1D Arrays - Review

Exercises:

1. Binary Search
2. Binary Search
3. Sort
4. Sort
5. What is the output

1.

A. Because of its efficiency, binary search is the best search for any array, regardless of its size and order.

True / False

1.

A. Because of its efficiency, binary search is the best search for any array, regardless of its size and order.

True / **False**

1.

B. Under what circumstances should we use binary search?

1.

B. Under what circumstances should we use binary search?

1. The array must be sorted.
2. It is recommended to be used on large arrays, but it could be used on any sorted array.

2. Using Binary Search, which elements in the array below are compared to the target?

A. target = 40

10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**95**	**100**
0	1	2	3	4	5	6	7	8	9	10

A. target = 40

2. Using Binary Search, which elements in the array below are compared to the target?

A. target = 40

10	20	30	40	50	60	70	80	90	95	100
0	1	2	3	4	**5**	6	7	8	9	10

```
first = 0
last  = 10
mid = (first + last)/2 = 5
40 < list[5] => search the lower half:
       last = mid - 1 = 4
```

first	last	first <= last	mid	target ? list[mid]
0	10	True	5	40 < 60

2. Using Binary Search, which elements in the array below are compared to the target?

A. target = **40**

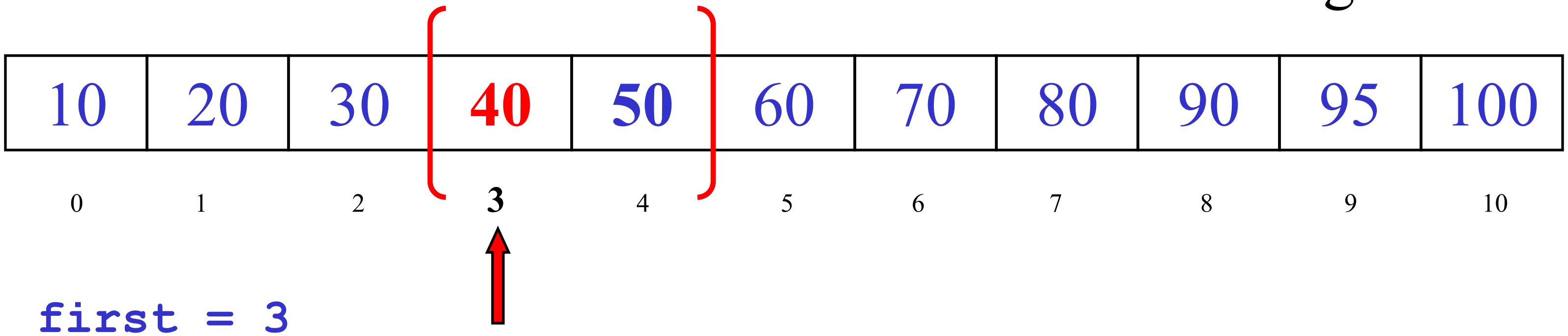
10	**20**	**30**	**40**	**50**	60	70	80	90	95	100
0	1	**2**	3	4	5	6	7	8	9	10

```
first = 0
last  = 4
mid   = (0 + 4) / 2 = 2
40 > list[2] => search the upper half
      first = mid + 1 = 3
```

first	last	first <= last	mid	target ? list[mid]
0	10	True	5	40 < 60
0	4	True	2	40 > 30

2. Using Binary Search, which elements in the array below are compared to the target?

A. target = 40



10	20	30	40	50	60	70	80	90	95	100
0	1	2	3	4	5	6	7	8	9	10

```
first = 3
last  = 4
mid = (3 + 4)/2 = 3
40 == list[3]
```

first	last	first <= last	mid	target ? list[mid]
0	10	True	5	40 < 60
0	4	True	2	40 > 30
3	4	True	3	40 == 40

2. Using Binary Search, which elements in the array below are compared to the target?

A. target = 40

10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**95**	**100**
0	1	2	3	4	5	6	7	8	9	10

ANSWER:

60, 30, 40

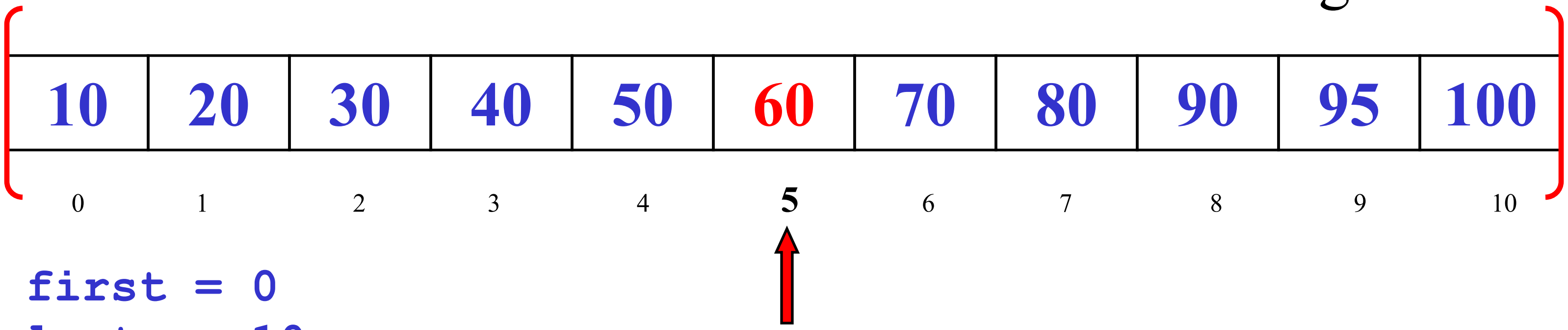
2. Using Binary Search, which elements in the array below are compared to the target?

B. target = 75

10	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**95**	**100**
0	1	2	3	4	5	6	7	8	9	10

2. Using Binary Search, which elements in the array below are compared to the target?

B. target = 75



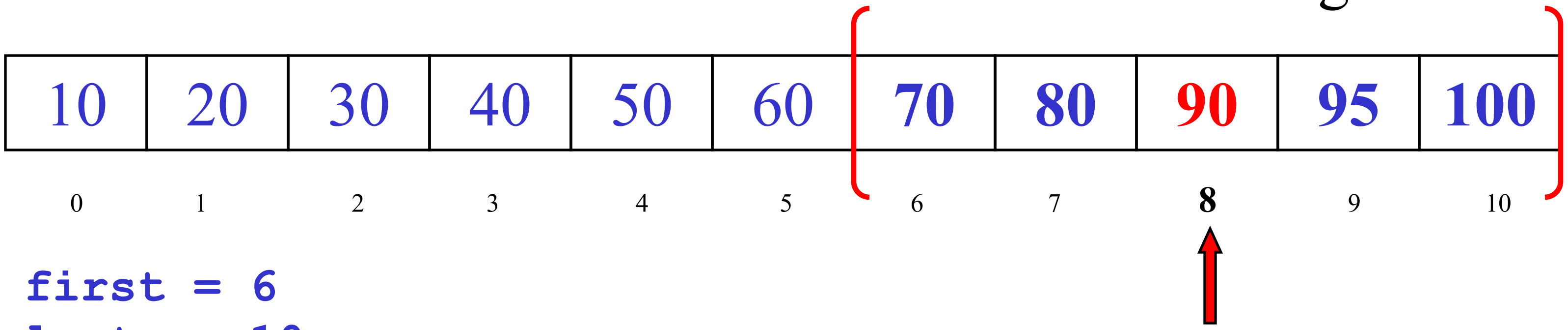
10	20	30	40	50	60	70	80	90	95	100
0	1	2	3	4	**5**	6	7	8	9	10

```
first = 0
last  = 10
mid = (first + last)/2 = 5
75 > list[5] => search the upper half:
       first = mid + 1 = 6
```

first	last	first <= last	mid	target ? list[mid]
0	10	True	5	75 > 60

2. Using Binary Search, which elements in the array below are compared to the target?

B. target = 75



10	20	30	40	50	60	70	80	90	95	100
0	1	2	3	4	5	6	7	8	9	10

```
first = 6
last  = 10
mid = (6 + 10)/2 = 8
75 < list[8] => search the lower half
      last = mid - 1 = 7
```

first	last	first <= last	mid	target ? list[mid]
0	10	True	5	75 > 60
6	10	True	8	75 < 90

2. Using Binary Search, which elements in the array below are compared to the target?

B. target = 75

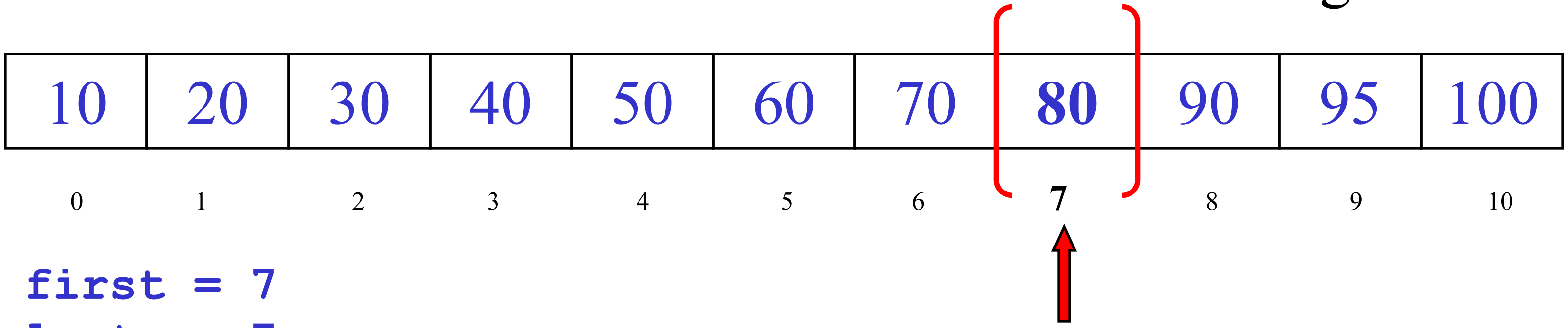
10	20	30	40	50	60	**70**	**80**	90	95	100
0	1	2	3	4	5	**6**	7	8	9	10

```
first = 6
last  = 7
mid = (6 + 7)/2 = 6
75 > list[6] => search the upper half
       first = mid + 1 = 7
```

first	last	first <= last	mid	target ? list[mid]
0	10	True	5	75 > 60
6	10	True	8	75 < 90
6	7	True	6	75 > 70

2. Using Binary Search, which elements in the array below are compared to the target?

B. target = 75



10	20	30	40	50	60	70	80	90	95	100
0	1	2	3	4	5	6	7	8	9	10

```
first = 7
last  = 7
mid = (7 + 7)/2 = 7
75 < list[7] => search the lower half
      last = mid - 1 = 6
```

first	last	first <= last	mid	target ? list[mid]
0	10	True	5	75 > 60
6	10	True	8	75 < 90
6	7	True	6	75 > 70
7	7	True	7	75 < 80

2. Using Binary Search, which elements in the array below are compared to the target?

B. target = 75

10	20	30	40	50	60	70	**80**	90	95	100
0	1	2	3	4	5	6	**7**	8	9	10

```
first = 7
last  = 6
first = 7 > 6 = last => STOP:target not found!
```

first	last	first <= last	mid	target ? list[mid]
0	10	True	5	75 > 60
6	10	True	8	75 < 90
6	7	True	6	75 > 70
7	7	True	7	75 < 80
7	6	False		

2. Using Binary Search, which elements in the array below are compared to the target?

B. target = 75

10	20	30	40	50	60	70	80	90	95	100
0	1	2	3	4	5	6	7	8	9	10

ANSWER:

60, 90, 70, 80

3.

A. The ____________________ sort finds the smallest element from the unsorted sub-list and swaps it with the element at the beginning of the unsorted data.

3.

A. The selection sort finds the smallest element
from the unsorted sub-list and swaps it with the element at the beginning of the unsorted data.

3.

B. The efficient version of the ____________ sort does not exchange elements.

3.

B. The efficient version of the insertion sort does not exchange elements.

It is based on shifting elements!

4. An array contains the elements shown below.
Show the contents of the array after two passes of the

A. Insertion sort algorithm.

11, 85, 3, 18, 20, 15, 5, 12, 70, 8, 50

4. An array contains the elements shown below.
Show the contents of the array after two passes of the

A. Insertion sort algorithm.

11, | 85, 3, 18, 20, 15, 5, 12, 70, 8, 50

Pass #1: Insert 85 (the array does not change!)

11, 85, | 3, 18, 20, 15, 5, 12, 70, 8, 50

4. An array contains the elements shown below. Show the contents of the array after two passes of the

A. Insertion sort algorithm.

```
11,| 85,  3, 18, 20, 15,  5, 12, 70,  8, 50
```

Pass #1: Insert 85 (the array does not change!)

```
11, 85,|  3, 18, 20, 15,  5, 12, 70,  8, 50
```

Pass #2: Insert 3

$$\underbrace{3,\ 11,\ 85,}_{\text{Sorted}} \;|\; \underbrace{18,\ 20,\ 15,\ 5,\ 12,\ 70,\ 8,\ 50}_{\text{Unsorted}}$$

4. An array contains the elements shown below.
Show the contents of the array after two passes of the

B. Selection sort algorithm.

```
11, 85,  3, 18, 20, 15,  5, 12, 70,  8, 50
```

4. An array contains the elements shown below.
Show the contents of the array after two passes of the

B. Selection sort algorithm.

11, 85, 3, 18, 20, 15, 5, 12, 70, 8, 50

Pass #1: swap 3 and 11

3, | 85, 11, 18, 20, 15, 5, 12, 70, 8, 50

4. An array contains the elements shown below.
Show the contents of the array after two passes of the

B. Selection sort algorithm.

```
11, 85,  3, 18, 20, 15,  5, 12, 70,  8, 50
```

Pass #1: swap 3(smallest) and 11

```
 3,| 85, 11, 18, 20, 15,  5, 12, 70,  8, 50
```

Pass #2: swap 5(smallest) and 85

$$\underbrace{3,\ \ 5,}_{\text{Sorted}}\ \Big|\ \underbrace{11,\ 18,\ 20,\ 15,\ 85,\ 12,\ 70,\ \ 8,\ 50}_{\text{Unsorted}}$$

5. Predict the output (no computers please, and show how did you get the answer).

```
int size = 16;
int score[100] =
     {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8};
int a[11] = {0};


for (int i = 0; i < size; i++)
    a[score[i]]++;     // a[score[i]] = a[score[i]] + 1;

for (int i = 0; i < 11; i++)
    if (a[i])          // a[i] != 0
        cout << i << " " <<  a[i] << endl;
```

5. Predict the output (no computers please, and show how did you get the answer).

```
int size = 16;
int score[100] =
     {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8};
int a[11] = {0};


for (int i = 0; i < size; i++)
    a[score[i]]++;     // a[score[i]] = a[score[i]] + 1;

for (int i = 0; i < 11; i++)
    if (a[i])          // a[i] != 0
        cout << i << " " <<  a[i] << endl;
```

```
score: {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8}
    a: { 0, 0, 0,  0,  0, 0, 0, 0,  0, 0, 0 }
```

a[0] – is a counter for **0**s, a[1] – is a counter for **1**s, … and so on, the last one is a[10] – a counter for **10**s

5. Predict the output (no computers please, and show how did you get the answer).

```
int size = 16;
int score[100] =
     {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8};
int a[11] = {0};


for (int i = 0; i < size; i++)
    a[score[i]]++;     // a[score[i]] = a[score[i]] + 1;

for (int i = 0; i < 11; i++)
    if (a[i])          // a[i] != 0
        cout << i << " " <<  a[i] << endl;
```

```
score: {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8}
    a: { 1, 0, 0,  0,  0, 1, 0, 2,  2, 6, 4 }
```

a[10] – is equal to 4, meaning that there are 4 scores equal to 10

5. Predict the output (no computers please, and show how did you get the answer).

```
int size = 16;
int score[100] =
     {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8};
int a[11] = {0};


for (int i = 0; i < size; i++)
    a[score[i]]++;     // a[score[i]] = a[score[i]] + 1;

for (int i = 0; i < 11; i++)
    if (a[i])               // a[i] != 0
        cout << i << " " <<  a[i] << endl;
```

```
score: {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8}
    a: { 1, 0, 0,  0,  0, 1, 0, 2,  2, 6, 4 }
```

The second loop displays the non-zero scores and their counters.

What would a better name for a[] be?
Any applications?

```
for (int i = 0; i < 11; i++)
    if (a[i])                // a[i] != 0
        cout << i << " " <<  a[i] << endl;
```

```
score: {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8}
    a: { 1, 0, 0,  0, 0, 1, 0, 2,  2, 6, 4 }
         0  1  2   3  4  5  6  7   8  9 10   // INDEX
```

```
 0      1
 5      1
 7      2
 8      2
 9      6
10      4
```

What would a better name for a[] be? **frequencyArray[]**

Any applications? **Histogram**

```
for (int i = 0; i < 11; i++)
    if (a[i])              // a[i] != 0
        cout << i << " " <<  a[i] << endl;
```

```
score: {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8}
    a: { 1, 0, 0,  0, 0, 1, 0, 2,  2, 6, 4 } // Frequency Array
         0  1  2   3  4  5  6  7   8  9 10   // INDEX
```

```
  0      1 | *
  5      1 | *
  7      2 | **
  8      2 | **
  9      6 | ******
 10      4 | ****
          //  Histogram
```

Any applications? Bucket Sort

Bucket Sort can be used on
arrays of integers within
a small range.

```
for (int i = 0; i < 11; i++)
    if (a[i])               // a[i] != 0
        cout << i << " " <<  a[i] << endl;
```

```
score: {10, 9, 9, 10, 9, 8, 9, 7, 10, 5, 0, 9, 9, 7, 10, 8}
    a: { 1, 0, 0,  0, 0, 1, 0, 2,  2, 6, 4 } //
         0  1  2   3  4  5  6  7   8  9 10   // INDEX
```

```
k = 0;
for (int i = 0; i < 11; i++)
{
    while (a[i])
    {
        score[k++] = i;
        a[i]--;
    }
}
```

```
score: {0, 5, 7, 7, 8, 8, 9, 9, 9, 9, 9, 9, 10, 10, 10, 10}
    // SORTED! "Bucket Sort" or "Sorting by counting"
```