This is just for practice.
Your final exam does not necessarily look like this.

- This is a closed-book exam. You are not allowed to use any printed/written material, laptops/cell-phones. Please turn off your cell phones and put them in your bags.
- Manage your time. We start the exam at 9:00 and end the exam at 12:00. You have 3 hours.
- There will be 16 pages including this cover page. Write your answers in the provided boxes. In the unlikely case that you need more space, use the blank page at the end; if you do so, indicate it on this page (the first page of the exam).
- In the unlikely case that you find the exam too long/hard, do not panic. The marks will be scaled so that the highest mark gets the full mark.
- There are more important things in life than this exam. So, relax and smile (but still manage your time).
1. Short Answer

Provide your short answers in the provided boxes. There is no need to justify your answers.

1. True or False: An abstract class can implement an interface.

2. True or False: A private method of class A can be called in any class inherited from A.

3. True or False: 2019n ∈ O(n log n)

4. True or False: insertion-sort runs in O(n log n) in the best case.

5. True or false: The height of a heap depends on the order of insertions.

6. Consider the following pseudocode:

   ```c
   void bar ( int n)
   {
   int i = 1;
   int res = 1;
   while ( i < n)
   {
   for ( int j=1; j < n; j++)
   for ( int k=j/2; k < j; k++)
   res *= 2;
   i = i+2;
   }
   }
   ```

   What is the worst-case running time of \( \text{bar}(n) \)?

   Express your answer using \( O \)-notation in terms of \( n \), and be as precise as possible. No justification is needed.

7. Assume you have \( n \) integers in the range \((0, n^2)\). These integers are all square roots of other integers. Indicate whether it is possible to sort these numbers in \( O(n) \) or not. Justify your answer in a few sentences (be precise).

8. True or false: Given a node \( x \) in a binary search tree of size \( n \), deleting the parent of \( x \) takes \( O(1) \) time in the best case.

9. True or False: If we implement a stack using an array, we should know the maximum size of the stack when initiating it.

10. Assume a dictionary of \( n \) KVPs is implemented using a sorted linked list.

    True or False: it is possible to delete an item in \( O(\log n) \).

11. True or False: there is a unique heap formed by a given set of keys.

12. True or False: we can construct a tree given its in-order and pre-order traversal.
13. True or False: a b-tree of size \( n \) with order \( d \) tree can have height \( O(1) \) in the best case.

14. In the space on the right, draw a binary tree formed by keys \( a, b, c, d \) that has the same post-order and in-order traversal (in other words, if we print keys in the post-order traversal and in the in-order traversal, the same strings should be printed).

15. True or False: When we re-hash a hash table, we need to update the hash function.

16. True or False: Assume a universe formed by a subset of integers. If the universe is congested around a number \( x \) (that is, for majority of numbers like \( y \) the value of \( |y - x| \) is a small number) then hash tables are not useful for maintaining dictionary operations.

17. True or False: Heaps are useful when we need to support the \texttt{predecessor}(k) query.

18. Assume we need to sort a set of double numbers in the range \([0, 100]\). Indicate the most suitable sorting algorithms among the followings:
   (a) quick-sort   (b) counting-sort   (c) counting sort.

19. Assume we need to support the following queries for a dictionary: \texttt{search}(k), \texttt{insert}(k), \texttt{delete}(k).
   Indicate the most appropriate data structure.
   (a) binary search tree   (b) hash table   (c) heap

20. True or False: it is possible to merge two heaps stored in two arrays of size \( n \) to form a heap of size \( 2n \) in time \( O(n) \).

21. True or False: The number of edges in a tree of size \( n \) can be \( O(1) \).

22. Indicate whether the graph \( G \) below is planar. Justify your answer.

23. Indicate whether the graph \( G \) below is bipartite. Justify your answer.

24. Color the graph \( G \) below using three colors G (green), R (red), and B (blue).
2. Recursive Algorithms

Given a string $S$, write an algorithm that forms a palindrome string from $S$ that mirrors the second half of $S$. For example, if $S$ is initially *LifeIsGood*, it should become *dooGssGood*. Your algorithm should be recursive, that is, you cannot use any loop.

3. Stacks & Queues

Assume you have a stack with items $a_1, a_2, \ldots, a_n$ in it in the same order, that is, $a_1$ is at the bottom, $a_2$ is on top of it, and so on (so $a_n$ is on the top of the stack). We want to build a queue with the same items so that $a_1$, $a_2$ is after it, and so on so that $a_n$ is the last item (the queue looks like $a_n \rightarrow a_{n-1} \rightarrow \ldots \rightarrow a_1$). Explain how to do this with only stack and queue operations. You can use additional stacks and queues but you cannot use an array or any other data structure (and you do not know how the stack and list are implemented).
4. Data Structure Applications

1. Given a string $S$ and an initially empty stack $st$, indicate how we reverse $S$. Your algorithm should not use any data structure except for $st$ (e.g., no additional array is permitted).

2. Given a main stack $st$ that contains a set of keys and an initially empty stack $st'$, indicate whether we can reverse the order of items in $st$ without using any data structure other than $st$ and $st'$. As an example, if $st$ has keys $a, b, c$ (in the same order), it should have $c, b, a$ after the operation (and $st'$ stays empty before and after).
5. Sorting Algorithms

1. Consider quick-sort with a pivot selection strategy in which the pivot is always either the largest item in the input or the median (the middle item). Write down a recursion for the worst-case time complexity of the algorithm. Write down the solution to the recursion using big-O notation.

2. Consider a variant of quick-sort that stops recursively calling itself when the input has length less than 1000. Instead, it uses insertion-sort to sort the input subarray with length less than 1000. Write down the recursion for this variant of quick-sort. What is the best-case time complexity of this algorithm?
6. Hash Tables

Consider the following hash table of size 7 with hash-function \( h_1(k) = k \mod 7 \). For the following questions, it suffices to show the final tables.

1. Add the following keys to the heap and show the final table. Assume we use chaining to resolve collisions. The operations are \( \langle \text{insert}(46), \text{insert}(78), \text{insert}(11), \text{insert}(14), \text{delete}(46) \rangle \).

2. Perform the following operations on the original table. Assume we use linear probing to resolve collisions. The operations are \( \langle \text{insert}(4), \text{insert}(14), \text{delete}(24), \text{delete}(49), \text{insert}(20) \rangle \).

3. Perform the following operations on the original table. Assume we use double-hashing with \( h_2(k) = \lfloor k/7 \rfloor \mod 7 \) to resolve collisions. The operations are \( \langle \text{insert}(37), \text{insert}(15), \text{insert}(23) \rangle \).
7. More Hashing

1. Consider the following hash table of size 7 which is maintained by Cuckoo hashing with hash-functions $h_1(k) = k \mod 7$ and $h_2(k) = k^2 \mod 7$. Apply the following operations & show your work. The operations are $\langle \text{insert}(6), \text{insert}(10), \text{insert}(20) \rangle$.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>21</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

2. Consider a hash function $H$ which maps items from a universe $U$ to indices $\{0, 1, \ldots, M - 1\}$ of a hash table of size $M$. Let $f(i)$ indicate the number of items from $U$ which are mapped to index $i$. Assume for any two indices $i$ and $j$, we have $f(i) = f(j)$ as long as $i, j \neq 0$ (we cannot make any statement for index 0). Indicate whether dictionary operations can be supported in $O(1)$ using this hash function.
8. Binary Trees

1. Apply the following operations (in the same order) on the Binary Search Tree below. It suffices to draw the final tree. The operations are `insert(4), delete(3)`.

   ![Binary Search Tree Diagram]

3. Describe an algorithm that reports the node that is ‘furthest’ from the root of a binary tree (the node with the maximum depth). For example, for the tree on the right, node h should be reported. The tree is given with a pointer to its root, and you can refer to standard binary tree operations like ‘left’ and ‘right’ queries. You can describe algorithm in English words or Java code.
9. 2-3 Trees

Consider the following 2-3 tree.

1. Draw the tree when after we apply the operation \textit{insert(10)} on it.

2. Draw the original tree after we apply the operation \textit{delete(14)} on it.
B-Trees

Consider the following B-tree with $d = 2$.

1. Draw the tree when after we apply the operation $\text{insert}(7)$ on it.

2. Draw the original tree after we apply the operation $\text{delete}(69)$ on it.
10. Heaps

1. Consider a heap stored in array \( A = [18, 11, 5, 9, 7, 2, 4] \). Draw the resulting tree when we apply the operation \( \text{insert}(12) \).

2. Consider a heap stored in array \( B = [15, 14, 13, 5, 2, 10, 6] \). Show the resulting tree when we apply the operation \( \text{extractMax}() \).

3. Indicate whether the following statement is correct or not. Justify your answer.
   In a heap of size \( n \), \( \text{insert}(k) \) takes \( O(1) \) time in the best case.
11. More Heaps

In the class, we learned how to form a max-heap (heapify) an arbitrary array in $O(n)$.

1. Trace the algorithm to heapify a tree stored in the following array. $A = [1, 2, 3, 7, 5, 6, 8, 10]$.

2. Indicate whether the following algorithm correctly heapifies an input array $B$ or not. You should either provide an example for which the algorithm does not heapify $B$ or prove that the algorithm indeed correctly heapifies the input.

```java
public void buildHeap() {
    for (int i = 1; i < n; i++)
        reheapUp(B, i);
}
```
12. Minimum Spanning Trees

1. Trace the Kruskal’s algorithm to find the minimum-spanning-tree of the following graph. You should write down the edges in the MST in the same order they were added to the tree. Indicate each edge with its two endpoints.

2. Trace the initial steps of the Prim’s algorithm for finding the minimum-spanning-tree of the following graph. You should write down the first three edges that are added to the tree and the state of the heap before adding the fourth edge. Indicate each edge with its two endpoints.

3. Assume a graph the input graph to the Prim’s algorithm is a tree. Explain what the time complexity of the algorithm is in this case. You should write down the time complexity, using big-O notation, in terms of $n$, the number of vertices.