

1. For the array below, show how the array elements are rearranged in the first pass through the array (at the end of the first time through the outer loop) for each sort.

**Selection** sort:

**Bubble** sort:

Original arrays:

| 16 | 21 | 14 | 17 | 15 |

| 16 | 21 | 14 | 17 | 15 |

After one pass through the arrays:

| 14 | 21 | 16 | 17 | 15 |

| 16 | 14 | 17 | 15 | 21 |

2. The sorts we studied place the array elements into ascending order (smallest to largest value). For both the selection and the bubble sort, show how the code would be changed to be able to sort an array into descending order (largest to smallest value), rewriting below only the specific statement(s) that change. Be sure to include the number for each statement that you rewrite.

Selection Sort

Bubble Sort

9 if (list[minIndex] > list[smallestIndex])

10 if (a[j] < a[j+1])

3. Given this array to be searched (listLength is 8):

| 10 | 20 | 35 | 47 | 55 | 61 | 74 | 78 | | | | | |  
0 1 2 3 4 5 6 7

- a) If the searchItem is 47, what value is returned by:

SeqSearch 3

- b) If the search key is 30, what value is returned by:

SeqSearch -1

- c) If the search key is 80, what value is returned by:

SeqSearch -1

- d) Would it be possible to search this array using a binary search? **yes** or **no**

4. Given this sorted array to be searched (listLength is 14):

10	15	20	25	30	35	40	45	50	55	60	65	70	75
0	1	2	3	4	5	6	7	8	9	10	11	12	13

a) Trace the operation of the binary search for searchItem = 35:

first = 0 last = 13 mid = 6 searchItem is compared with 40

then first = 0 last = 5 mid = 2 searchItem is compared with 20

then first = 3 last = 5 mid = 4 searchItem is compared with 30

then first = 5 last = 5 mid = 5 searchItem is compared with 35

How many key comparisons were made in the binary search? 4

How many key comparisons would have been required using the book's sequential search? 6

b) Trace the operation of the binary search for searchItem = 62:

first = 0 last = 13 mid = 6 searchItem is compared with 40

then first = 7 last = 13 mid = 10 searchItem is compared with 60

then first = 11 last = 13 mid = 12 searchItem is compared with 70

then first = 11 last = 11 mid = 11 searchItem is compared with 65

then first = 11 last = 10

How many key comparisons were made in the binary search? 5

How many key comparisons would have been required using the book's sequential search? 14

## Sorts

```
1 void SelectionSort (int list[], int length)
  {
2   int index;
3   int smallestIndex;
4   int minIndex;
5   int temp;

6   for (index = 0; index < length - 1; index++)
    {
7     smallestIndex = index;

8     for (minIndex = index + 1; minIndex < length; minIndex++)
9       if (list[minIndex] < list[smallestIndex])
10        smallestIndex = minIndex;

11    temp = list[smallestIndex];
12    list[smallestIndex] = list[index];
13    list[index] = temp;
    } // end of one pass through the array loop
  }
```

```
1 void BubbleSort (int a[], int length)
  {
2   int bottom;
3   int j;
4   int lastSwapNdx;
5   int hold;

6   bottom = length - 1;
7   while (bottom > 0)
    {
8     lastSwapNdx = 0;

9     for (j = 0; j < bottom; j++)
10      if (a[j] > a[j+1])
11       {
12         hold = a[j];
13         a[j] = a[j+1];
14         a[j+1] = hold;
15         lastSwapNdx = j;
16       }

17    bottom = lastSwapNdx;
18  } // end of outer loop to make one pass through the array
  }
```

## Searches

```
int SeqSearch (const int list[], int listLength, int searchItem)
{
    int loc;
    bool found = false;

    for (loc = 0; loc < listLength; loc++)
        if (list[loc] == searchItem)
            {
                found = true;
                break;
            }
    if (found)
        return loc;
    else return -1;
}
```

```
int BinarySearch (const int list[], int listLength, int searchItem)
{
    int first = 0;
    int last = listLength - 1;
    int mid;
    bool found = false;

    while (first <= last && ! found)
        {
            mid = (first + last) / 2;
            if (list[mid] == searchItem)
                found = true;
            else if (list[mid] > searchItem)
                last = mid - 1;
            else first = mid + 1;
        }
    if (found)
        return mid;
    else return -1;
}
```