1. Carrano, Chapter 9, Exercise 11: Apply the selection sort, bubble sort, and insertion sort to

(a) An inverted array: 8 6 4 2.

Solution: (Selection sort)

Original array: 8 6 4 2
1st pass: 2 6 4 8
2nd pass: 2 4 6 8
3rd pass: 2 4 6 8

Note the 3rd swap is still necessary, since the computer does not know the first two are sorted yet.

Solution: (Bubble sort)

Original array: 8 6 4 2
1st pass: 6 8 4 2
   6 4 8 2
   6 4 2 8
2nd pass: 4 6 2 8
   4 2 6 8
3rd pass: 2 4 6 8

Solution: (Insertion sort)

Original array: 8 6 4 2
1st pass: 6 8 4 2
2nd pass: 4 6 8 2
3rd pass: 2 4 6 8

(b) An ordered array: 2 4 6 8.

Solution: (Selection sort)

Original array: 2 4 6 8
1st pass: 2 4 6 8
2nd pass: 2 4 6 8
3rd pass: 2 4 6 8

Note selection sort has no way of terminating early.

Solution: (Bubble sort)

Original array: 2 4 6 8
1st pass: 2 4 6 8

Bubble sort terminates early, since if no adjacent swaps have occurred in a given pass, the array is already sorted.

Solution: (Insertion sort)
Original array: 2 4 6 8
1st pass: 2 4 6 8
2nd pass: 2 4 6 8
3rd pass: 2 4 6 8

Insertion sort has no way of terminating early.

2. Carrano, Chapter 9, Exercise 12: How many comparisons would be needed to sort an array containing 25 elements using the bubble sort in

(a) the worst case?

   Solution: In the worst case, bubble sort does not terminate early. Recall the loop structure:
   
   ```
   for (int pass = 1; (pass < n) && !sorted; ++pass)
   {
       sorted = true;
       for (int index = 0; index < n-pass; ++index)
       {
           /* body of loop; if any swap occurs, sorted becomes false */
       }
   }
   ```
   
   So all together, the body of the loop executes
   
   \[(n - 1) + (n - 2) + \cdots + 2 + 1 = \frac{n(n-1)}{2}\]
   
   times. To calculate this, notice the sum is equal to
   
   \[1 + 2 + \cdots + (n-1) + n = \frac{n(n+1)}{2} - n = \frac{n(n-1)}{2}.\]
   
   So the body of the loop (and thus the main comparison) executes
   
   \[\frac{25 \cdot 24}{2} = 300\]
   
   times for \(n = 25\).

(b) the best case?

   In the best case, the array is already sorted, and the bubble sort terminates after the first pass of \(n - 1\) comparisons. So if \(n = 25\), there are \(24\) comparisons in the best case.

   Why? (Don’t worry about comparisons used to traverse the loop, but only the comparing the actual array elements.)

3. Carrano, Chapter 9, Exercise 16: Trace the mergesort algorithm as it sorts the following array into ascending order. List all the calls to `mergesort` and to `merge` in the order in which they occur

   20 80 40 25 60 30
Solution:

Function calls

mergesort(theArray, 0, 5) 20 80 40 25 60 30
mid = 2
  mergesort(theArray, 0, 2)
    mid = 1
      mergesort(theArray, 0, 1)
        mid = 0
          mergesort(theArray, 0, 0)
            base case: return
          mergesort(theArray, 1, 1)
            base case: return
          merge(theArray, 0, 0, 1) 20 80 40 25 60 30
        base case: return
      merge(theArray, 0, 1, 2) 20 40 80 25 60 30
    base case: return
  merge(theArray, 1, 1, 2) 20 40 80 25 60 30
mergesort(theArray, 2, 2)
  base case: return
mergesort(theArray, 3, 5)
  mid = 4
    mergesort(theArray, 3, 4)
      mid = 3
        mergesort(theArray, 3, 3)
          base case: return
        mergesort(theArray, 4, 4)
          base case: return
        merge(theArray, 3, 3, 4) 20 40 80 25 60 30
      base case: return
    merge(theArray, 3, 4, 5) 20 40 80 25 30 60
    base case: return
  merge(theArray, 3, 5, 5) 20 25 30 40 60 80
  base case: return
  merge(theArray, 2, 2, 5) 20 25 30 40 60 80

4. Carrano, Chapter 9, Exercise 18: Trace the quicksort algorithm as it sorts the following array into ascending order. List the calls to quicksort and to partition in the order in which they occur.

20 80 40 25 60 10 15

Solution:

Function calls

quicksort(theArray, 0, 6) 20 80 40 25 60 10 15
partition(theArray, 0, 6) 20 80 40 25 60 10 15
  pivot = 20, lastS1 = 0
  for (lastUnknown = 1 to 6)
lastUnknown = 1
(80 < 20) false: no change
lastUnknown = 2
(40 < 20) false: no change
lastUnknown = 3
(25 < 20) false: no change
lastUnknown = 4
(60 < 20) false: no change
lastUnknown = 5
(10 < 20) true:
  lastS1 = 1, swap
lastUnknown = 6
(15 < 20) true:
  lastS1 = 2, swap

swap pivot
pivotIndex = lastS1 = 2
quicksort(theArray, 0, 1)
partition(theArray, 0, 1)
  pivot = 15, lastS1 = 0
  for (lastUnknown = 1 to 1)
    lastUnknown = 1
    (10 < 15) true:
      lastS1 = 1, swap
  swap pivot
  pivotIndex = lastS1 = 1
quicksort(theArray, 0, 0)
  base case: return
quicksort(theArray, 2, 1)
  base case: return
quicksort(theArray, 3, 6)
partition(theArray, 3, 6)
  pivot = 25, lastS1 = 3
  for (lastUnknown = 4 to 6)
    lastUnknown = 4
    (60 < 25) false: no change
    lastUnknown = 5
    (80 < 25) false: no change
    lastUnknown = 6
    (40 < 25) false: no change
  swap pivot
  pivotIndex = lastS1 = 3
quicksort(theArray, 3, 2)
  base case: return
quicksort(theArray, 4, 6)
partition(theArray, 4, 6)
  pivot = 60, lastS1 = 4
for (lastUnknown = 5 to 6)
    lastUnknown = 5
    (80 < 60) false: no change 10 15 20 25 60 80 40
    lastUnknown = 6
    (40 < 60) true:
        lastS1 = 5, swap 10 15 20 25 40 60 80
        swap pivot
        pivotIndex = lastS1 = 5
    quicksort(theArray, 4, 4)
    base case: return
    quicksort(theArray, 6, 6)
    base case: return

5. Implement a radix sort on the following data, which represent an ordinary deck of cards. Each card has a suit, chosen from C, D, H, S (clubs, diamonds, hearts, spades). The suits are to be ordered in this fashion: C < D < H < S. In addition each card has one of thirteen values, represented by

2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < T < J < Q < K < A

(T is ten, J is jack, Q is queen, K is king, and A is ace). Use a radix sort to sort the following hand of cards into suits, with the cards in each suit sorted as well.

S8, CT, D4, SA, C4, D3, DJ, H8, S9, H4

Show your work. Your final answer should be

C4, CT, D3, D4, DJ, H4, H8, S8, S9, SA.

Solution:

Original list: S8 CT D4 SA C4 D3 DJ H8 S9 H4
Group last digit: 3 group: D3; 4 group: D4, C4, H4;
                8 group: S8, H8; 9 group: S9;
                T group: CT; J group: DJ; A group: SA
Recombine list: D3 D4 C4 H4 S8 H8 S9 CT DJ SA
Group first digit: C group: C4, CT; D group: D3, D4, DJ;
                  H group: H4, H8; S group: S8, S9, SA
Recombine list: C4 CT D3 D4 DJ H4 H8 S8 S9 SA