Answer the following questions in a clear and precise manner:

1. **For an array of size n, n ≥ 8, how many times is the 8-Min box used in method min8Min? State and explain an exact bound.**
   n^2
2. **For an array of size n, n ≥ 8, how many times is the 8-Min box used in method isSorted8min? How many times is the 8-Sort box used in method isSorted8Sort. State and explain exact bounds.**
   for sizes n = 8, 8-Min box is used once.
   for sizes n > 8, 8-Min box is used n times.
3. **For an array of size n, n ≥ 8, state and explain the worst case asymptotic performance of the algorithms sortSelect8Sort and sortSelect8Min**
   for n>=8, the asymptotic performance of my sortSelect8Sort is n(n - 4)/32. O(n^2)

   for n>=8, the asymptotic performance of my sortSelect8Min is O(n^2), exactly as much as that of a standard selection sort.

4. **For an array of size n, n ≥ 8, state and explain the worst case asymptotic performance of the algorithm sortMerge8Sort. How does it compare to the standard Merge Sort with respect to the number of "comparisons"?**

   the sortMerge8Sort has an asymptotic performance of O(nlogn). it generally takes 27% of

5. **Assume the 8-Sort box is a stable sorting box. Under this assumption, is your algorithm sortMerge8Sort stable? What about sortSelect8Sort? Explain your answers.**
   For my algorithm, the sortMerge8Sort is not stable and the sortSelect8Sort is stable. This is because the merge sort behaves a lot like a merge sort, and merge sort is usually unstable. The selection sort, however always keeps the track of minimums.

6 Instead of an 8-Sort box, the ruler of Sortakastan grants you access to a more powerful Magic box. For an array of size n, you have access to a √n-Sort box. Describe how you would use such a box (you do not need to implement the resulting algorithm). How many times is the Magic boxed used when sorting an array of size n (express in terms of n and in the asymptotic sense)?

for Selection Sort, the number of comparisons are (n^2)/16 which could be dumbed down as (n^2)/(2*Size of the box)
if the size of the box was sqrt(n), the final complexity would be (n^2)/(2*sqrt(n))
or (1/2)(n^1.5)
Which can be written as O(n*sqrt(n))

To use such a box, we would instead of padding the array to the size of 8, we would pad it to a size of sqrt(n) or a multiple of sqrt(n).
For selection sort, instead of skipping 4 elements at a time, we'd skip (1/2)sqrt(n) elements at a time and continue accordingly.
For merge sort, instead of using 4 elements from each array at a time, we'd use (1/2)sqrt(n) elements at a time and then sort them.

Data:

Sort	500	1000	5000	10000	50000	100000
Selection	125249	500499	12502499	50004999	1250024999	705082703
Selection 8 sort	7750	31125	780625	3123750	78118750	312487500
Selection 8 min	125249	500499	12502499	50004999	1250024999	705082703
Merge	3863	8711	55224	120472	718120	1536050
Merge magic	655	1508	15306	33112	178392	381784

Time (Milliseconds)

Sort	500	1000	5000	10000	50000	100000
Selection	10	17	86	159	3594	14451
Selection 8 sort	31	130	311	945	16072	62371
Selection 8 min	24	74	313	1217	30422	93230
Merge	6	10	8	11	31	30
Merge magic	14	36	105	140	1185	3097

Speedups

Sort	500	1000	5000	10000	50000	100000
Selection 8 sort	0.06187674 1530871	0.06218793 6439433	0.06243751 7491503	0.06246875 4373938	0.06249375 0174992	0.4431926 90262322
Selection 8 min	1	1	1	1	1	1
Merge magic	0.16955733 89	0.17311445 2990472	0.27716210 343329	0.27485224 7825221	0.24841530 6633989	0.2485492 00872367

The speed up for the Selection8Sort is directly proportional to the size of the input. This is because it uses the traditional method of selection sorting, but instead of using individual elements as the smallest unit of an array, it uses the 8sort box as the smallest unit.

The speed up for Selection8min is constant because it makes comparisons based on the smallest element from a padded 8min box. The number of comparisons that Selection8min makes is the same as that of a standard selection sort.

The speed up for Merge Magic depends loosely on the size of the input and and largely on how much of the array is sorted

(2) Consider the performance of all three Magic Box algorithms. Is one of the Magic Box sorts consistently faster for all sizes of n? Explain the behavior you observe.

For my code, MagicMerge sort is consistently faster than the other algorithms for all sizes of n.
Magic merge has a consistent speed up over the others. Selection8Sort and Selection 8min are both O(n^2).
Merge sort is always faster than selection sort with a complexity of O(nlogn) which makes magic merge significantly faster than the other sorts.