COS 126	General Computer Science	Fall 2016

Written Exam 2

This exam has 9 questions (including question 0) worth a total of 70 points. You have 50 minutes. Write all answers inside the designated spaces.

Policies. The exam is closed book, except that you are allowed to use a one page cheatsheet (8.5-by-11 paper, both sides, in your own handwriting). No electronic devices are permitted.

Discussing this exam. Discussing the contents of this exam before solutions have been posted is a violation of the Honor Code.

This exam. Do not remove this exam from this room. Print your name, NetID, and the room in which you are taking the exam in the space below. Mark your precept number. Also, write and sign the Honor Code pledge. You may fill in this information now.

Name:

NetID:

Exam room:

Precept:

P01	P01A	P01B	P01C	P01D	P02	P02A	P03	P04
○	○	○	○	○	○	○	○	○
P05	P05A	P06	P11	P11A	P12	P13	P14	
○	○	○	○	○	○	○	○	

"I pledge my honor that I will not violate the Honor Code during this examination."

Signature

0	1	2	3	4	5	6	7	8	total

0. **Miscellaneous. (1 points)**

   (a) Write your name, NetID, and the room in which you are taking the exam in the space provided on the front of this exam.

   (b) Mark your precept number on the front of this exam.

   (c) Write and sign the Honor Code pledge on the front of this exam.

1. **Properties of reference types. (6 points)**

   For each description on the left, write the letter corresponding to the best-matching term on the right. Use each letter at most once.

___ A set of values and operations on those values.

___ A data type for which the data-type value of any instance cannot change, such as `String`.

___ An object that is no longer accessible in a program.

___ The process of automatically identifying and freeing memory when it is no longer in use.

___ Mechanism by which Java passes both primitive types and object references to methods.

___ Two (or more) variables that refer to the same object.

A. orphan

B. aliasing

C. garbage collection

D. encapsulation

E. immutable

F. mutable

G. pass by value

H. pass by reference

I. design by contract

J. data type

2. **Object-oriented programming. (9 points)**

For each box below, write the letter corresponding to the best-matching description of that part of the program. Use each letter at most once.

A. reference type

B. primitive type

C. instance variable declaration

D. local variable declaration

E. constructor

F. instance method

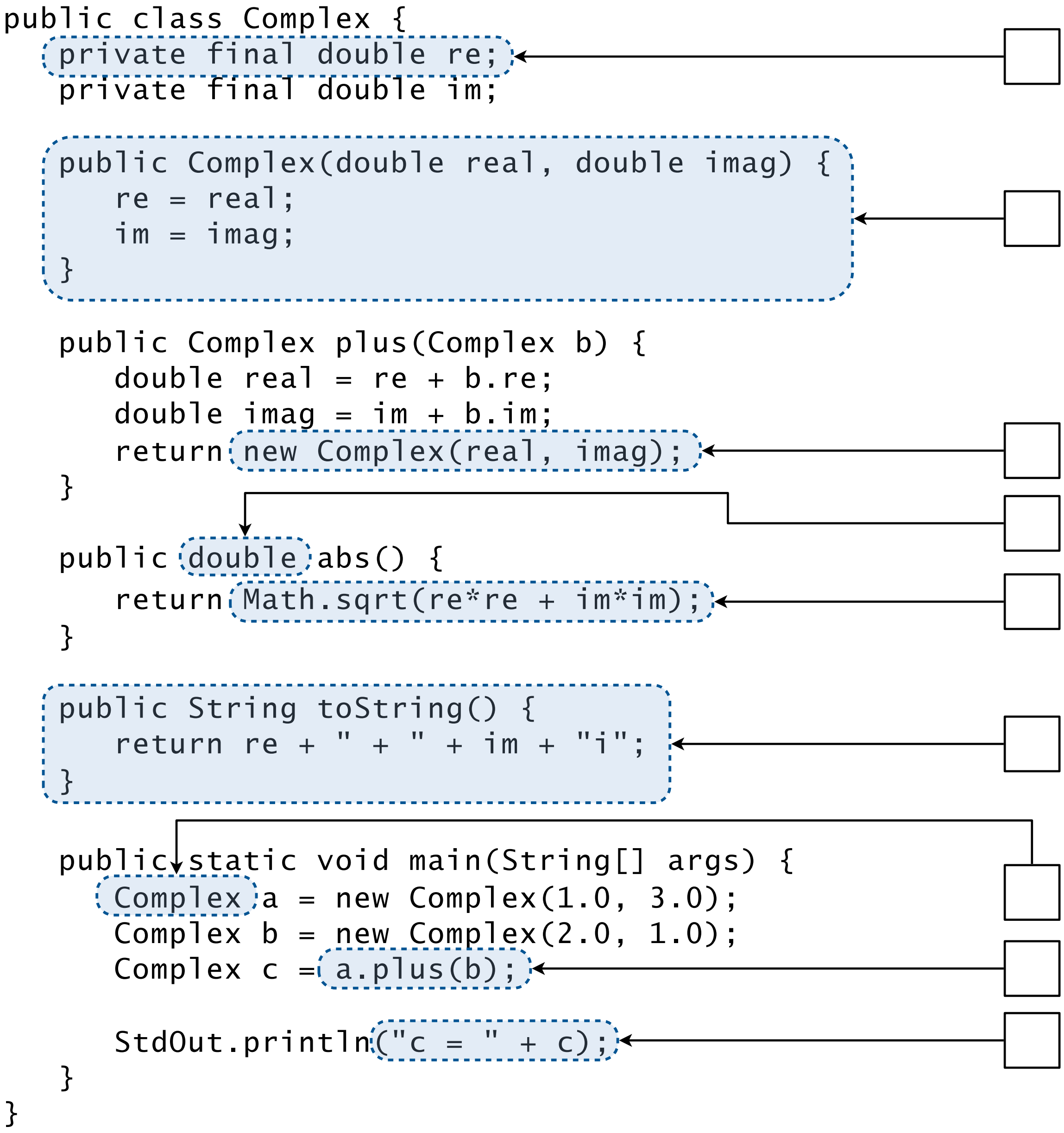
G. static method

H. static method call

I. constructor call

J. instance method call

K. `toString()` method call



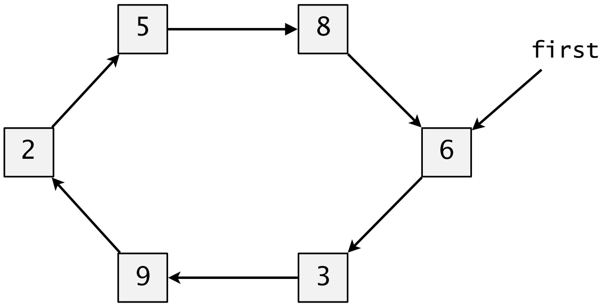
```
public class Complex {
   private final double re;
   private final double im;

   public Complex(double real, double imag) {
      re = real;
      im = imag;
   }

   public Complex plus(Complex b) {
      double real = re + b.re;
      double imag = im + b.im;
      return new Complex(real, imag);
   }

   public double abs() {
      return Math.sqrt(re*re + im*im);
   }

   public String toString() {
      return re + " + " + im + "i";
   }

   public static void main(String[] args) {
      Complex a = new Complex(1.0, 3.0);
      Complex b = new Complex(2.0, 1.0);
      Complex c = a.plus(b);

      StdOut.println("c = " + c);
   }
}
```

3. **Linked structures. (8 points)**

Suppose that the `Node` data type is defined as

```
private class Node {
   private int item;
   private Node next;
}
```

and that `first` is a variable of type `Node` that refers to one node in a circularly linked list containing $n > 0$ nodes, as in this diagram:



For each code fragment on the left, determine how many integers it prints as a function of n by choosing the best-matching term at right. Use each letter at most once.

___

```
for (Node x = first; x != null; x = x.next)
    StdOut.println(x.item);
```

___

```
for (Node x = first; x != first; x = x.next)
    StdOut.println(x.item);
```

___

```
Node x = new Node();
x = x.next;
while (x != first) {
    StdOut.println(x.item);
    x = x.next;
}
```

___

```
Node x = first;
do {
    x = x.next;
    StdOut.println(x.item);
} while (x != first);
```

A. 0

B. 1

C. $n - 1$

D. n

E. $n + 1$

F. $2n$

G. n^2

H. infinite loop

I. run-time error

4. **Sorting and searching. (10 points)**

(a) Suppose that you are performing insertion sort on the following array:

44	22	17	49	33	10

Which pairs of keys get compared at some point during the sort?
Mark all that apply.

22–44	17–22	17–49	22–33	10–22	10–33
☑	□	□	□	□	□

(b) Suppose that you are merging the following two sorted subarrays during mergesort.

11	22	33	88

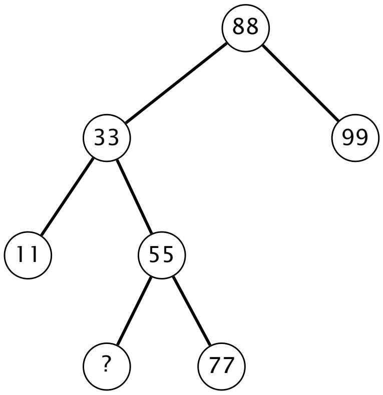
44	55	66	77

Which pairs of keys get compared at some point during the merge?
Mark all that apply.

11–44	11–22	11–55	22–55	44–88	77–88
☑	□	□	□	□	□

5. **Symbol tables. (10 points)**

(a) Suppose that you insert each sequence of keys at right into an initially empty binary search tree. Mark each sequence that would produce the BST below. The question mark (?) denotes some unrevealed integer.



yes	*no*	*keys to insert (in order given)*
○	○	`88 33 11 55 44 77 99`
○	○	`11 22 33 55 77 88 99`
○	○	`88 99 33 11 55 22 77`
○	○	`88 33 99 11 55 77 66`

(b) Assume that standard input consists of a sequence of n integers, separated by whitespace. What is the order of growth of the worst-case running time of the following code fragment as a function of n?

```
ST<Integer, Integer> st = new ST<Integer, Integer>();
while (!StdIn.isEmpty()) {
    int x = StdIn.readInt();
    if (st.contains(x)) st.put(x, st.get(x) + 1);
    else                st.put(x, 1);
}
for (int x : st.keys()) {
    for (int i = 0; i < st.get(x); i++)
        StdOut.println(x);
    }
}
```

1	$\log n$	n	$n \log n$	n^2	2^n
○	○	○	○	○	○

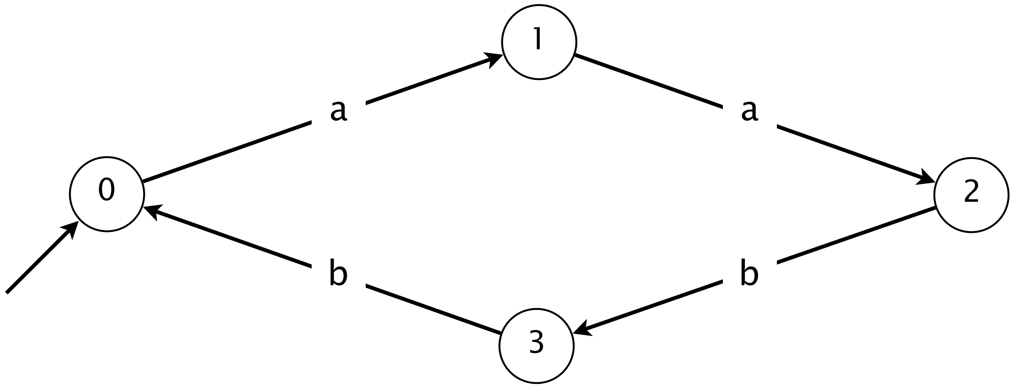
(c) Describe what the code fragment in part (b) prints? Write your answer in the box. Your answer will be graded on correctness, clarity, and conciseness.

6. **Regular expressions and DFAs. (10 points)**

Consider the language over the alphabet $\{a, b\}$ that contains all binary strings whose second-to-last symbol is a (and no other strings).

(a) Write a regular expression that specifies the language L.

(b) Design a DFA that recognizes the language L by completing the partial DFA below. *Hint:* state 3 represents all binary strings that end in `ab`.



last two symbols are ab

i. Which state is the *start state*? Mark one.

0	1	2	3
○	○	○	○

ii. Which states are *accept states*? Mark all that apply.

0	1	2	3
□	□	□	□

iii. Which transitions are labeled with the symbol `a`? Mark all that apply.

0 → 1	1 → 2	2 → 1	2 → 2	2 → 3	3 → 1	3 → 2	3 → 3
☑	☑	□	□	□	□	□	□

iv. Which transitions are labeled with the symbol `b`? Mark all that apply.

0 → 0	0 → 1	0 → 2	1 → 1	1 → 2	1 → 3	2 → 3	3 → 0
□	□	□	□	□	□	☑	☑

7. **Theory of computing. (8 points)**

(a) For each informal description on the left, choose the best-matching term on the right that formalizes it. Use each letter at most once.

___ An efficient solution to a problem.

___ An algorithm.

___ You can program a computer to implement any algorithm.

___ A class of problems that probably cannot be solved efficiently.

A. Turing machine

B. Universal Turing machine

C. polynomial-time algorithm

D. exponential-time algorithm

E. P

F. NP

G. NP-complete

(b) For each statement on the left, match the concept on the right that would imply it. Use each letter at most once.

___ If you can solve SAT efficiently on a computer that harnesses properties of dark matter, then you can also do so on a Macbook Pro.

___ If you can solve TSP efficiently, then you can also solve SAT efficiently.

___ If you can solve SAT efficiently, then you can also solve TSP efficiently.

___ TSP cannot be solved efficiently.

A. Cook–Levin theorem

B. Kleene's theorem

C. Church–Turing thesis

D. extended Church–Turing thesis

E. Karp's poly-time reductions

F. P ≠ NP conjecture

8. **Circuits. (8 points)**

The 3-bit *even parity* function $f(x, y, z)$ is 1 if an even number of its inputs are 1, and 0 otherwise. Which of the following represent the even parity function? Check all that apply.

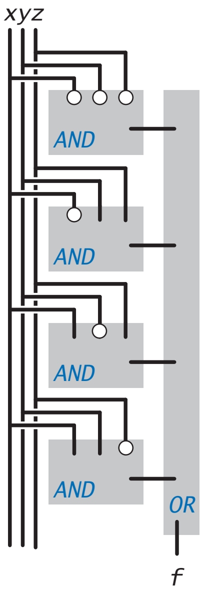
yes *no*

● ○

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

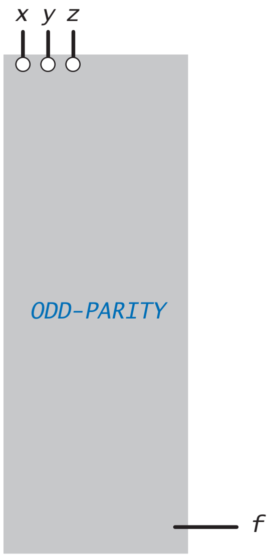
yes *no*

○ ○



yes *no*

○ ○



yes *no*

○ ○ $f = xyz + xy'z' + x'yz' + x'y'z$

yes *no*

○ ○

```
public static boolean f(boolean x, boolean y, boolean z) {
    if (x && y) return !z;
    if (x || y) return z;
    return !z;
}
```