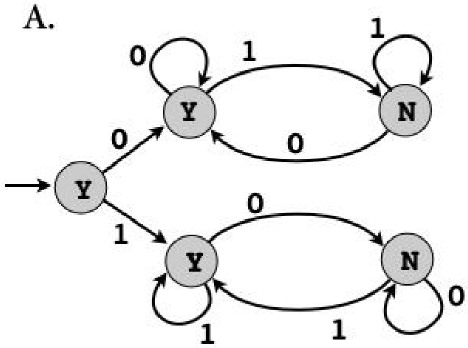
   C. BSTs are a reasonable way to implement efficient symbol tables for typical clients.

   D. Inserting and deleting data items is much easier in BSTs than in sorted arrays.

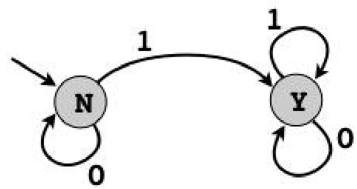
   E. BSTs use less space than linked lists.

2. **Deterministic Finite Automata** (12 points). Consider the following DFAs:

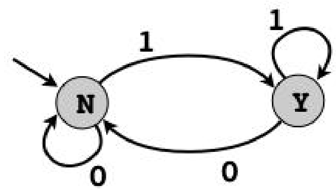
A.



B.



C.



Match each DFA with one of the descriptions below by writing its letter in the blank to the left of the corresponding description. Since there are two more descriptions than DFAs, you must leave two options blank.

______ Bitstrings with at least one `1`

______ Bitstrings with an equal number of occurrences of `01` and `10`

______ Bitstrings with more `1`s than `0`s

______ Bitstrings with an equal number of occurrences of `0` and `1`

______ Bitstrings that end in `1`

3. **Linked Lists** (15 points). Consider the following Java class, which implements a linked list data structure:

```
public class Test
{
   private Node start;
   private Node finish;

   private class Node
   {
      private int key;
      private Node next;
      public Node(int key)
      {  this.key = key; this.next = null;  }
   }

   public Test()
   {  start = null; finish = null;  }

   // Instance methods A(), B(), C(), D(), E().

}
```

Answer the questions on the following page about these five instance methods for this class:

```
public void A(int key)
{
   Node x = new Node(key);
   if (start == null)
       start = x;
   else finish.next = x;
   finish = x;
}

public int B()
{  return finish.key;  }


public void C()
{
   Node t = start;
   while (t != null)
   {
     System.out.print(t.key + " ");
     t = t.next;
   }
   System.out.println();
}
```

```
public void D()
{
   if (start == finish)
      start = finish = null;
   else
   {
      Node t = start;
      while (t.next != finish)
          t = t.next;
      t.next = null;
      finish = t;
   }
}

public Test E()
{
   Test two = new Test();
   while (start != null)
   {
      two.A(this.B());
      this.D();
   }
   return two;
}
```

(*i*) (10 points) Match each instance method with one of the descriptions below by writing its letter in the blank to the left of the corresponding description. Since there are twice as many descriptions as instance methods, you must leave five options blank.

______ Prints the list in order

______ Prints the list in reverse order

______ Adds a key at the beginning of the list

______ Adds a key to the end of the list

______ Copies the list in order

______ Copies the list in reverse order

______ Returns the key in the first element of the list

______ Returns the key in the last element of the list

______ Removes the first element of the list

______ Removes the last element in the list

(*ii*) (5 points) In the space provided, write the output produced by the following code:

```
Test test = new Test();
test.A(1); test.A(7); test.A(4); test.A(9);
test.C();
test.D();
(test.E()).C();
```

first line of output: ______________________________

second line of output: ______________________________

4. **ADTs and APIs** (10 points). Complete the following API for a `StringSET` abstract data type by filling in the blanks to the left of the given method descriptions with full method signatures. (one answer is provided for you). For each method you must provide a return type (if any), name, parameter types, and parameter names. Method names are your choice, but each name should clearly and succinctly identify the behavior of its method.

```
public class StringSET
```

________________________________	`constructor;` `initialize to empty set`
`public void add(String s)`	`add the given string` `to the set`
________________________________	`remove the given string` `from the set`
________________________________	`return true if the given` `string is in the set;` `false otherwise.`
________________________________	`return the union of this` `set and the given set`
________________________________	`return the intersection of` `this set and the given set`

5. **Mergesort** (10 points). Consider the following implementation of recursive mergesort:

```
public class Mergesort
{
  public static void sort(Comparable[] a, Comparable[] aux, int lo, int hi)
  {
     if (hi - lo <= 1) return;
     int mid = lo + (hi - lo) / 2;
     sort(a, aux, lo, mid);
     sort(a, aux, mid, hi);
     merge(a, aux, lo, mid, hi); // Sorts a[lo..hi-1].

     System.out.print(lo + " " + hi + " ");
     for (int i = 0; i < a.length; i++)
        System.out.print(a[i] + " ");
     System.out.println();
  }

  public static void sort(Comparable[] a)
  {
     int N = a.length;
     Comparable[] aux = new Comparable[N];
     sort(a, aux, 0, N);
  }
}
```

Note that the last three lines of the recursive method have been instrumented to print the values of the indices and the contents of the array. Give the output produced by these methods when invoked by the following code (the first line is filled in for you):

```
Character[] a = { 'y', 'i', 'b', 'w', 'l', 'o', 'l' };
Mergesort.sort(a);
```

```
1    3    y b i w l o l

____ ____ ____________________

____ ____ ____________________

____ ____ ____________________

____ ____ ____________________

____ ____ ____________________
```

6. **Architecture** (6 points). Suppose that we plan to expand the TOY architecture to support 22-bit instructions (rather than 16-bit instructions) with the following format:

```
            0  1  2  3  |  4  5  6  7  8  9  | 10 11 12 13 14 15 | 16 17 18 19 20 21
Format 1:    opcode          destination d          source s            source t
Format 2:    opcode          destination d      -  -  -  -  - address -  -  -  -  -
```

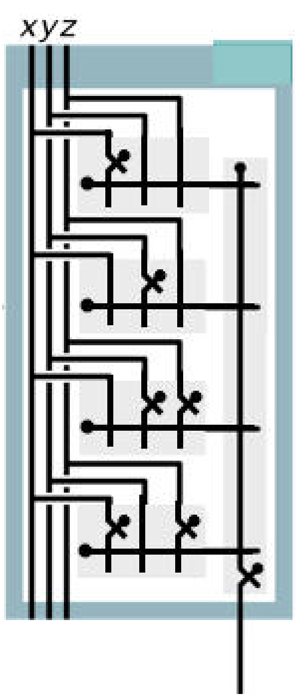
Answer the three questions below, by writing one of the following numbers in the blank to the right of each question: 2, 4, 8, 12, 16, 22, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384.

(*i*) How many different opcodes can be encoded ? ____________

(*ii*) How many registers could the machine have ? _____________

(*iii*) How many words could the addressable memory have ? _____________

7. **Circuits** (8 points). In the space to the right, give the truth table for the Boolean function computed by the following circuit:



8. **Theory** (12 points). In the blanks, mark each of the statements below as true (T) or false (F).

   A. ______ Your iPhone is equivalent to a Universal Turing Machine (UTM).

   B. ______ Any app on your iPhone is equivalent to a UTM.

   C. ______ A UTM can simulate the operation of any Turing machine, including itself.

   D. ______ If P=NP, there is no problem that is in NP but is not NP-complete.

   E. ______ A UTM can decide whether a given string is in the language described by a given regular expression.

   F. ______ No Turing machine can decide whether a given DFA halts.

9. **Church-Turing Thesis** (6 points). In the blanks, mark each of the statements about the Church-Turing thesis below as true (T) or false (F).

   A. ______ The Church-Turing thesis is a theory about our universe: it cannot be proven mathematically.

   B. ______ The Church-Turing thesis implies that no computer can solve the halting problem.

   C. ______ The Church-Turing thesis implies that there is no need to investigate computational limits for every computer individually.

10. **TSP performance** (5 points). Suppose that we have a program that finds a solution to the traveling salesperson problem which takes 1000 seconds to find an optimal solution for a 1000-city tour, and, for an N-city tour, we suspect it takes time proportional to 2^N.

How much time should we estimate for an optimal 2000-city tour? You may use a power of two in your answer—an answer of the form "$99 \cdot 2^{99}$ seconds" is preferred.

11. **Debugging** (5 points) *Do not attempt this question until you have completed the rest of the exam, as it may take too much time.* One of the instance methods in question 3 has a bug. In the space provided, briefly describe the bug and write the name of the method.

description of bug: ____

method with bug: ________