SEARCH AND SORTING ALGORITHMS

Short Answer

1. * Discuss the efficiency and inefficiency of *linear search*. Repeat the question for *binary search*.
2. * What are the best case and worst case of *linear search.* Repeat the question for *binary search*.
3. * Identify the key difference(s) between *linear search* and *binary search.*
4. * Identify the best case and worst case of *bubble sort*. Repeat the question for *selection sort, insertion sort*, and *interchange sort.*

5. ** What is the major difference between *insertion sort* and the other sorting algorithms?
6. ** Identify the key difference(s) between *bubble sort* and *interchange sort*.
7. ** Describe the idea shared by both *selection sort* and *interchange sort*.
8. ** *Linear search* must examine, on average, half of the N elements of an array while *binary search* needs to only check $\log_2 N$. Thus, *binary search* is a great deal more efficient than *linear search*. Why, then, should we still study and use *linear search*?
9. ** The efficiency of *linear search* could be improved by adding a sentinel. Explain why.
10. ** Why is *selection sort* more efficient than *bubble sort* on large arrays?
11. ** If you want to sort an array in descending order, which part of the *bubble sort* algorithm should be modified? Repeat the question for *selection sort, insertion sort*, and *interchange sort.*
12. ** Consider the following array of integers, `{1, 4, 6, 7, 9, 10, 14}`. For *linear search*, which elements will be examined to determine that 9 is in the array, and which elements will be examined to determine that 11 is not in the array? Repeat the question for *binary search.*
13. ** Consider the following array of integers, `{26, 48, 12, 92, 28, 6, 33}`. For *bubble sort*, show the resulting array for each of the first three iterations of the outer loop. Repeat the question for *selection sort, insertion sort*, and *interchange sort.*
14. ** Consider an array of 5 integers. What is the maximum number of data moves if the array is sorted by *bubble sort*? Justify your answer. Repeat the question for *selection sort, insertion sort*, and *interchange sort.*
15. ** Consider an array of 5 integers. What is the maximum number of comparisons if the array is sorted by *bubble sort*? Justify your answer. Repeat the question for *selection sort, insertion sort*, and *interchange sort.*

16.** This C++ fragment implements *bubble sort* for ascending order. Is the code valid? If no, suggest how to fix the errors.

```
for (i = 1; i < n; i++)
   for (j = n - 1; j <= i; j--)
       if (a[j] > a[j - 1])
           a[j] = a[j - 1]);
           a[j-1] = a[j];
```

17.** Which sorting algorithm is demonstrated by this C++ code segment? Explain.

```
int s, t, numItems = 10;
for (s = (numItems - 1); s >= 0; s--){
   for (t = 1; t <= s; t++){
       if (arr[t - 1] > arr[t]) {
           int temp = arr[t - 1];
           arr[t - 1] = arr[t];
           arr[t] = temp;
       }
   }
}
```

18.** Consider the following array of strings, [“Imogen”, “Fletcher”, “Kirstie”, “Zoe”, “Gavin”]. Is the given array could be searched using *binary search*? Justify your answer.

19.** *Selection sort compares each pair of keys at most once*. Is the given statement TRUE or FALSE? Justify your answer.

20.** Given the following array, {1, 2, 3, 6, 5, 9}. Which sorting algorithm(s) might you choose for the given array? Justify your answer.

21.** Consider a situation where swap operation is very costly. Which sorting algorithm(s) should be preferred so that the number of swaps are minimized in general? Justify your answer.

22.*** Assume that the following timings were taken of three algorithms as they processed lists of integers. Which of these algorithms is most likely to be *selection sort*? What characteristics of the sort (and its performance) led you to your choice?

	time to process 2000 integers	time to process 4000 integers	time to process 8000 integers
Algorithm 1	0.1431 sec	0.5722 sec	2.2989 sec
Algorithm 2	0.8011 sec	1.4300 sec	2.4512 sec
Algorithm 3	0.0132 sec	0.0304 sec	0.0634 sec

23.*** We could improve the efficiency of *bubble sort* by early termination using only an additional Boolean variable. Describe how to do that with your pseudo-code.

24.*** Consider a situation when all elements of the input array are identical. Which sorting algorithm will take least time?

25.*** If a *linear search* is performed on an array, and it is known that some items are searched for more frequently than others, how can the contents of the array be reordered to improve the average performance of the search?

Fill-in-the-Blank

1. * Fill in each of the following blanks with an appropriate word.
   - *Linear search starts with the _____ element while binary search starts with the _____ element.*
   - *The processing of reversing the positions of two variables is called _____.*
   - *A _____ algorithm is a method of locating a specific item of information in a larger collection of data.*
2. * Fill in each of the following blanks with an appropriate algorithm name.
   - _____ *will not work properly unless the values in the array are sorted.*
   - _____ *repeatedly compares adjacent pairs and swaps them if they are in the wrong order.*
   - _____ *iteratively chooses the smallest element of the unsorted array and swaps it with the left-most element of the unsorted array.*
   - _____ *iteratively picks up the left-most element of the unsorted array and finds an appropriate position in the sorted array for this element.*
   - _____ *repeatedly compares an element with every other elements of the array and swaps them if they are in the wrong order.*

3. ** Consider the following array of integers, `{15,20,10,18}`, and a sequence showing the array after each elements swap operation.
   - *If the sequence is 15,10,20,18 -- 15,10,18,20 -- 10,15,18,20, then the array is being sorted by _____.*
   - *If the sequence is 10,20,15,18 -- 10,15,20,18 -- 10,15,18,20, then the array is being sorted by _____.*
   - *If the sequence is 15,20,10,18 -- 10,15,20,18 -- 10,15,18,20 -- 10,15,18,20, then the array is being sorted by _____.*
4. ** Fill in each of the following blanks with appropriate indexical values.
   - *You are using binary search to look for the number 28 in the array* {5, 11, 25, 28, 45, 78, 100, 120, 125}*. When the algorithm stops, the value of the variable first (or left) is _____ and the value of the variable last (or right) is _____.*
   - *You are using binary search to look for the number 50 in the array {*11, 12, 16, 18, 23, 29, 33, 48, 54, 57, 68, 77, 84, 98*}. When the algorithm stops, the value of the variable first (or left) is _____ and the value of the variable last (or right) is _____.*

5. ** Fill in each of the following blanks with an appropriate number.
   - *32 teams qualified for the 2018 World Cup. If the names of the teams were arranged in sorted order (an array), the binary search would have to examine at most _____ items to find the location of a particular team in the array.*
   - *In 2013, there were 193 member states in the United Nations. If the names of these states were sorted alphabetically in an array, the binary search would have to examine at most _____ names to locate a particular name in the array.*
   - *Given an array of prime numbers that are from 2 to 311 in sorted order: {2, 3, 5, 7, 11, 13, ..., 307, 311}. There are 64 items in the array. The binary search would have to examine about _____ items before concluding that 52 is not in the array, and therefore not prime.*
   - *The 2014 "Catalogue of Life" contains about 1,580,000 names of species. If these names were sorted in an array, in the worst case, how long would it take to use binary search to find the name of a particular species in the array? _____.*
6. ** For each of the following situations, choose the sorting algorithm that will perform the best. (Some questions might have more than one answer, but you only need to list one.)
   - *I am sorting data that is stored over a network connection. Based on the properties of that connection, it is extremely expensive to "swap" two elements. But looping over the elements and looking at their values is very inexpensive. I want to minimize swaps above all other factors. A good choice would be: _____.*
   - *I have an array that is already sorted. Periodically, some new data comes in and is added to the array at random indexes, messing up the ordering. I need to re-sort the array to get it back to being fully ordered. I do not want to use very much additional memory during the sort. A good choice would be: _____.*
   - *Instead of sorting the entire data set, you only need the k smallest elements where k is an input to the algorithm but is likely to be much smaller than the size of the entire data set. A good choice would be: _____.*
   - *You are the county clerk. You receive birth records from the towns in the county in batches throughout the year. You need to sort these records by date and up to now you are having 1,000,000 records. A good choice of sorting algorithm would be: _____.*
7. ** Fill in the blanks with appropriate numbers. *Consider the following lists of partially sorted numbers,* [1 3 4 8 9 || 5 2]*. The double bars represent the sort marker. The insertion sort needs _____ comparisons and _____ elements needed to be moved to sort the next number.*
8. ** Suppose that you have an array of length 2N consisting of N B's followed by N A's.
   Here is an example array when N = 10: B B B B B B B B B B A A A A A A A A A A

How many comparisons does it take to *insertion sort* the array as a function of N? Use tilde notation to simplify your answer. _____

9. ** Choose the best answer for each of the following single-choice questions about *selection sort*.

- *Through experiment, you determine that selection sort performs 5000 comparisons when sorting an array of some size k. If you doubled the size of the array to 2k, approximately how many comparisons would you expect it to perform?*

  a) 5000 b) 10000 c) 20000
  d) 40000 e) The value would depend on the contents of the array

- *Through experiment, you determine that selection sort performs 5000 data moves when sorting an array of some size k. If you doubled the size of the array to 2k, approximately how many moves would you expect it to perform?*

  a) 5000 b) 10000 c) 20000
  d) 40000 e) The value would depend on the contents of the array

- *A selection sort of 100 items has completed 42 iterations of the main loop. How many items are now guaranteed to be in their final spots (never to be moved again)?*

  a) 21 b) 41 c) 42 d) 43

10.** Consider the following array, `{32, 33, 5, 2, 14, -4, 22, 39, 34, -9}`.

Each of the following is a view of a sort in progress of the given array, i.e., the array is shown after a few of passes of the outermost loop has completed.

- `{2, 5, 32, 33, 14, -4, 22, 39, 34, -9}`
  Sorting algorithm: _____
- `{2, 5, -4, 14, 22, 32, -9, 33, 34, 39}`
  Sorting Algorithm: _____
- `{-9, -4, 2, 5, 14, 33, 22, 39, 34, 32}`
  Sorting algorithm: _____

Which sort is which? Choose between *bubble sort, selection sort, and insertion sort*. Each sort is used exactly once.

11.** Fill in the blanks in the following code segment to implement *sequential search* on the given array. If the number is found, the function should return its array subscript. Otherwise, –1 is returned indicating the value did not appear in the array.

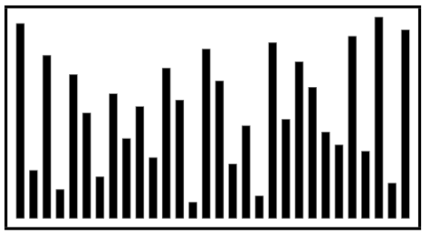
```
int searchList(const int list[], int numElems, int value) {
    int index = ________;     // Used as a subscript to search array
    int position = -1;        // To record position of search value
    bool found = false;  // Flag to indicate if the value was found
    while ( _____________________ && _____________________) {
```

```
            if (____________________) {   // If the value is found
                found = true;    // Set the flag
                position = index;     // Record the value's subscript
            }
            index++;         // Go to the next element
        }
        return ____________________;
    }
```

12. ** Fill in the blanks in the following pseudo-code segment to implement a search algorithm of the given array. Which search algorithm is it?

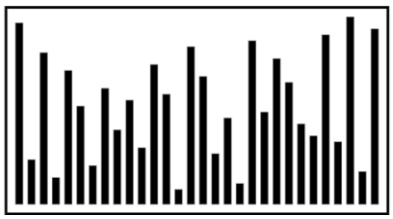
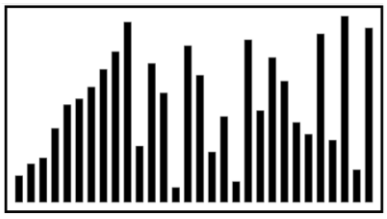
```
array people[5]
people = [“Imogen”, “Fletcher”, “Kirstie”, “Zoe”, “Gavin”]
found = False
x = 0
searchfor = inputName()
while found = False …… x < 5
    if people[x] = searchfor then
        found = ……
        print “found at position” + ……
    else
        x = x + 1
    endif
endwhile
```

13. *** Suppose we want to sort a collection of sticks according to heights.

From: 

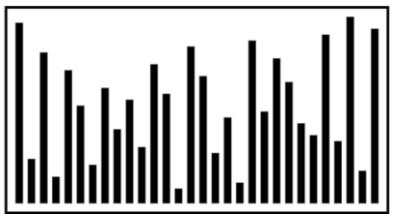
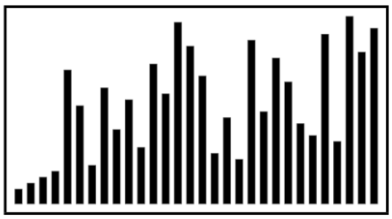
To: 

Each of the following diagrams show the state of the sticks part way through sorting using a different sorting algorithm. For each diagram, say which algorithm is being used, and give a brief explanation for your choice.

From:  Part way: 

The algorithm above is _____ because _____

_____

From:  Part way: 

The algorithm above is _____ because _____

_____

14. *** The column on the left is the original input of strings to be sorted; the column on the right are the strings in sorted order; the other columns are the contents at some intermediate step during one of the five sorting algorithms.

Which columns demonstrate the *selection sort* and *insertion sort*?

0	prim	heap	flip	edge	flip	miss	edge
1	left	left	heap	find	left	load	find
2	load	load	java	flip	load	loop	flip
3	push	lazy	left	hash	loop	list	hash
4	sink	node	load	heap	miss	left	heap
5	time	hash	loop	java	null	lazy	java
6	loop	loop	miss	lazy	prim	heap	lazy
7	flip	flip	null	left	push	java	left
8	null	null	prim	list	sink	flip	list
9	miss	miss	push	load	swim	hash	load
10	trie	edge	rank	loop	time	edge	loop
11	swim	find	sink	miss	trie	find	miss
12	java	java	sort	time	find	node	node
13	sort	list	swim	sort	heap	null	null
14	rank	prim	time	rank	java	prim	prim
15	heap	rank	trie	sink	list	push	push
16	list	sort	list	null	rank	rank	rank
17	find	swim	find	swim	sort	sink	sink
18	tree	tree	tree	tree	tree	sort	sort
19	edge	trie	edge	prim	edge	swim	swim
20	hash	time	hash	push	hash	time	time
21	node	sink	node	node	node	tree	tree
22	lazy	push	lazy	trie	lazy	trie	trie
23	type	type	type	type	type	type	type

15. *** The column on the left is the original array of 24 integers to be sorted; the rightmost column contains the integers in sorted order; the other columns are the contents of the array at some intermediate step during one of the five sorting algorithms listed below. Match each algorithm by writing its number in the box under the corresponding column. Use each number once.

(0) Original array (2) Selection sort (4) Insertion sort

(1) Sorted array (3) Bubble sort (5) Exchange sort

0	63	11	11	11	11	11
1	21	19	19	19	19	19
2	19	21	21	21	21	21
3	32	29	25	25	25	25
4	45	31	29	29	29	29
5	60	32	31	31	31	31
6	31	44	32	32	32	32
7	79	45	44	44	44	44
8	48	48	45	45	45	45
9	11	60	48	48	48	48
10	71	63	63	50	50	50
11	88	71	50	52	52	52
12	29	79	52	63	60	60
13	99	88	60	99	63	63
14	89	89	67	89	99	67
15	44	99	71	79	89	71
16	86	86	79	86	88	79
17	52	52	81	88	86	81
18	92	92	86	92	92	86
19	50	50	88	71	79	88
20	25	25	89	60	71	89
21	67	67	92	67	67	92
22	93	93	93	93	93	93
23	81	81	99	81	81	99
	0					**1**

True or False

Choose T (True) or F (False) for each of the following statements and then briefly explain in one or two sentences.

1. T F The maximum number of comparisons that *linear search* performs in searching a value in an array of size N is always N.
2. T F *Binary search* does not work unless the array is sorted.
3. T F For *binary search*, the successful search requires only $\log_2 N$ comparisons, but the unsuccessful search requires N comparisons.
4. T F *Binary search* terminates when the variables first becomes equal to last.
5. T F *Linear search* will be more efficient if the array is sorted.
6. T F *Insertion sort* works best to sort data as it arrives, one piece at a time, perhaps from a network.
7. T F *Selection sort* makes the same number of comparisons regardless of the input array.
8. T F *Bubble sort* is an easy way to arrange data into ascending order, but it cannot arrange data into descending order.
9. T F *Interchange sort* compares adjacent elements and swaps them if they are in wrong order.
10. T F There is no difference between the best case and worst case of *selection sort.*
11. T F All sorting algorithms (i.e., bubble sort, selection sort, insertion sort and exchange sort) requires two nested loops in their implementation.
12. T F All sorting algorithms (i.e., bubble sort, selection sort, insertion sort and exchange sort) requires the same number of comparisons when considering a given array of N integers.
13. T F *Selection sort* makes one data exchange at the end of every pass.
14. T F The first swap that *insertion sort* would make on the list, (5, 3, 4, 9, 1), is 5 and 3.
15. T F *Bubble sort* gains efficiency by splitting the data in half.

Algorithm Workbench

1. * Provide appropriate modifications to *selection sort* so that it can be used for descending order. Repeat the question for *selection sort, insertion sort*, and *interchange sort.*

For each of the following requirements, consider both cases: the array is unsorted, and the array is already sorted.

2. ** Write the function, `remove`, to take three parameters: an array of integers, the number of elements in the array, and an integer (say, `removeItem`). The function should find and delete the first occurrence of `removeItem` in the array. If the value does not exist or the array is empty, output an appropriate message. (Note that after deleting the element, the number of elements in the array is reduced by 1.)
3. ** Write the function, `removeAt`, to take three parameters: an array of integers, the number of elements in the array, and an integer (say, `index`). The function should delete the array element indicated by `index`. If `index` is out of range or the array is empty, output an appropriate message. (Note that after deleting the element the number of elements in the array is reduced by 1.).
4. ** Write the function, `removeAll`, to take three parameters: an array of integers, the number of elements in the array, and an integer (say, `removeItem`). The function should find and delete all the occurrences of `removeItem` in the array. If the value does not exist or the array is empty, output an appropriate message. (Note that after deleting the element, the number of elements in the array is reduced.)
5. ** Print the elements of an array in the decreasing frequency. If two numbers have same frequency, print the one which came first. For example,

Input	Output
{2, 5, 2, 8, 5, 6, 8, 8}	{8, 8, 8, 2, 2, 5, 5, 6}
{2, 5, 2, 6, -1, 9999999, 5, 8, 8, 8}	{8, 8, 8, 2, 2, 5, 5, 6, -1, 9999999}

6. ** Count the number of inversions in an array of integers, indicating how far the array is from being sorted. If array is already sorted, the inversion count is 0. If array is sorted in reverse order, the inversion count is the maximum. Formally speaking, two elements `a[i]` and `a[j]` form an inversion if `a[i] > a[j] and i < j`. For example,

Input	Output	Explanation
{8, 4, 2, 1}	6	(8,4), (4,2), (8,2), (8,1), (4,1), (2,1)
{3, 1, 2}	2	(3,1), (3,2)

7. ** Write a function to receive an array of strings and arrange the strings in ascending order.

For example,

Input	[Farm, Zoo, Car, Apple, Bee, Golf, Bee, Dog, Golf, Zoo, Zoo, Bee, Bee, Apple]
Output	[Apple, Apple, Bee, Bee, Bee, Bee, Car, Dog, Farm, Golf, Golf, Zoo, Zoo, Zoo]

Then write another function that accepts the newly sorted array and outputs the set of distinct strings, each of which associates with its number of occurrences. For example,

Input	[Apple, Apple, Bee, Bee, Bee, Bee, Car, Dog, Farm, Golf, Golf, Zoo, Zoo, Zoo]
Output	Apple: 2, Bee:4, Car:1, Dog:1, Farm:1, Golf:2, Zoo:3

8. ** Write code to receive a set of positive integers, representing test scores for a class, and output how many times a particular number appears in the list. You may assume that the data set has at most 100 numbers and -999 marks the end of the input data. The numbers must be output in increasing order.

For example, for the following data

55 80 78 92 95 55 78 53 92 65 78 95 85 92 85 95 95

the output is

Test Score	53	55	65	78	80	85	92	95
Count	1	2	1	3	1	2	3	4

9. ** Your state is in a process of creating a weekly lottery. Once a week, five distinct random integers between 1 to 40 (inclusive) are drawn. If a player guesses all the numbers correctly, the player wins a certain amount. Write a program that does the following:

a) Generate five distinct random lottery numbers between 1 to 40 (inclusive) and stores them in an array.

b) Sort the array containing the lottery numbers.

c) Prompt the player to select five distinct integers between 1 and 40 (inclusive) and stores the number in an array. The player can select the numbers in any order and the array containing the numbers need not to be sorted.

d) Determine whether the player guessed the lottery numbers correctly. If the player guessed the lottery number correctly, output the message "You win!"; otherwise, output the message "You lose!", as well as the lottery numbers.

Your program should allow a player to play the game as many times as he wants to play. Before each play, generate a new set of lottery numbers.

10.* Given an array A containing 0s, 1s and 2s only. Write code to sort the given array such that all 0s should come first, then all 1s and all 2s in last. This is a variation of famous Dutch national flag problem or three-ways partitioning. Solve the task by implementing the following given algorithm.

The problem was posed with three colours, 0, 1, and 2. The array is divided into four sections:

1. a[1..Lo-1] zeroes (red)
2. a[Lo..Mid-1] ones (white)
3. a[Mid..Hi] unknown
4. a[Hi+1..N] twos (blue)

The unknown region is shrunk while maintaining these conditions

1. Lo := 1; Mid := 1; Hi := N;
2. while Mid <= Hi do
   i. Invariant: a[1..Lo-1]=0 and a[Lo..Mid-1]=1 and a[Hi+1..N]=2; a[Mid..Hi] are unknown.
   ii. case a[Mid] in
      - 0: swap a[Lo] and a[Mid]; Lo++; Mid++
      - 1: Mid++
      - 2: swap a[Mid] and a[Hi]; Hi--