Q1. Insertion sort $8n^2$
Merge sort $64n\lg n$

Insertion sort beats Merge sort:

$8n^2 < 64n\lg n$
$n^2 < 8n\lg n$ Divide by 8
$n < 8\lg n$ Divide by n
$n/8 < \lg n$ Divide by 8

Using MS Excel: $n = 43$

Q2. A — $100n^2$
B — $2n$

A beats B:

$100n^2 < 2n$

Using MS Excel: $n = 15$

Q3. $F_1(n) = n^2$

$F_2(n) = n^2 + 100n$

$F_3(n) = n$ n is odd a

$= n^3$ n is even b

$F_4(n) = n$ $n <= 100$ a

$= n^3$ $n > 100$ b

f_j	f_k	$f_j(n)$ is $O(f_k(n))$	$f_j(n)$ is $\Omega(f(k(n))$
F_1	F_2	Yes	Yes
F_1	F_3 a	No	Yes
	F_3 b	Yes	No
F_1	F_4 a	No	Yes
	F_4 b	Yes	No
F_2	F_1	Yes	Yes
F_2	F_3 a	No	Yes
	F_3 b	Yes	No
	F_4 a	No	Yes
	F_4 b	Yes	No
F_3 a	F_1	Yes	No
	F_2	Yes	No
	F_4 a	Yes	Yes
	F_4 b	Yes	No
F_3 b	F_1	No	Yes
	F_2	No	Yes
	F_4 a	No	No Yes
	F_4 b	Yes	Yes
F_4 a	F_1	Yes	No
	F_2	Yes	No
	F_3 a	Yes	Yes
	F_3 b	Yes	No
F_4 b	F_1	No	Yes
	F_2	No	Yes
	F_3 a	No	Yes
	F_3 b	Yes	Yes

Q4. Using MS Excel:

$(1/3)^n < 17 < \log(\log n) < \log(n) < \sqrt{n} < \log^2(n) < n$
$< \sqrt{n}\log^2 n < n\log n < (3/2)^n$

Q5a.

```
for i=1 to n do                                   O(n)
    for j=1 to n do begin                         O(n)
        c[i,j] = 0                                O(1)
        for k=1 to n do                           O(n)
            c[i,j] = c[i,j] + A[i,k] * B[k,j]     O(1)
    end
```

O(1) O(1) ~~O(n)~~ O(1)

~~k loop: O(n) · O(1) = O(n)~~
~~j loop: O(n) · [O(1) · O(n)] = O(n²)~~
~~i loop: O(n) · O(n²) = O(n³)~~

~~i loop: O(n)~~
~~j loop:~~

~~k loop: O(n) × O(1) = O(n)~~
~~j loop: [O(1) + O(n)]~~

~~k loop: O(n) × O(1) = O(n)~~
~~j loop: O(n) × O(n) = O(n²)~~

---

k loop: $O(n) \times O(1) = O(n)$
j loop: $O(n) \times [O(1) + O(n)] = O(n) \times O(n) = O(n^2)$
i loop: $O(n) \times O(n^2) = O(n^3)$

Q5b. for $i = 1$ to $(n-1)$ do
for $j = i+1$ to n do
begin
for $k = 1$ to j do $\Big\}$ $\sum_{k=1}^{j} 1 = j$
$O(1)$
end

Complexity:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} j$$

But $\sum_{j=i+1}^{n} j = \sum_{i=1}^{n} j - \sum_{i=1}^{i} j$

$$= \left[\frac{n(n+1)}{2}\right] j - \left[\frac{i(i+1)}{2}\right] j$$

$$= \left[\frac{n(n+1)}{2} - \frac{i(i+1)}{2}\right] j$$

Complexity: $\sum_{i=1}^{n-1} \left(\frac{n(n+1)}{2} - \frac{i(i+1)}{2}\right) j$

$$= \frac{n-1}{2}\left(n^2 + n - i^2 - i\right) j$$

$$= \frac{j}{2}\left[n^3 + n^2 - ni^2 - ni - n^2 - n + i^2 + i\right]$$

$$= \frac{j}{2}\left[n^3 - (n-1)i^2 - (n-1)i\right]$$

So, $O(n^3)$

Q5c.

```
for i=1 to n-1 do
  if odd(i) then begin
    for j=i to n do          } i to n
      x=x+1          O(1)    }
    for j=1 to i do          } 1 to i
      y=y+1          O(1)    }
```

for 1 to n, $O(n)$

if statement: $\left(\frac{n-1}{2}\right) \times O(n) = O(n^2)$

i Loop: 1 to n-1, So

Complexity: $O(n) \times O(n^2) = O(n^3)$

Q6.

```
count = 0
x = 2
while x<n do begin
  x = 2*x
  count = count+1
end
cout << count
```

n is +ve power of 2

n	2	4	8	16
count	0	1	2	3

value of n: value when loop terminates

$n = 2^{count+1}$

$count+1 = \log_2 n$

$count = \log_2 n - 1$

Q7. Because the problem takes $f(n)$ milliseconds, so time t is ~~1000 for sec~~

~~1000~~	1000	sec	10^3
	1000×60	min	6×10^4
	$1000 \times 60 \times 24$	day	1.44×10^6
	$1000 \times 60 \times 24 \times 30$	month	4.32×10^7

Assuming every month to be of 30 days

	1 sec	1 min	1 day	1 month
$\lg(n)$	2^{10^3}	$2^{6\times10^4}$	$2^{1.44\times10^6}$	$2^{4.32\times10^7}$
$\sqrt{n}$	$(10^3)^2$	$(6\times10^4)^2$	$(1.44\times10^6)^2$	$(4.32\times10^7)^2$
n	10^3	6×10^4	1.44×10^6	4.32×10^7
$n\lg(n)$	140/141	4895		
n^2	3.16×10^1	2.45×10^2	1.2×10^3	6.57×10^3
n^3	1×10^1	3.9×10^1	1.13×10^2	3.51×10^2
2^n	9.97	1.59×10^1	2×10^1	2.54×10^1
$n!$	6	8	9	11

→ Approximate

→ square root

→ cube root

→ $\log_2$

→ Inverse factorial

~~$n\lg(n) = t$~~ ~~$2^n = t$~~

~~$\lg(n) = t/n$~~ ~~$n \cdot 2 = t$~~

Using MS Excel

Q8. Selection sort:

```
for i=1 to n-1 do begin
    min = i
    for j = i+1 to n do begin
        if element_j < element_min then
            min = j
        swap element_i with element_min
end
```

First iteration n-1 comparisons
Second " n-2 "
Last " 1 "

Total comparisons: $(n-1)+(n-2)+\cdots+2+1$

$$= \frac{n(n-1)}{2}$$

So, best case or worst case, both are $\theta(n^2)$ as the number of comparisons remain the same.

Q9. Iterative binary search

```
low = 0
high = n-1
while low <= high do begin
    mid = (low + high)/2
    if A[mid] > value
        high = mid - 1
    else if A[mid] < value
        low = mid + 1
    else
        return mid
end
return 'not found'
```

Worst case: Item not found

~~Through each iteration~~

After 1st iteration items remaining : $n/2$
" 2nd " " " : $n/4$
" k " " " : $n/2^k$

Worst case: $n/2^k$ items left

that means $n \leq 2^k$

So, complexity is $O(\lg(n))$

Q10. Because we need to come up with the ~~algorithm~~ $O(n \lg(n))$, it is advised to use merge sort for sorting the array and then binary search to check if integers ~~xxxx~~ exists in the set.

Merge sort: $O(n \log(n))$
Binary sort: $O(\lg(n))$

Q11. The running time of algorithm ~~a means~~ A is at least $O(n^2)$ means that ~~$\Theta(n^2)$ is the lower bound~~ the algorithm A is the upper bound for $O(n^2)$, or we can say that $O(n^2)$ is the lower bound for A. This statement holds true for any running time and tells us nothing that can be used to judge algorithm A.

Q12.

		O	Ω	Θ
n^k	c^n	Yes	No	No
2^n	$2^{n/2}$	No	Yes	No
$\lg(n!)$	$\lg(n^n)$	Yes	Yes	Yes

A __ $O(B)$
A __ $\Omega(B)$
A __ $\Theta(B)$