Algorithm Design & Analysis

Assignment-5 (Recursion, Divide-and-Conquer Strategy, Greedy Method)

1. You are given *R* red marbles and *B* blue marbles. Your task is to arrange the *R* + *B* marbles in a line such that certain restrictions are satisfied (see below). You print all possible arrangements under the given restrictions, and the total count of arrangements possible.

Part 1

In this part, the restriction is that no two red marbles may be placed consecutively. Write a single recursive function to print the arrangements and return the count of possibilities. Do not use any global or static variables. Place the available marbles one by one in an array of size *R* + *B*. When *u* red and *v* blue marbles are placed, find out the options (red and/or blue) at the (*u* + *v* + 1)-st position. For each available option, recursively compute the acceptable configurations with *u* + *v* + 1 marbles placed in the array.

Part 2

In this part, two red marbles may appear in consecutive positions, but three or more red marbles are not allowed to come consecutively. Write a second recursive function following the same line of programming logic as in Part 1.

Sample output

```
Enter number of red marbles (R)  : 2
Enter number of blue marbles (B) : 4

Part 1
Rbrbbb      rbbrbb          rbbbrb          rbbbbr          brbrbb
Brbbrb      brbbbr          bbrbrb          bbrbbr          bbbrbr

Total number of possibilities is 10

Part 2
Rrbbbb      rbrbbb          rbbrbb          rbbbrb          rbbbbr
Brrbbb      brbrbb          brbbrb          brbbbr          bbrrbb
Bbrbrb      bbrbbr          bbbrrb          bbbrbr          bbbbrr

Total number of possibilities is 15
```

2. Write a program to find the k^{th} maximum from a set of *n* distinct numbers where the average case time complexity is O(*n*). In this program, use *randomized partition* method for selecting the pivot element.

Sample Input:

57 50 21 13 32 1 9 45 26 17 43 29 17 8 12 35 98 52 78 93
k = 9

Sample Output: 35

3. Given an array of *n* elements, find the k^{th} smallest element with worst case complexity being linear time (i.e. O(*n*)). Assumption is that the array consists of non-repeating elements.

   Sample Input:

   67 54 22 16 32 1 9 45 26 17 43 29 19 8 12 35 98 52 78 83
   k = 7

   Sample Output: 19

4. You are given an array of n jobs. Each job has a predefined deadline and profit. If you finish the job on or before its deadline, you are awarded with a profit associated with that job. A single processor is assigned, so only one job can be scheduled at a time. Each job takes one unit of time. Write an $O(n^2)$-time greedy function to find the subset of jobs that maximizes the total profit.

   Sample Input:

Job	Profit	Deadline
A	10	2
B	19	1
C	77	2
D	35	2
E	15	3

   Sample Output:

   Sequence of jobs for earning maximum profit: D, C, E

   Maximum profit: 127