Solution - 1

1.

A1: (a) (c) (d) (g)

A2: (a) (b) (c) (d) (f) (g) (h)

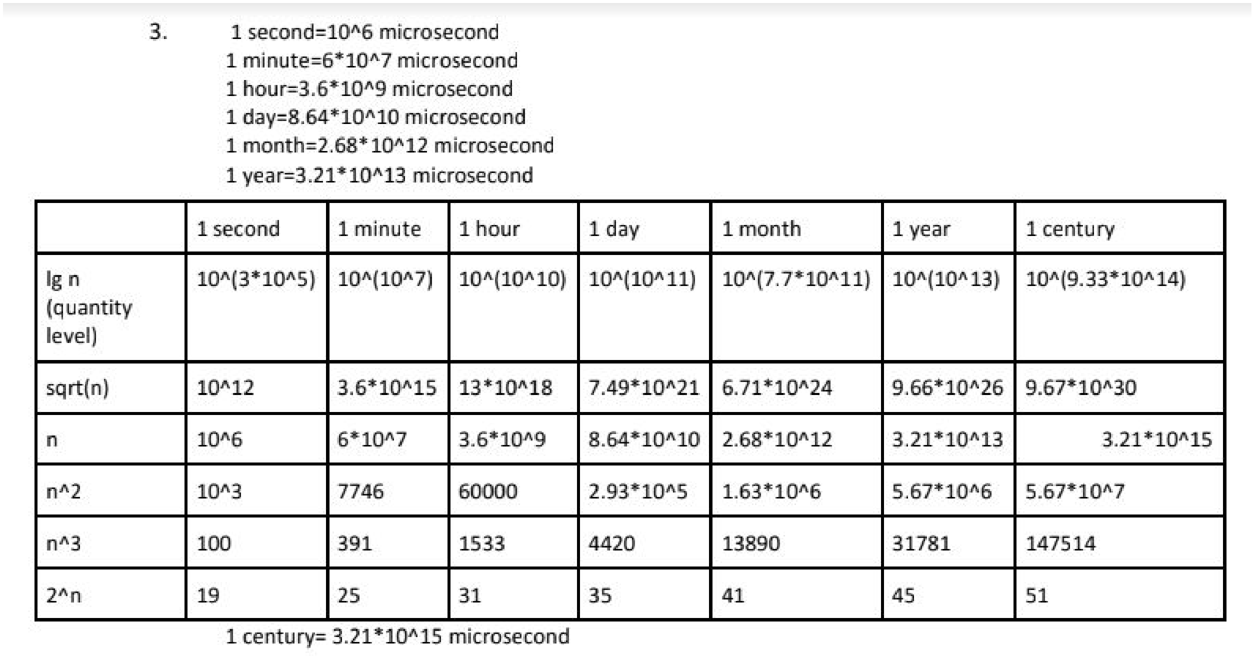
A3: (b) (f) (h)

A4: (b) (e) (f) (h)

A5: (c)

2. a), f)

3.



3. 1 second=10^6 microsecond
1 minute=6*10^7 microsecond
1 hour=3.6*10^9 microsecond
1 day=8.64*10^10 microsecond
1 month=2.68*10^12 microsecond
1 year=3.21*10^13 microsecond

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
lg n (quantity level)	10^(3*10^5)	10^(10^7)	10^(10^10)	10^(10^11)	10^(7.7*10^11)	10^(10^13)	10^(9.33*10^14)
sqrt(n)	10^12	3.6*10^15	13*10^18	7.49*10^21	6.71*10^24	9.66*10^26	9.67*10^30
n	10^6	6*10^7	3.6*10^9	8.64*10^10	2.68*10^12	3.21*10^13	3.21*10^15
n^2	10^3	7746	60000	2.93*10^5	1.63*10^6	5.67*10^6	5.67*10^7
n^3	100	391	1533	4420	13890	31781	147514
2^n	19	25	31	35	41	45	51

1 century= 3.21*10^15 microsecond

4.

The algorithm does not correctly print out all the items that appear more than once. It also prints all items in the array because we initialize j to the value of i.

Counterexample:

A = [1, 2, 3]

Output: 1 2 3

Explanation: None of the printed values appeared more than once.

5.

The algorithm does not compute power correctly, because the value of y is not properly updated.

Counterexample:

POW(2, 3)

Output: The program will never terminate because of the infinite loop.

6. The algorithm is not correct. Value of y is not getting decremented and so the while loop will run infinite number of times. A counter example is x =1 and y=1. Since y > 0 we enter the while loop. Since y is odd we calculate e. After this y is still one so the while loop is executed once more. These steps will get repeated infinite number of times. Hence, the algorithm will not terminate and is not correct.

7. Big Oh – n^2

Theta – n^2

Omega – n^2

Small Oh – n^k (k > 2) (b) Big Oh - n

Theta - n

Omega - n

Small Oh – n^k (k > 1)

8.

The function *Maxmium* prints the following for the given input:
1 0
2 1
4 3
4 2

9.

```
def function(names):
    // insertion sort
    for i = 2 to names.length:
        key = A[j]
        i = [j - 1]
        while i > 0 and A[i] > key:
            A[i + 1] = A[i]
            i = i – 1
        A[i + 1] = key

    // count
    result = 1
    for i = 2 to names.length:
        if names[i] != names[i – 1]:
        result ++
    return result
```

Running time: O(n*n) theta(n*n) omega(n*n)