Name: NetID:

Final

CS 2110, 21 December 2021, 2:00pm EST

	1	2	3	4	5	6	7	Total
Question	Name	Loop Invs	Short answer	OO	Graphs	Data structures	Concur-rency	
Max	1	10	21	14	13	31	10	100

- This exam is open-note. Collaboration is strictly forbidden. Do not share any portion of your exam with anyone. Do not discuss the exam with anyone besides the instructors until after grades have been released.
- This exam is designed to take no more than **120 minutes**. If you have SDS accommodations, allocate additional time proportional to this duration (e.g. those granted 50% extra time should plan to spend 180 minutes on the exam). Do not spend more than this nominal time working on the exam problems. It is *your responsibility* to manage your time so that you can print, work for the nominal time, scan, and upload by the deadline while accommodating conflicts and technical hiccups.
- Submit to Gradescope *before* the end of your submission window (as shown on the exam distribution page). Aim to submit early, and contact the instructors immediately if you encounter major technical difficulties.
- **Write your answers in the boxes or spaces provided**. Ambiguous answers will be considered incorrect. You should be able to fit your answers easily into the allocated space.
- In some places we have abbreviated or condensed code to reduce the number of pages that must be printed for the exam. In others, code has been obfuscated to test your understanding of the language. This does not mean that it's good style.
- Policy details and submission instructions are available on Canvas. Good luck!

Academic Integrity Statement: I pledge that I have neither given nor received unauthorized aid on this exam. I will not discuss the exam with anyone in this course who has not yet taken it.

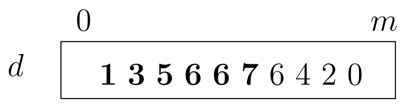
(signature)

1. Name (1 point)

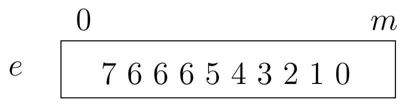
Is your name at the top of this page and your NetID at the top of pages 1..10 correct? If either is incorrect, or if you are missing pages, do not proceed; Contact the instructors immediately! Sign the Academic Integrity Statement before submitting your exam.

2. Loop Invariants (10 points)

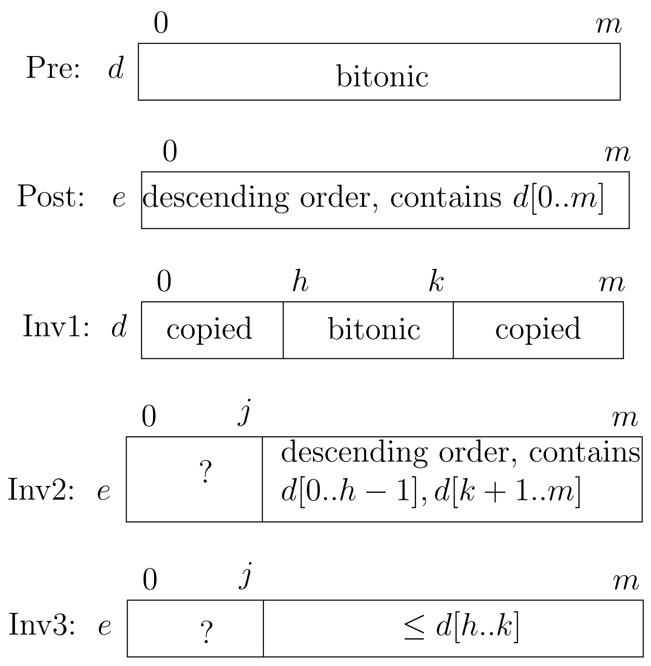
A sequence of integers is *bitonic* if it is first ascending and then descending. The following array is bitonic: The bolded values are ascending, rest are descending.



We write a while-loop that copies bitonic segment $d[0..m]$ into a new array e in *descending* order. Example: For the bitonic sequence $d[0..9]$ given above, $e[0..9]$ will be:



(Only part of array d is being sorted, not the whole array.) To the right are the precondition, postcondition, and invariant for this problem. The invariant has three parts. Note that values are put in $e[0..m]$ starting at the end, starting with $e[m]$.


Pre: d
0
m
bitonic
Post: e
0
m
descending order, contains d[0..m]
Inv1: d
0
h
k
m
copied
bitonic
copied
Inv2: e
0
j
m
?
descending order, contains d[0..h − 1], d[k + 1..m]
Inv3: e
0
j
m
?
≤ d[h..k]

(a) 3 points In the box to the right, write the initialization for the while-loop. Do not declare any variables.

h= 0;
k= m;
j= m;

(b) 2 points The loop has the form **while** (B) *repetend*. To the right, write condition B. Do NOT write anything else.

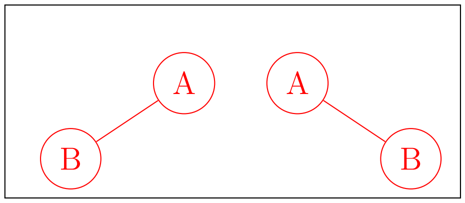
$h \le k$ or $0 \le j$ or equivalent

(c) 5 points In the box to the right, write the repetend.

```
if ($d[h] < d[k]$)    // could be <=
      { $e[j]$= $d[h]$; $h$= $h+1$; }
else { $e[j]$= $d[k]$; $k$= $k-1$; }
j= j - 1;
```

3. Short Answer (21 points)

(a) 3 points. Show that you cannot uniquely construct a binary tree from its preorder and postorder traversals by drawing to the right two *different* binary trees that have the same preorder and postorder traversals.
Hint: Make the trees as small as possible.



(b) 4 points. Consider the following method declaration:

```
void needHelp(Number[] nums) {...}
```

Which of the argument types on the right can be passed to method `needHelp()`? Write the letter for each allowed type in the box below, or write "None" if none are allowed. Note that `Integer` is a subclass of `Number`.

M, N

```
M. Integer[]
N. Number[]
O. Object[]
P. ArrayList<Number>
Q. ArrayList<Integer>
R. Object
```

(c) 4 points. Consider hashing with linear probing using a 6-element array `c[0..5]` to maintain a set of Integers. Initially, `c` contains no elements and is thus: `[null, null, null, null, null, null]`. The hash function to be used is the integer itself: `hashcode(i) = i`. Draw array `c` after these values have been added: 10, 11, 18, 12. Do not be concerned with the load factor.

[18, 12, null, null, 10, 11]

(d) 6 points. Consider the following snippet of Swing code for changing some text in a window when a button is clicked:

```
JLabel txt = ...;
JButton b = ...;
b.addActionListener(v -> txt.setText("Action: " + v.getActionCommand()));
```

1. What variable or parameter corresponds to the event *source*? b
2. What variable or parameter corresponds to the *event*? v
3. What *interface* does the anonymous function implement? ActionListener

(e) 2 points. Consider an array `people` of objects of class `Person`. Each person has a last name and a first name. We want to sort based on last names, but if two people have the same last name, they should be ordered by their first name. For example, these Persons are in order:
Mary Jones, Jack Smith, Judy Smith.

We do this with the sequence of two statements shown above to the right.

Between insertion-sort and selection-sort, only one is appropriate to use here. Which one, and why?

```
Sort array people based on
     their first names;
Sort array people based on
     their last names;
```

insertion-sort
it must be stable

(f) 2 points. Consider a graph with n nodes and e edges. Algorithms B and C perform the same task but with different asymptotic complexity. Algorithm B's time complexity is in $O(n^2 \lg(n))$ while C's is in $O(e\ n)$. Which algorithm would you expect to run faster for *large, dense* graphs? Write its name (and nothing else) in the box.

B

4. Object-Oriented Programming (14 points)

(a) 7 points You are working on the naughty list for Santa. You want to keep track of which people had a conflict, and you want to sort people using that information for Santa's final decisions.
This will be done using class `Person`. Two of its fields are shown to the right.
Example: `Larry` had 9 conflicts with `Barry`, so the pair (object for Larry, 9) occurs in field `conflictWith` in the object for Barry.

```
public class Person
         implements Comparable<Person> {
 /** The id of this person */
 public int pId;

 /** A pair (i, val) is in conflictWith
   * if person i had a conflict with this
   * person val times. */
 public HashMap<Person, Integer>
                          conflictWith;
```

Complete the parts of the class indicated in the boxes below. Note that many other features of this class are missing.

```
/** Constructor: An instance with pId id that has no conflicts with anyone*/
public Person(int id) {
```

pId= id;
conflictWith= new HashMap<Person, Integer>();

```
 }

/** = number of times this had conflicts with every person in conflictWith. */
  public int allConflicts() {...}

/** Return -1 or 1 depending on whether this persons's allConflicts()
    * are < or > p's allConflicts(). If this Person's allConflicts() is
    * equal to p's allConflicts(), return -1, 0, or 1 depending on
    * whether this persons's pId is <, =, or > p's pID.
 * Precondition: p is not null */
public @Override int compareTo(Person p) {
```

```
    int val= Integer.compare(allConflicts(), p.allConflicts());
    if (val == 0) return Integer.compare(pID, p.pID);
    return val;
```

```
}
```

(b) 7 points In discussing concurrency, we introduced class `ArrayQueue` to maintain a bounded queue in an array. Part of it is shown to the right. Make it iterable by completing methods `hasNext` and `next` of inner class `BQueueIterator`, below.

Note: the values are in successive elements of array `q` starting in `q[h]` and with wrap-around: After the value in `q[q.length-1]` comes the value in `q[0]`.

Don't introduce new fields, but of course you can have local variables.

```
class ArrayQueue<E> implements Iterable<E> {
  /** Elements of the queue are q[h], q[h+1],
    * ... q[h+n-1] (all taken mod q.length) */
  protected E[] q; //
  protected int n   // size of queue
  protected int h;  // 0 <= h < q.length
     ...      }

  public Iterator<E> iterator() {
     return new BQueueIterator();
  }
```

```
/** An instance is an iterator over this bounded queue. */
private class BQueueIterator implements Iterator<E> {
    /** If k = n, all values have been enumerated.
      * If k < n, q[(h+k) % q.length] is the next one to be enumerated */
    private int k= 0;

    public BQueueIterator() {} // Constructor: an iterator over this queue

    /** = "there is another value to enumerate." */
    public boolean hasNext() {
```

```
                return k < n;
```

```
    }

    /** Return the next value to enumerate, thus enumerating it.<br>
     * Throw a NoSuchElementException if there is no next element. */
    public E next() {
```

```
        if (!hasNext()) throw new NoSuchElementException();

        E val= q[(h+k) % q.length];    OR    k= k+1;
        k= k + 1;                            return q[(h+k-1) % q.length];
        return val;
```

```
 } }
```

5. Graphs(13 points)

The answer to each of the first 3 questions is one of these algorithms: Kruskal, BFS, DFS. Write the answer in the box to the right of each question. Don't write anything else in the box.

(a) 2 points Consider a graph with all edge weights 1. Dijkstra's shortest path algorithm to find the distance from one node to all others visits the nodes in the same order as one of the algorithms. Which one?

BFS

(b) 2 points A map (an undirected graph) of Cornell gives the paths from each building to neighboring buildings. Which algorithm is best suited to determining the smallest number of buildings one must "visit" in walking from Gates Hall to Willard Straight?

BFS

(c) 2 points The US has finally decided to rebuild its train system. It has built an undirected graph whose nodes are cities and whose edges are possible routes between cities; the value on an edge is the cost of building a railroad along that route. It wants to use this graph to determine which railroads to build along which edges. Which algorithm would you use to determine the minimum cost to build a train system?

Kruskal

(d) 7 points To the right are two properties of a spanning tree of an undirected connected graph G. Based on (1), we wrote this abstract algorithm for creating a spanning tree.

(A1) While G has a cycle, throw out one edge of a cycle.

(1) A spanning tree of G is a maximal set of edges that contains no cycle.

(2) A spanning tree of G is a minimal set of edges that connects all nodes.

(d1) Below, give the abstract algorithm A2 that results from using property (2).

Start with all nodes and no edges of G.
While the nodes are not all connected, add an edge that connects two unconnected components. (You can also say "that does not introduce a cycle.")

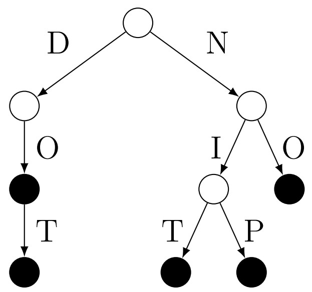
(d2) Each of the abstract algorithms (A1) and (A2) has a loop. Write down the number of iterations each loop takes (as a function of the number n of nodes and the number e of edges).

A1 iterations	A2 iterations
e - (n-1)	n-1

6. Data structures (31 Points)

(a) Data structure for words (15 points)

The tree to the right illustrates a neat data structure for maintaining a set of words in the English language. Assume all letters are in upper case. This tree contains the words "DO", "DOT", "NIT", "NIP", and "NO". A word in the set is found by reading the characters on a path going down to a black node.



prefix tree

Each node has a 26-element array c[0..25] for the children, each of which represents an uppercase letter and is either a (pointer to) a child or null. Each node has a boolean field to say whether it ends a word or not (black or white in the tree to the right).

(a1) 3 points For the children of a node, `c[0]` is for character 'A', `c[1]` for 'B', etc. Let char variable `ch` contain a uppercase letter. In the box fill in the **index** so that the array reference returns the value in `c` corresponding to character `ch`:

c[ch – 'A']

(a2) 12 points Below, give the tightest expected and worst-case Big-O time complexity bounds, in terms of n and s, for searching for a word of length `n` in a set of Strings of size `s` when the set is implemented as (1) a prefix tree, as above, (2) a balanced Binary Search Tree (BST) of Strings, and (3) a HashSet<String>. Remember: *The cost of determining whether one String equals another, as well as the cost of computing a String's hash code, depends on the length of the String.*

Prefix tree as above. Expected:	O(n)	Worst-case:	O(n)
Balanced BST. Expected:	O(n log s)	Worst-case:	O(n log s)
HashSet<String>. Expected:	O(n)	Worst-case:	O(s*n)

(b) Complexity (10 points) Consider an undirected connected graph with n nodes and e edges. Each node is an object of class `Node`, and the neighbors of a node are given as a `LinkedList<Node>`. In method `trav` on the next page, each statement has a label on it, so we can talk about it.

Consider the call `trav(node, new HashSet<Node>())` of method `trav`. To the right of each statement in the body of `trav`, write the total number of times it is executed during execution of this call, in terms of n and e. For 'd', write the number of times the for-each statement is encountered, not how many iterations it does.

```
/** Visit all nodes reachable from u along unvisited paths
  * Precondition. visit contains all visited nodes. */
static void trav(Node u, Set<Node> visit) {

    a: if (visit.contains(u))             1 + 2e (we accept 2e, 1 + e, e)

    b:     return;                        1 + 2e - n (we accept e - n)

    c: visit.add(u);                      n

    d: for (Node node : u.neighbors)      n

    e:    trav(node, visit);              2e
```

(c) Heaps (6 points) Array segment $c[0..s-1]$ is supposed to contain a heap: a tree with no holes that satisfies certain properties, as discussed in the lecture on priority queues and heaps. Write the body of the following function. (Recursion is not necessary and makes it harder.)

Hint: Our solution begins with this statement: `boolean minHeap= c[0] < c[1];`.

```
/** Return -1, 0, or 1 depending on whether c[0..s-1] is a min-heap,
  * not a heap, or a max-heap.
  * Precondition: s > 1. The elements of c[0..s-1] are all different.
  *               c[0..s-1] represents a complete tree. */
static int isHeap(int[] c, int s) {
```

```
boolean minHeap= c[0] < c[1];
for (int k= 1; k < s; k= k+1) {                    // Note. Many people tried to compare c[k]
    if (minHeap != c[(k-1)/2] < c[k]) return 0;   // to its children instead of its parent.
}                                                   // That's harder and much more work!
return minHeap ? -1 : 1;
```

```
}
```

7. Concurrency (10 Points)

(a) Deadlock (4 points).
The two threads to the right are run concurrently, sharing Sets `setX`, and `setY`. Is it possible for these two threads to deadlock? Explain why or why not.

```
Thread 1: while (true) {
    synchronized(setX) {
      synchronized(setY) {
        while (!setY.isEmpty()) {
          setX.add(setY.remove(0));
 } } } }

Thread 2: while (true) {
   synchronized(setY) {
     if (!setY.isEmpty()) setY.remove(0);
   }
   synchronized(setX) set.add("CS2110");
 }
```

No, deadlock is not possible.
Thread 2 does not try to lock both lists at once, and neither thread blocks or spins except to acquire a lock

(b) Synchronization (6 points).
Consider the following class, an instance `man` of which is shared among threads T1, T2, and T3:

```
class Manifest {
  boolean flag;

  synchronized void edit() throws InterruptedException {
    System.out.print("X ");
    Thread.sleep(2000);  // Sleep for 2 seconds
    System.out.print("Y ");
    while (!flag) { wait(); }  // Wait for signal
    flag= false;
    System.out.print("Z ");
} }
```

First, T1 calls `man.edit()`. Then, 3 seconds later, T2 calls `man.edit()`. 1 second later, T3 calls `man.edit()` and blocks. So far, the program has produced the output "X Y X ". Answer the following questions about this point in time (each answer is some combination of "T1", "T2", and "T3", or the word "None"):

Which thread(s) are currently on the lockist? T3

Which thread(s) are currently on the waitlist? T1

Which thread(s) currently hold the lock for `man`? T2