**Question 1:**

1. If we assume that every “Wooden blocks toy’s” color as same numbers like blue = 1, red = 2. We can use quick sort to these numbers to separate in some place of the array. So the space complexity of quicksort is O(log n) & the time complexity of quicksort is O(n log n).
2. The same function from the above problem will also work for this case. We can use quicksort to these numbers to separate in some place of the array. So the space complexity of quicksort is O(log n) & the time complexity of quicksort is O(n log n).
3. The same function from the above problem will also work for this case. We can use quicksort to these numbers to separate in some place of the array. So the space complexity of quicksort is O(log n) & the time complexity of quicksort is O(n log n).

**Question 2:**

**A)**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Median of three**: a[0] = 1, a[(0 + 8)/2] = 5, a[8] = 9; In this case pivot is 5 (so it’s good self call)

| 1 | 2 | 3 | 4 | 9 | 6 | 7 | 8 | 5 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | j | i |  |  | j |  |

| 1 | 2 | 3 | 4] | 5 | [6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | REC1 |  |  |  |  | REC2 |  |  |

REC1:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Making bad self call**: Pivot is 4

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | j | i |  |  |  |  |  |

| 1 | 2 | 3] | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | REC3 |  |  |  |  |  |  |  |

REC3:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Good self-call**: pivot is 2

| 1 | 3 | 2 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| j | i |  |  |  |  |  |  |  |

| 1] | 2 | [3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

REC2:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Good self call**: Pivot 8

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | j | i |  |

| 1 | 2 | 3 | 4 | 5 | 6 | 7] | 8 | [9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | REC4 |  |  |  |

REC 4:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Pivot**:7

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | j | i |  |  |

| 1 | 2 | 3 | 4 | 5 | 6] | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

Now Array is sorted and every recursive call is called alternating between bad and self-good calls.

**B)**

| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Good self call**: Pivot is 3

| 8 | 7 | 6 | 5 | 4 | 9 | 2 | 1 | 3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i |  |  |  |  |  |  | j |  |

| 1 | 7 | 6 | 5 | 4 | 9 | 2 | 8 | 3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i |  |  |  |  | j |  |  |

| 1 | 2 | 6 | 5 | 4 | 9 | 7 | 8 | 3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | j | i |  |  |  |  |  |  |

| [1 | 2] | 3 | [5 | 4 | 9 | 7 | 8 | 6] |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | REC2 |  |  |  |

REC2:

| 1 | 2 | 3 | 5 | 4 | 9 | 7 | 8 | 6 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Making bad self call**: Pivot is 4

| 1 | 2 | 3 | 5 | 6 | 9 | 7 | 8 | 4 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | j | i |  |  |  |  |  |

| 1 | 2 | 3 | 4 | [6 | 9 | 7 | 8 | 5] |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | REC3 |  |  |  |

REC3:

| 1 | 2 | 3 | 4 | 6 | 9 | 7 | 8 | 5 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Making good self call**: Pivot is 7

| 1 | 2 | 3 | 4 | 6 | 9 | 5 | 8 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | i | j |  |  |

| 1 | 2 | 3 | 4 | 6 | 5 | 9 | 8 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | j | i |  |  |

| 1 | 2 | 3 | 4 | 6 | 5] | 7 | [8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | REC4 |  |  |  |  |

REC4:

| 1 | 2 | 3 | 4 | 6 | 5 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Making good self call**: Pivot is 5

| 1 | 2 | 3 | 4 | 6 | 5 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | j | i |  |  |  |  |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

Now Array is sorted and every recursive call is called alternating between bad and self-good calls.

**C)**

| 9 | 1 | 8 | 2 | 7 | 3 | 6 | 4 | 5 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Median of three**: a[0] = 9, a[(0 + 8)/2] = 7, a[8] = 5; In this case pivot is 7 (so it’s good self call)

| 9 | 1 | 8 | 2 | 5 | 3 | 6 | 4 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i |  |  |  |  |  |  | j |  |

| 4 | 1 | 8 | 2 | 5 | 3 | 6 | 9 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | i |  |  |  | j |  |  |

| 4 | 1 | 6 | 2 | 5 | 3 | 8 | 9 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | j | i |  |  |

| [4 | 1 | 6 | 2 | 5 | 3] | 7 | [9 | 8] |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | REC1 |  |  |  | REC2 |  |

REC1:

| 4 | 1 | 6 | 2 | 5 | 3 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Making bad self call**: Pivot is 2

| 4 | 1 | 6 | 3 | 5 | 2 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | j |  |  |  |  |  |  |  |

| 1 | 4 | 6 | 3 | 5 | 2 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| j | i |  |  |  |  |  |  |  |

| 1] | 2 | [6 | 3 | 5 | 4] | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | REC3 |  |  |  |  |  |

REC3:

| 1 | 2 | 6 | 3 | 5 | 4 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Median of three**: a[0] = 6, a[(0 + 3)/2] = 3, a[3] = 4; In this case pivot is 4(so it’s good self call)

| 1 | 2 | 6 | 3 | 5 | 4 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | i | j |  |  |  |  |  |

| 1 | 2 | 3 | 6 | 5 | 4 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | j | i |  |  |  |  |  |

| 1 | 2 | [3] | 4 | [5 | 6] | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | REC4 |  |  |  |  |

REC4:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

Pivot 5

| 1 | 2 | 3 | 4 | 6 | 5 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

| 1 | 2 | 3 | 4 | 6 | 5 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | j | i |  |  |  |  |

| 1 | 2 | 3 | 4 | 5] | [6 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

No recursive call

REC 2:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Median of three**: a[0] = 9, a[(0 + 1)/2] = 9, a[1] = 8; In this case pivot is 9(so it’s good self call)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | j | i |

Now Array is sorted and every recursive call is called alternating between bad and self-good calls.

**D)**

| 5 | 1 | 4 | 2 | 3 | 9 | 7 | 6 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**BAD-self call**: Pivot = 8

| 5 | 1 | 4 | 2 | 3 | 9 | 7 | 6 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | i |  | j |  |

| 5 | 1 | 4 | 2 | 3 | 6 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | j | i |  |

| 5 | 1 | 4 | 2 | 3 | 6 | 7] | 8 | [9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | REC1 |  |  |  |  |  |

REC1

| 5 | 1 | 4 | 2 | 3 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Good self call**: Pivot is 4

| 5 | 1 | 7 | 2 | 3 | 6 | 4 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i |  |  |  | j |  |  |  |  |

| 3 | 1 | 7 | 2 | 5 | 6 | 4 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | i | j |  |  |  |  |  |

| 3 | 1 | 2 | 7 | 5 | 6 | 4 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | j | i |  |  |  |  |  |

| 3 | 1 | 2] | 4 | [5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | REC2 |  |  |  | REC3 |  |  |  |

REC2:

| 3 | 1 | 2 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Making bad self call**: Pivot is 1

| 3 | 2 | 1 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i,j |  |  |  |  |  |  |  |  |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

REC3

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Making good self call**: Pivot is 6

| 1 | 2 | 3 | 4 | 5 | 7 | 6 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | j | i |  |  |  |

| 1 | 2 | 3 | 4 | 5] | 6 | [7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

Now Array is sorted and every recursive call is called alternating between bad and self-good calls.

**Question 3:**

1. **K = 5**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Median of three**: a[0] = 1, a[(0 + 8)/2] = 5, a[8] = 9; In this case pivot is 5 (so it’s good self call)

| 1 | 2 | 3 | 4 | 9 | 6 | 7 | 8 | 5 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | j | i |  |  | j |  |

| 1 | 2 | 3 | 4] | 5 | [6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | L |  |  | E |  | G |  |  |

K = 5 so quickSelect(E, 1) will call, and since 5 is the only element and it will return 5

**B) K = 3**

| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Good self call**: Pivot is 3

| 8 | 7 | 6 | 5 | 4 | 9 | 2 | 1 | 3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i |  |  |  |  |  |  | j |  |

| 1 | 7 | 6 | 5 | 4 | 9 | 2 | 8 | 3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i |  |  |  |  | j |  |  |

| 1 | 2 | 6 | 5 | 4 | 9 | 7 | 8 | 3 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | j | i |  |  |  |  |  |  |

| [1 | 2] | 3 | [5 | 4 | 9 | 7 | 8 | 6] |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| L |  | E |  | G |  |  |  |  |

K = 3 so quickSelect(E, 1) will call, and since 3 is the only element and it will return 3

**C) k = 8**

| 9 | 1 | 8 | 2 | 7 | 3 | 6 | 4 | 5 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Median of three**: a[0] = 9, a[(0 + 8)/2] = 7, a[8] = 5; In this case pivot is 7 (so it’s good self call)

| 9 | 1 | 8 | 2 | 5 | 3 | 6 | 4 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i |  |  |  |  |  |  | j |  |

| 4 | 1 | 8 | 2 | 5 | 3 | 6 | 9 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | i |  |  |  | j |  |  |

| 4 | 1 | 6 | 2 | 5 | 3 | 8 | 9 | 7 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | j | i |  |  |

| [4 | 1 | 6 | 2 | 5 | 3] | 7 | [9 | 8] |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | L |  |  | E | G |  |

Since k = 8, so quickSelect(G, 2) will call

quickSelect(G, 2):

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Median of three**: a[0] = 9, a[(0 + 1)/2] = 9, a[1] = 8; In this case pivot is 9(so it’s good self call)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | j | i |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8] | [9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | L | E |

K = 8 so quickSelect(E, 2) will call, and since 8 is the only element and it will return 8

**D) K = 5**

| 5 | 1 | 4 | 2 | 3 | 9 | 7 | 6 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**BAD-self call**: Pivot = 8

| 5 | 1 | 4 | 2 | 3 | 9 | 7 | 6 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | i |  | j |  |

| 5 | 1 | 4 | 2 | 3 | 6 | 7 | 9 | 8 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | j | i |  |

| 5 | 1 | 4 | 2 | 3 | 6 | 7] | 8 | [9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | L |  |  |  | E | G |

Since k = 5, so quickSelect(L, 7) will call

quickSelect(L, 7)

| 5 | 1 | 4 | 2 | 3 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Good self call**: Pivot is 4

| 5 | 1 | 7 | 2 | 3 | 6 | 4 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i |  |  |  | j |  |  |  |  |

| 3 | 1 | 7 | 2 | 5 | 6 | 4 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | i | j |  |  |  |  |  |

| 3 | 1 | 2 | 7 | 5 | 6 | 4 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | j | i |  |  |  |  |  |

| 3 | 1 | 2] | 4 | [5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | L |  | E |  | G |  |  |  |

Since k = 5, so quickSelect(G,3) will call

quickSelect(G,3):

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

**Making good self call**: Pivot is 6

| 1 | 2 | 3 | 4 | 5 | 7 | 6 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | j | i |  |  |  |

| 1 | 2 | 3 | 4 | 5] | 6 | [7 | 8 | 9 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

L E G

K = 5 so quickSelect(L, 1) will call, and since 5 is the only element and it will return 5

**Question 4:**

2n/3 (normal division)

This means ~33.3 percent of the calls are good self-call. The green place is good pivots, red means bad pivots zone.

| 1 | 2 | 3 | 4 | 5 | 7 | 8 | 9 | 10 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bad |  |  | Good |  |  | Bad |  |

**a)**

A good case occurs if every self-call is good

* The height of the recursion tree is one less than the length of the descending sequence n, (2/3)n, (2/3)2n, . . ., 1, 0. By exercise, this length is ≤ 1 +log3/2n, which is O(log n)(Note: Length of the path is 1 less than num nodes on that path.)
* Good self-calls don’t occur every time –but we know the probability of a good self-call is 1/3.
* If all self-calls are good, the height of the tree is m = 1+log3/2n. This implies that in general, recursion ends after m good self calls.

**b)** Not the same. Because the good case occurring chance is less. The above shows that it affects the height of the tree. Also the probability of good call is more less than previous