Written homework provides an excellent framework for achieving the goals of obtaining a working knowledge of data structures, perfecting programming skills, and developing critical thinking strategies to aid the design and evaluation of algorithms. Since programming has a high overhead in terms of program entry and debugging, all important topics in this course cannot be covered via programming projects. Written homework exercises allow students to learn important material without a high time investment. Although the point value is low, the benefits are great. You can perfect your programming skills without spending hours at the computer and can get feedback on your thinking skills from your study partners. Students that consistently do quality homework, have far superior test scores. Because assignments are done as a group and any questions are discussed in class or during office hours, written solutions to the homework will not be provided.

Note, these exercises may be done in groups of one, two, or three. If more than one person is involved, list all the names on one set of answers. Turn in only one copy of the assignment for all group members. Groups may change throughout the semester. Answers should not be compared with others not in your group. You will learn much more by working in a group than you will learning working by yourself.

Note that when you have a max heap, deleting the max is efficient, but deleting the minimum is not allowed. We talk about max and min heaps – but they are two different kinds of heaps, (usually) not one heap that does both.

1. This question involves the coding of the solution to a game with teddy bears. The game starts when I give you some bears. You can then give back some bears, but you must follow these rules (where n is the number of bears that you have):
   • If n is even, then you MAY give back exactly n/2 bears.
   • If n is divisible by 3 or 4, then you MAY multiply the last two digits of n and give back this many bears. (By the way, the last digit of n is n%10, and the next-to-last digit is ((n%100)/10).
   • If n is divisible by 5, then you MAY give back exactly 42 bears.
   • The goal of the game is to end up with EXACTLY 42 bears.
   For example, suppose that you start with 250 bears. Then you could make these moves:
     --Start with 250 bears.
     --Since 250 is divisible by 5, you may return 42 of the bears, leaving you with 208 bears.
     --Since 208 is even, you may return half of the bears, leaving you with 104 bears.
     --Since 104 is even, you may return half of the bears, leaving you with 52 bears.
     --Since 52 is divisible by 4, you may multiply the last two digits (resulting in 10) and return these 10 bears. This leaves you with 42 bears.
     --You have reached the goal!

   Write a recursive method to meet this specification. Note that you are trying to find if there is any way you can win. You cannot just try one thing and see if you win.

   ```java
   public static boolean bears(int n)
   // Postcondition: A true return value means that it is possible to win
   // the bear game by starting with n bears. A false return value means that
   // it is not possible to win the bear game by starting with n bears.
   // Examples:
   // bear(250) is true (as shown above)
   // bear(42) is true
   // bear(84) is true
   // bear(53) is false
   // bear(41) is false
   // Hint: To test whether n is even, use the expression ((n % 2) == 0).
   ```

2. Given the following max heap (stored as an array), show what the heap looks like after the largest element is removed and the heap is reformed. Note, the definition requires that after deletion from
a heap, the tree has the following property: the array representation of the tree has no empty slots except at the end of the array.

3. Assume you have the following minheap implemented as an array (complete tree) in which the array contains [4, 7, 9, 10, 12, 13, 11]. (Verify that this is a min heap.) Show the minheap following each of the operations: insert(8), insert(5), delete().

4. What feature of heaps allows them to be efficiently implemented using a partially filled array?
   A. Heaps are binary search trees.
   B. Heaps are complete binary trees.
   C. Heaps are full binary trees.
   D. Heaps contain only integer data.

5. If a max heap is implemented using a partially filled array called data, and the array contains n elements (n > 0), where is the entry with the greatest value?
   A. data[0]
   B. data[n-1]
   C. data[n]
   D. data[2*n + 1]
   E. data[2*n + 2]

6. Select the true statement about the worst-case time for operations on heaps.
   A. Neither insertion nor removal in a heap is better than O(n),.
   B. Insertion is better than O(n), but removal is not.
   C. Removal is better than O(n), but insertion is not.
   D. Both insertion and removal are better than O(n),.

7. Suppose that we have implemented a priority queue by storing the items in a max heap (using an array for the heap items). We are now executing a reheapification upward (after insertion) and the out-of-place node is at data[i] with priority given by data[i].priority. Which of the following boolean expressions is TRUE to indicate that the reheapification IS NOT YET DONE.
   A. (i > 0)
   B. (data[(i-1)/2].priority < data[i].priority)
   C. (i > 0) && (data[(i-1)/2].priority < data[i].priority)
   D. (i > 0) || (data[i/2].priority < data[i].priority)

8. Write the C++ code to insert an item into a d-heap of size 4. For efficiency, use shifting instead of multiplication or division.

9. Consider the binomial queue below. Show the heap after delete min.
10. Show the results of adding the following numbers to an initially empty binomial queue. Show each step of your work:

24 1 54 6 14 6 22

11. Show the results of merging the queue from problem 10 with the queue below.

12. Show the results of merging the following two leftist heaps:

13. Show the results of deleteMin on the following leftist heap: