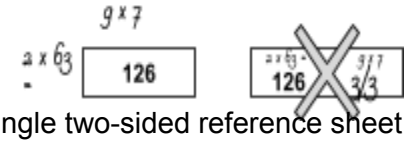
COS126 **Written Exam 2** **Spring** '23

Instructions. This exam has eight (8) questions worth a total of one hundred (100) points. You have eighty (80) minutes.

This exam is preprocessed by computer. Write neatly, legibly, and darkly. If you use a pencil, use extra care to write darkly. Put all answers (and nothing else) inside the designated boxes. Fill in bubbles and checkboxes completely: ⚫ and ⬛ (not ✔ or ✘). Place only your answer, no calculations, inside a box:

To change an answer, erase it completely and redo.



Resources. The exam is closed book, except that you are allowed to use a single two-sided reference sheet (8.5-by-11 paper, two-sided, in your own handwriting). No electronic devices are permitted.

Honor Code. This exam is governed by Princeton's Honor Code. Discussing the contents of this exam before solutions have been posted is a violation of the Honor Code.

NAME:

NETID (not alias)

PRECEPT

P01	P02	P02A	P03	P04	P05	P06	P07	
◯	◯	◯	◯	◯	◯	◯	◯	
P08	P08A	P10	P11	P12	P13	P14	P15	ISC
◯	◯	◯	◯	◯	◯	◯	◯	◯

EXAM ROOM McCosh 50 ◯ McCosh 10 ◯ McCosh 62 ◯ OTHER _____

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

Signature _____

TOY REFERENCE CARD

INSTRUCTION FORMATS

```
             | . . . . | . . . . | . . . . | . . . .|
  Format RR: | opcode  |    d    |    s    |    t   |  (0-6, A-B)
  Format A:  | opcode  |    d    |       addr       |  (7-9, C-F)
```

ARITHMETIC and LOGICAL operations

```
    1: add              R[d] <- R[s] +  R[t]
    2: subtract         R[d] <- R[s] -  R[t]
    3: and              R[d] <- R[s] &  R[t]
    4: xor              R[d] <- R[s] ^  R[t]
    5: shift left       R[d] <- R[s] << R[t]
    6: shift right      R[d] <- R[s] >> R[t]
```

TRANSFER between registers and memory

```
    7: load address     R[d] <- addr
    8: load             R[d] <- M[addr]
    9: store            M[addr] <- R[d]
    A: load indirect    R[d] <- M[R[t]]
    B: store indirect   M[R[t]] <- R[d]
```

CONTROL

```
    0: halt             halt
    C: branch zero      if (R[d] == 0) PC <- addr
    D: branch positive  if (R[d] >  0) PC <- addr
    E: jump register    PC <- R[d]
    F: jump and link    R[d] <- PC; PC <- addr
```

Register 0 always reads 0.
Loads from M[FF] come from stdin.
Stores to M[FF] go to stdout.

16-bit registers (two's complement)
16-bit memory locations
 8-bit program counter

NOTE: All memory locations not specified have undefined values.

1. After executing this TOY program, what is the final value of **R[2]** in decimal? Place your answer, and only your answer, in the rectangle below the program.

```
10:  7C05
11:  7101
12:  7201
13:  1222
14:  2CC1
15:  DC13
16:  0000
```

NOTE: All memory locations not specified have undefined values. There may be more than one correct answer.

2. How would you change memory location **10** to make **R[2]**'s final value be 2?

Place your answer in the rectangle to right of **10:**
One hex symbol per box.

```
10:  7
11:  7101
12:  7201
13:  1222
14:  2CC1
15:  DC13
16:  0000
```

NOTE: All memory locations not specified have undefined values.There may be more than one correct answer.

3. How would you change memory location **13** to make **R[2]**'s final value be 2?

Place your answer in the rectangle to right of **13:**
One hex symbol per box.

```
10:  7C05
11:  7101
12:  7201
13:  1
14:  2CC1
15:  DC13
16:  0000
```

NOTE: All memory locations not specified have undefined values. There may be more than one correct answer.

4. How would you change memory location **15** to make **R[2]**'s final value be 2?

Place your answer in the rectangle to right of **15:**
One hex symbol per box.

```
10:  7C05
11:  7101
12:  7201
13:  1222
14:  2CC1
15:  DC
16:  0000
```

```
01  public class Point {
02    private final double x;  // x-coordinate
03    private final double y;  // y-coordinate
04    public Point(double x, double y) {
05         this.x = x;
06         this.y = y;
07    }
08
09    public Point() {
10         x = 0;
11         y = 0;
12    }
13
14    public double x() {
15         return x;
16    }
17
18    public double y() {
19         return y;
20    }
21
22    private double compute(double x, double y) {
23         return Math.sqrt(x * x + y * y);
24    }
25
26    public double r() {
27         return compute(x, y);
28    }
29
30    public double distanceTo(Point that) {
31         double dx = this.x - that.x;
32         double dy = this.y - that.y;
33         return compute(dx, dy);
34    }
35    public String toString() {
36         return "(" + x + ", " + y + ")";
37    }
38    public static void main(String[] args) {
39         Point p = new Point();
40         StdOut.println("p  = " + p);
41         StdOut.printf("x=%f,y=%f,r=%f\n",
42                       p.x(), p.y(), p.r());
43         Point q = new Point(0.5, 0.5);
44         StdOut.println("q  = " + q);
45         StdOut.println(p.distanceTo(q));
46         q = p;
47         StdOut.println("dist(q, p) = " +
48         q.distanceTo(p));
49    }
50  }
```

Consider the Java class, **`Point`,** on the left. The code compiles successfully. Answer the following questions by filling in the bubble ⚫. Selecting *Not sure* will give partial credit of .5 points.

1. The **`Point`** class is an abstract data type.
◯ True ◯ False ◯ Not sure (.5)

2. **`Point`** is an immutable data type.
◯ True ◯ False ◯ Not sure (.5)

3. All the methods defined in the **`Point`** class are examples of instance methods.
◯ True ◯ False ◯ Not sure (.5)

4. Removing **`this.`** (lines 31-32) will result in a compilation error.
◯ True ◯ False ◯ Not sure (.5)

5. The local variables accessed in line 23 shadow the instance variables in lines 02-03.
◯ True ◯ False ◯ Not sure (.5)

6. It is possible to keep the API unchanged if the instance variables are changed to polar coordinates (radius & angle instead of x & y):
```
private final double r;
private final double theta;
```
◯ True ◯ False ◯ Not sure (.5)

7. The statement at line 46 will throw an exception since **`q`** and **`p`** were initialized with different **`Point`** constructors.
◯ True ◯ False ◯ Not sure (.5)

Question 3 — Linked Data Structures — 15 points

Assume you have access to the `private Node` class:

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

Now consider the following method which operates on linked lists:

```
public boolean slinky(Node head) {
    Node a = head;
    if (a == null) return true;
    Node b = a.next;
    while (b != null && b != a) {
        b = b.next;
        if (b == null) return true;
        b = b.next;
        a = a.next;
    }
    return (b == null);
}
```

Answer the following questions by filling in one bubble ⚫.

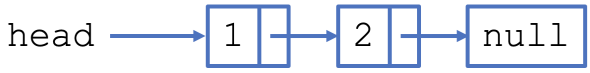
---

1. What does **`slinky(head)`** return on the following empty list?

◯ `true` ◯ `false` ◯ does not `return`

---

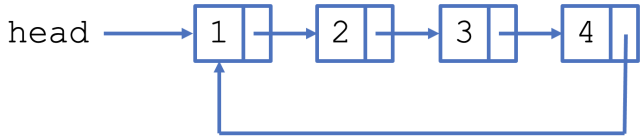
2. What does **`slinky(head)`** return on the following null-terminated list?



◯ `true` ◯ `false` ◯ does not `return`

---

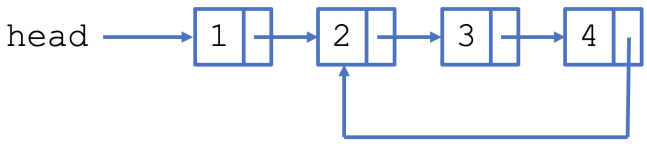
3. What does **`slinky(head)`** return on the following list with a cycle?



◯ `true` ◯ `false` ◯ does not `return`

---

4. What does **`slinky(head)`** return on the following list with a cycle?



◯ `true` ◯ `false` ◯ does not `return`

---

5. **`slinky`** always returns **`false`** when the linked data structure ___. Select the best answer.

◯ is sorted ◯ isn't sorted ◯ is null-terminated
◯ is empty ◯ isn't empty ◯ is doubly-linked
◯ has a cycle ◯ has no cycle

In the following functions, the input array, **`inArray`**, is of size **`N`**. What is the order of growth in terms of **`N`** for each function?

```
/* What is the asymptotic growth rate of mystery1? */
public static int mystery1(int[] inArray, int N) {
    int sum = 0;

    for (int i = 0; i < N; i++)
        for (int j = 0; j < i / 2; j++)
            sum += inArray[j];

    for (int i = 0; i < N; i++)
        sum += inArray[i];

    return sum;
}
```

1. Fill in a single bubble ⚫ to indicate the order of growth:

1	*logN*	*N*	*NlogN*	N^2
◯	◯	◯	◯	◯

N^3	2^N	3^N	*N!*
◯	◯	◯	◯

```
/* What is the asymptotic growth rate of mystery2? */
public static int mystery2(int[] inArray, int N) {
    int sum = 0;

    for (int i = 0; i < 10; i++)
        for (int j = 0; j < N; j++)
            sum += inArray[j];

    return sum;
}
```

2. Fill in a single bubble ⚫ to indicate the order of growth:

1	*logN*	*N*	*NlogN*	N^2
◯	◯	◯	◯	◯

N^3	2^N	3^N	*N!*
◯	◯	◯	◯

```
/* This function is called by mystery3 below. */
public static int mystery3_helper(int[] inArray,
                                  int left,
                                  int right) {
    int sum = 0;
    int mid = ((right - left) / 2) + left;

    if (left == right) return 0;
    if (left + 1 == right) return 0;

    for (int i = left; i <= right; i++)
        sum += inArray[i];

    return mystery3_helper(inArray, left, mid) +
           mystery3_helper(inArray, mid, right) +
           sum;
}

/* What is the asymptotic growth rate of mystery3
 * including the time spent in mystery3_helper?     */
public static int mystery3(int[] inArray, int N) {
    return mystery3_helper(inArray, 0, N - 1);
}
```

3. Fill in a single bubble ⚫ to indicate the order of growth:

1	*logN*	*N*	*NlogN*	N^2
◯	◯	◯	◯	◯

N^3	2^N	3^N	*N!*
◯	◯	◯	◯

Question 5 **Regular Expressions and DFAs** **12** points

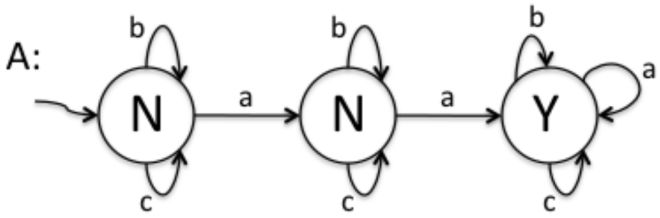
For each of the following sets of strings over the alphabet **{a, b, c}**, choose the single DFA that accepts that same set. Fill in the bubble ⚫ with the letter corresponding to the DFA.

1. The regular expression
   (b | c)* a (b | c)* a (a | b | c)*

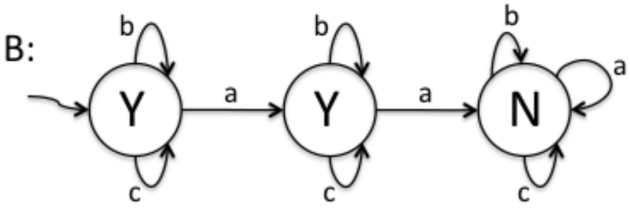
A B C D E

◯ ◯ ◯ ◯ ◯

A:



B:

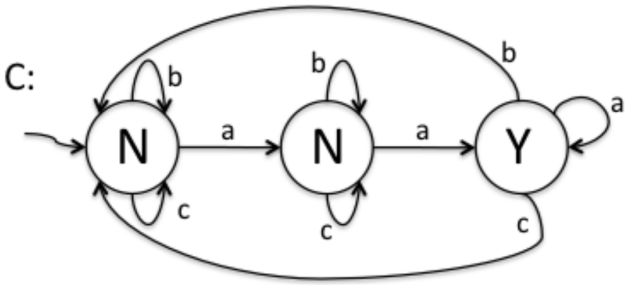


2. Strings containing at most one **a**

A B C D E

◯ ◯ ◯ ◯ ◯

C:

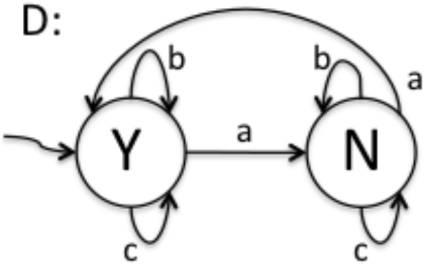


3. Strings containing an even number of **a**'s (including no **a**'s)

A B C D E

◯ ◯ ◯ ◯ ◯

D:



4. Strings containing at least two **a**'s

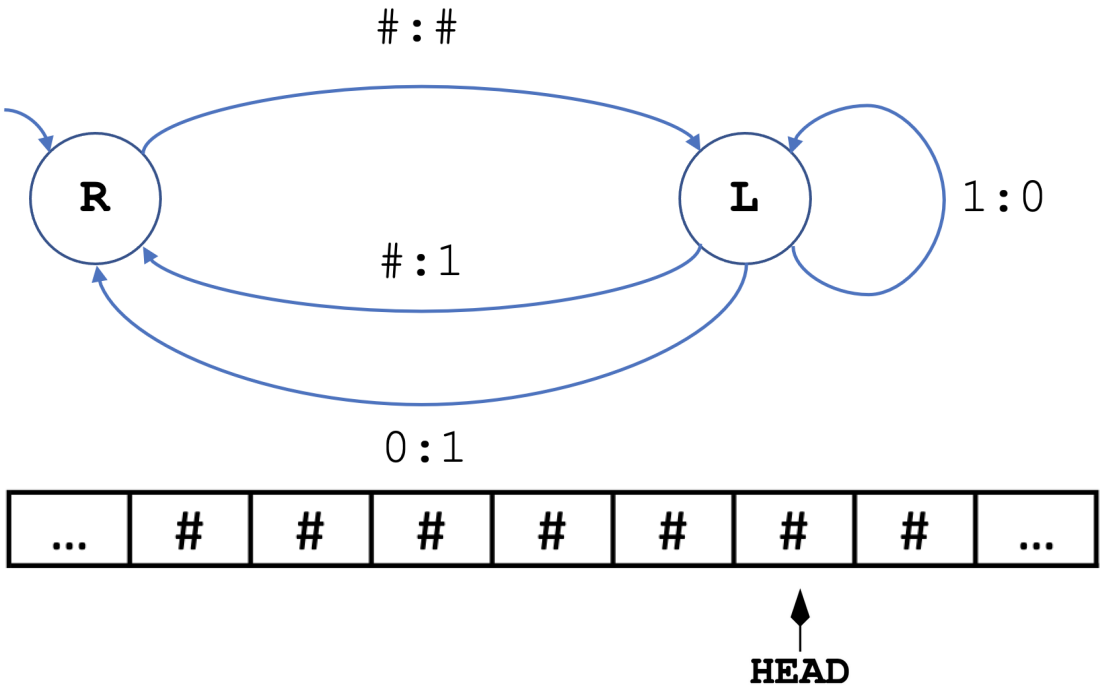
A B C D E

◯ ◯ ◯ ◯ ◯

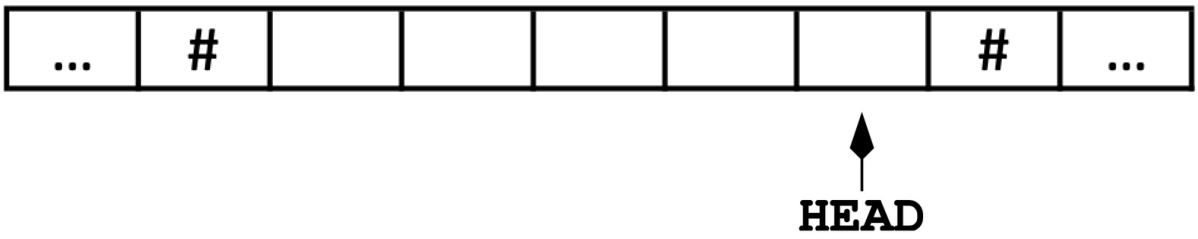
E: None of the above.

Question 6	**Turing Machines**	**12** points

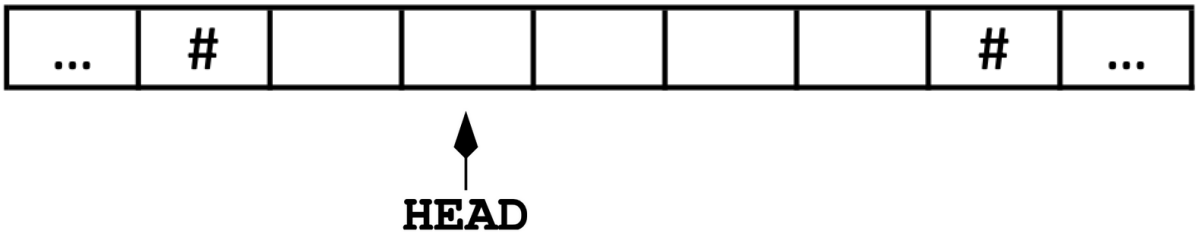
Consider the following Turing machine and tape: (Note: This Turing machine has no HALT, YES, or NO states. **# denotes empty.**)



1. The tape starts with all empty cells as shown above. The initial state is R. Show the state of the tape after the transition and head move in which the tape goes from zero (0) to one (1) non-empty cell. Fill in all cells with only a symbol (**0**, **1**) or **#**:



2. Show the state of the tape after the transition and head move during which the tape goes from three (3) to four (4) non-empty cells. Fill in all cells with only a symbol (**0**, **1**) or **#**:



3. What **best** describes what this Turing machine does in general?

◯ hamming decode ◯ hamming encode ◯ writes all 1's ◯ writes all 0's
◯ writes in alternating 1's and 0's ◯ counts in binary ◯ converts 1's to 0's
◯ converts 0's to 1's ◯ converts 0's to 1's, 1's to 0's, and #'s to 1's ◯ halts

Question 7	**Theory**	**12** points

Suppose that we have just proven that P is **not equal** to NP. In this context, verify the following statements by filling in a single bubble ⚫. Select *I'm not sure* for partial credit of .5 points.

1. P is equal to NP.
   ◯ True ◯ False ◯ No one is sure yet ◯ I'm not sure (.5 points)

2. There does not exist an efficient algorithm for finding the non-trivial integer prime factors of a number (FACTOR).
   ◯ True ◯ False ◯ No one is sure yet ◯ I'm not sure (.5 points)

3. The traveling salesperson problem (TSP) is not in P.
   ◯ True ◯ False ◯ No one is sure yet ◯ I'm not sure (.5 points)

4. The "Is this list sorted?" decision problem is not in NP.
   ◯ True ◯ False ◯ No one is sure yet ◯ I'm not sure (.5 points)

5. There exists an efficient algorithm for finding optimal TSP tours, but no one has been able to find it.
   ◯ True ◯ False ◯ No one is sure yet ◯ I'm not sure (.5 points)

6. There does not exist an efficient algorithm for the Boolean satisfiability problem (SAT).
   ◯ True ◯ False ◯ No one is sure yet ◯ I'm not sure (.5 points)

7. No algorithm can guarantee to efficiently find a tour of length less than k for any given k for all TSP point maps.
   ◯ True ◯ False ◯ No one is sure yet ◯ I'm not sure (.5 points)

8. For TSP, there does not exist a class of point maps (i.e., a subset of maps having a certain stated characteristic) for which finding the optimal tour in polynomial time is possible.
   ◯ True ◯ False ◯ No one is sure yet ◯ I'm not sure (.5 points)

Question 8	**Circuits**	**11** points

Consider a circuit that has three inputs A, B, C and produces an output of **1** *if and only if exactly one* of the inputs is **1**.

1. Complete the truth table for this circuit, filling in **0**'s or **1**'s in each box:

A	B	C	RESULT
0	0	0	0
		1	
	1		
			0
1	0	0	
			0
			0
1	1	1	

2. What is the minimum number of **3-input** AND and **3-input** OR gates you need to build this circuit assuming A, A', B, B', C, C' are all available as inputs only to the AND gates? Fill in a single bubble ⚫.

a) Minimum number of 3-input AND gates:

◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7 ◯ 8 ◯ I'm not sure (.5 points)

b) Minimum number of 3-input OR gates:

◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7 ◯ 8 ◯ I'm not sure (.5 points)