COS 126	Princeton University	Spring 2026

Final Exam Solutions

This exam has 15 questions worth a total of 150 points. You have 180 minutes to complete it.

Instructions. This exam is preprocessed by computer. Write neatly and legibly, using a dark pen or pencil. Put all answers (and nothing else) inside the designated spaces. *Fill in* bubbles and checkboxes completely (● and ■). To change an answer, erase it completely and rewrite it.

Resources. The exam is closed book, except that you may use a one-page reference sheet (8.5-by-11 paper, both sides, handwritten by you). No electronic devices are permitted.

Honor Code. This exam is governed by Princeton's Honor Code. Discussing the contents of this exam before solutions are posted violates the Honor Code.

Please complete the following information:

Name: Ada Lovelace

NetID: alovelace

Exam room: ● McCosh 50 ○ McCosh 60 ○ McCosh 64 ○ Other

Precept:

P01	P01A	P02	P03	P04	P04A	P05	P05A
●	○	○	○	○	○	○	○

P06	P10	P10A	P11	P12
○	○	○	○	○

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

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

Ada Lovelace

Signature

1. **Java keywords. (10 points)**

For each description on the left, write the letter of the best-matching Java keyword on the right. Use each letter at most once.

I	Specifies that there is one variable per class (rather than one variable per object).	**A.** `class`
E	Denotes the default value for an instance variable of a reference type.	**B.** `final`
F	Restricts an instance variable from being accessed directly by client code.	**C.** `import`
J	Refers to the invoking object within an instance method.	**D.** `new`
L	Indicates that a method does not return a value.	**E.** `null`
G	Identifies a method that client code is allowed to call.	**F.** `private`
A	Defines a data type.	**G.** `public`
D	Creates and initializes an object.	**H.** `return`
K	Causes an exception to occur.	**I.** `static`
B	Declares a variable whose value cannot change after initialization.	**J.** `this`
		K. `throw`
		L. `void`

2. **Properties of objects. (10 points)**

Which of the following statements about Java programming are true?

For each statement, fill in exactly one bubble: true or false.

true	*false*	
●	○	A *data type* is a set of values together with operations on those values.
●	○	Each value of a primitive numeric type uses a fixed number of bytes of memory (e.g., 8 bytes for each `double`).
●	○	An instance method can take more than one argument.
○	●	Every class must define exactly one constructor.
○	●	A class can define multiple instance methods but no two instance methods can have the same name.
○	●	An instance method can access the private instance variables of the invoking object, but *not* the private instance variables of another object of the same class.
○	●	If every instance variable in a class is declared `final`, then the data type is *immutable*.
●	○	A class can contain both static methods and instance methods.
○	●	Declaring a local variable with the same name as an instance variable results in a compile-time error.
●	○	Declaring a local variable with the same name as a parameter variable results in a compile-time error.

3. **Using data types. (9 points)**

Assume that the variables `s`, `t`, and `u` are initialized as follows:

```
String s = "AAA";
String t = "BBB";
String u = null;
```

For each Java expression on the left, write the letter of the best-matching value or description from the right. You may use each letter once, more than once, or not at all.

C	`s.charAt(2)`
F	`s.substring(1, s.length())`
I	`s + t.toLowerCase()`
A	`s.compareTo(t) <= 0`
B	`u == s`
G	`s.replace('A', 'B').replace('B', 'A')`
A	`t.charAt(1) == t.charAt(2)`
A	`"AAA".equals(s)`
M	`u.endsWith("A")`

A. `true`

B. `false`

C. `'A'`

D. `'B'`

E. `"A"`

F. `"AA"`

G. `"AAA"`

H. `"AAABBB"`

I. `"AAAbbb"`

J. `"aaabbb"`

K. `"BAA"`

L. `null`

M. *runtime exception*

4. **Data-type design. (9 points)**

Which of the following Java coding practices help make a data type *immutable*?
Fill in the checkbox for each statement that applies.

- [x] Declaring instance variables to be `private`.
- [x] Declaring instance variables to be `final`.
- [x] Declaring instance variables to be of immutable types.
- [ ] Shadowing instance variables.
- [ ] Declaring methods to be `static`.
- [x] *Defensively copying* a mutable constructor argument before storing the copy in an instance variable.
- [x] *Defensively copying* a mutable instance variable before returning the copy.
- [ ] *Overloading* instance methods.
- [ ] Defining a `toString()` method.

5. **Creating data types and debugging. (10 points)**

Here is the API for a data type that records the frequencies of integers from 0 to $n-1$.

`public class Histogram`	
`public Histogram(int n)`	*creates a histogram for values in the range* [0, *n*)
`public void hit(int val)`	*increments the count for* `val`
`public int count(int val)`	*returns the count for* `val`

Fix the following bug-infested implementation so that it conforms to the API. There is a twist: *you may only delete characters*; you may not add code or rearrange code.

For each line of code that contains characters to delete, fill in the corresponding checkbox and strike out the characters to be deleted.

bug

```
■   public java class Histogram {
■       private static final int n;
■       private int[] freq = new int[n];

■       public void Histogram(int n) {
□           this.n = n;
■            int[] freq = new int[n];
        }

■       private static void validate(int val) {
■           if (val <=  0 || val >= n)
■               throw  return new IllegalArgumentException("invalid value");
        }

□       public void hit(int val) {
□           validate(val);
■            return freq[val] += 1;
        }

□       public int count(int val) {
■           validate(int val);
□           return freq[val];
        }
    }
```

6. **Algorithms. (10 points)**

The leftmost column contains an array of 24 integers to be sorted; the rightmost column contains the integers in sorted order; the other columns are the possible contents of the array at some intermediate step of *insertion sort* or *mergesort* (as implemented in the lecture and textbook). By *intermediate step*, we mean at the very end of some iteration of insertion sort or immediately after some call to `merge()` in mergesort. *Hint: think about the properties of insertion sort and mergesort.*

Consider each column independently. Write the letter of the best-matching description under the corresponding column. You may use each letter once, more than once, or not at all.

71	35	22	22	22	35	22
46	42	31	31	31	42	23
35	46	35	35	35	46	24
42	67	42	42	42	71	26
67	71	46	46	46	31	28
31	31	54	54	54	54	29
56	56	56	56	56	56	30
54	54	59	59	59	67	31
91	91	67	67	67	22	35
59	59	71	71	71	59	42
22	22	80	80	80	80	46
80	80	91	91	91	91	54
72	72	23	28	72	28	55
29	29	24	29	29	29	56
30	30	26	30	30	30	58
28	28	28	58	28	72	59
85	85	29	72	85	23	67
58	58	30	85	58	55	69
55	55	55	55	55	58	71
23	23	58	23	23	85	72
24	24	69	24	24	24	80
97	97	72	97	97	26	85
26	26	85	26	26	69	91
69	69	97	69	69	97	97
A	B	C	C	D	E	F

A. Original array

B. Insertion sort only

C. Mergesort only

D. Both insertion sort and mergesort

E. Neither insertion sort nor mergesort

F. Sorted array

7. **Data structures. (12 points)**

(a) Consider the following program:

```
public class Mystery {
    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<Integer>();
        while (!StdIn.isEmpty()) {
            String s = StdIn.readString();
            if      (s.equals("*")) stack.push(stack.pop() * stack.pop());
            else if (s.equals("+")) stack.push(stack.pop() + stack.pop());
            else if (s.equals("-")) stack.push(-stack.pop());
            else stack.push(Integer.parseInt(s));
        }
        StdOut.println(stack.pop());
     }
}
```

Suppose that the program is run independently from scratch on each *standard input* stream on the left.

For each input, write the letter from the right of the best-matching result. You may use each letter once, more than once, or not at all.

H	`1 2 * 3 4 + +`	**A.** 0
		B. 1
D	`1 2 3 - 4 + * +`	**C.** 2
		D. 3
I	`1 2 * 3 + 4 5 6 +`	**E.** 4
		F. 5
J	`+ - 6 7`	**G.** 6
		H. 9
		I. 11
		J. *runtime exception*

(b) Consider the contents of the symbol table `st` at the end of the following code fragment:

```
ST<String, Queue<Integer>> st = new ST<String, Queue<Integer>>();
Queue<Integer> queue = new Queue<Integer>();
queue.enqueue(1);
st.put("A", queue);
queue.enqueue(2);
st.put("B", queue);
queue = new Queue<Integer>();
queue.enqueue(3);
st.put("C", queue);
st.get("A").enqueue(4);
st.put("A", st.get("C"));
queue.enqueue(5);
st.get("A").dequeue();
```

i. Which integers are in the queue returned by `st.get("A")`?
Fill in the checkbox of each integer in the queue.

0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ■

ii. Which integers are in the queue returned by `st.get("B")`?
Fill in the checkbox of each integer in the queue.

0 ☐ 1 ■ 2 ■ 3 ☐ 4 ■ 5 ☐

iii. Which integers are in the queue returned by `st.get("C")`?
Fill in the checkbox of each integer in the queue.

0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ■

8. **Theory of computing. (10 points)**

You are in the final round of a job interview at a post-LLM tech startup. Your interviewer asks you to fact-check the following LLM-generated claims about theoretical computer science, classifying each as

A. *learned from training* (known to be mathematically true)

B. *hallucination* (known to be mathematically false)

C. *science fiction* (not yet known to be either true or false)

For each computer science claim, fill in the best-matching bubble.

A	B	C	
○	●	○	*ChatGPT 1.0* reports $2 + 2 = 5$.
○	●	○	*Gemini* describes a new programming language for Android devices that can compute more mathematical functions than Java.
●	○	○	*Qwen* claims that a quantum computer is equivalent to an ordinary computer in terms of which problems are computable.
○	●	○	*Copilot* produces a Java program that determines whether any two Python functions always produce the same output on the same input.
○	○	●	*Grok* engineers a physically realizable computing device that harnesses black holes to solve the halting problem.
○	●	○	*DeepSeek* produces a computational problem that can be solved on a TOY machine, but not on a Turing machine.
○	●	○	*Apple Intelligence* designs a polynomial-time algorithm that determines whether any Turing machine halts on a given input tape. This revolutionary algorithm runs only on iOS.
●	○	○	*Mistral* writes a Java program that takes a positive integer n as input and outputs the prime factorization of n.
○	●	○	*Perplexity* develops a debugger that can determine whether any given line of Java code is ever executed.
●	○	○	*Llama* proclaims that every computational problem solvable by some DFA is also solvable by some Turing machine.
○	●	○	*ChatGPT 4.5* asserts that every computational problem solvable by some Turing machine is also solvable by some DFA.

9. **Machine learning. (6 points)**

For each application on the left, write the letter of the most appropriate machine-learning paradigm from the right. You may use each letter once, more than once, or not at all.

B	Predict whether a lung scan is normal or abnormal using a dataset of lung scans already labeled by doctors.
C	Group customers with similar shopping habits, without being given customer-category labels.
A	Train a robot arm to pick up objects by trying different movements and receiving a reward when it succeeds.
B	Translate an English sentence into Spanish using many examples of English sentences paired with Spanish translations.
B	Predict blood pressure from age, BMI, and blood sugar level using prior patients' records.
C	Detect unusual credit-card transactions in an unlabeled dataset.

A. Reinforcement learning

B. Supervised learning

C. Unsupervised learning

10. **Deep learning. (8 points)**

For each deep-learning description on the left, write the letter of the best-matching term from the right. Use each letter exactly once.

[B]	A mathematical function loosely inspired by biological neurons; it computes a weighted sum of its inputs and applies an activation function.
[H]	A learnable value that determines the strength of a connection between two neurons.
[A]	A function applied to a neuron's weighted sum to determine its output.
[E]	The process of passing data through a neural network to produce a prediction.
[C]	The process of computing how each parameter should be adjusted to reduce the model's error.
[F]	A function that measures how wrong the model's prediction is.
[G]	The way neurons and layers are arranged and connected in a neural network.
[D]	A neural-network architecture especially useful for image data.

A. Activation function

B. Artificial neuron

C. Backpropagation

D. Convolutional neural network

E. Forward pass

F. Loss function

G. Neural-network architecture

H. Weight

11. **TOY machine and number representation. (10 points)**

For each description on the left, write the letter of the best-matching integer from the right. Use each letter once, more than once, or not at all.

F	Number of TOY registers (including register 0).
C	Number of hexadecimal digits needed to specify a TOY main memory address.
F	Number of *bits* that the TOY *instruction register* stores.
D	Number of *bytes* in a Java `int`.
D	Number of 1s in the binary representation of the decimal integer 180.
G	Number of 1s in the binary representation of the decimal integer −1. *Assume 32-bit two's complement integer.*
F	Number of 1s in the binary representation of the decimal integer −67. *Assume 18-bit two's complement integer.*
I	Number of distinct negative values representable by an 8-bit two's complement integer.
L	Number of distinct TOY instructions whose opcode is 0.
A	Value produced by evaluating the Java expression: `256 * 256 * 256 * 256`

A. 0

B. 2^0 (1)

C. 2^1 (2)

D. 2^2 (4)

E. 2^3 (8)

F. 2^4 (16)

G. 2^5 (32)

H. 2^6 (64)

I. 2^7 (128)

J. 2^8 (256)

K. 2^{10} (1,024)

L. 2^{12} (4,096)

M. 2^{15} (32,768)

N. 2^{16} (65,536)

O. 2^{31}

P. 2^{32}

TOY REFERENCE CARD

INSTRUCTION FORMATS

```
              | . . . . | . . . . | . . . . | . . . . |
  Format RR:  | opcode  |    d    |    s    |    t    |  (1-6, 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
```

```
 16 16-bit registers:        R[0]  to R[F]
256 16-bit memory locations: M[00] to M[FF]
  1  8-bit program counter:  PC
```

R[0] always reads as 0000.
Loads from M[FF] come from stdin.
Stores to M[FF] go to stdout.

12. **TOY programming. (12 points)**

Set the program counter to `10` and run the following TOY program for different instructions in memory location `12`. Upon termination, which value is stored in memory location `17`?

```
10: 8A15
11: 8B16
12: [substitute instruction below]
13: 9C17
14: 0000
15: 6734
16: 0078
17: 0000   // result
```

(a) Assume `1CAB` is the instruction in memory location `12`.

6	7	A	C

(b) Assume `2C0B` is the instruction in memory location `12`.

F	F	8	8

(c) Assume `7CAB` is the instruction in memory location `12`.

0	0	A	B

Set the program counter to `10`, initialize standard input as described below, and run the following TOY program. Which value is printed to standard output?

```
10: 7100   R[1] = 0000
11: 8FFF   read R[F]
12: 9F15   M[15] = R[F]
13: 82FF   read R[2]
14: 1112   R[1] = R[1] + R[2]
15: 1112   R[1] = R[1] + R[2]
16: 91FF   write R[1]
17: 0000   halt
```

(d) Assume standard input contains: `1112 4112`.

8	2	2	4

(e) Assume standard input contains: `C011 C011 1112 1112`.

E	2	3	5

13. **Digital circuits. (9 points)**

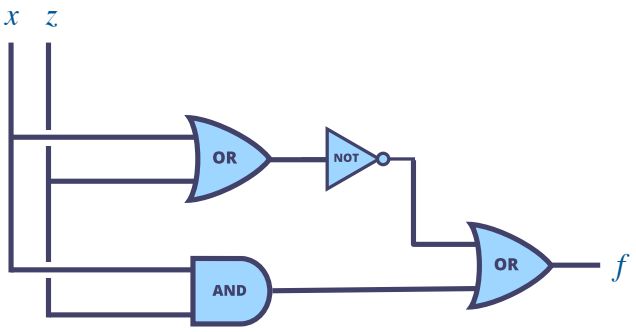
Consider a boolean function f of three variables x, y, and z. The function is 1 if the 3-bit string xyz is a *palindrome* (reads the same forward and backward), and 0 otherwise. For example, 000 is a palindrome, but 001 is not.

Which of the following represent the 3-bit palindrome function f?
Fill in the corresponding bubbles.

yes ● *no* ○

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

yes ● *no* ○



yes ○ *no* ●

$f = x'y'z' + x'yz' + xy'z$

yes ● *no* ○

$f = xz + x'z'$

yes ● *no* ○

$f = (xz' + x'z)'$

yes ● *no* ○

$f = (x \text{ xor } z)'$

14. **Java programming. (10 points)**

(a) Write a Java function that takes a `boolean[]` array of length $n > 0$ as an argument, and returns `true` if the array is a *palindrome* (equal to its reverse), and `false` otherwise.

For example, the array `[false, true, true, true, true, false]` is a palindrome, but `[false, false, true]` is not.

Write your code neatly and legibly in the space provided. Your program will be graded on correctness, clarity, and efficiency.

```
public static boolean isPalindrome(boolean[] a) {
    int n = a.length;
    for (int i = 0; i < n/2; i++) {
        if (a[i] != a[n-i-1]) return false;
    }
    return true;
}
```

(b) What is the order of growth of the worst-case running time of your `isPalindrome()` implementation in (a) as a function of n, where n is the length of the array?

○ $\Theta(\log n)$ ● $\Theta(n)$ ○ $\Theta(n \log n)$ ○ $\Theta(n^2)$ ○ $\Theta(n^3)$ ○ $\Theta(2^n)$

15. **Programming assignments. (15 points)**

Write a program `Kgram` that takes a string s of length n and an integer $0 < k \leq n$ as command-line arguments and prints the maximum frequency of any k-gram in s.

A k-gram is a substring of s consisting of k consecutive characters (with no wraparound). For example, if s = `"ABABAABABACDAB"` and k = 3, then the program should print `4`, because the k-gram `"ABA"` appears 4 times in s.

Write your code neatly and legibly in the space provided. Your program will be graded on correctness, clarity, and efficiency.

```
public class Kgram {
    public static void main(String[] args) {
        String s = args[0];
        int k = Integer.parseInt(args[1]);

        // compute frequency of each k-gram in s
        ST<String, Integer> st = new ST<String, Integer>();
        for (int i = 0; i <= s.length() - k; i++) {
            String kgram = s.substring(i, i+k);
            if (!st.contains(kgram)) st.put(kgram, 1);
            else                     st.put(kgram, 1 + st.get(kgram));
        }

        // compute and print maximum frequency
        int max = 0;
        for (String kgram : st.keys()) {
            max = Math.max(max, st.get(kgram))
        }
        StdOut.println(max);
    }
}
```

This page is intentionally left blank. Feel free to use for scratch work.