Problem 3.1 Asymptotic Analysis (8 points) Considering the following pairs of functions f and g, show for each pair whether or not it belongs to each of the relations f ∈ Θ(g), f ∈ O(g), f ∈ o(g), f ∈ Ω(g), f ∈ ω(g), g ∈ Θ(f), g ∈ O(f), g ∈ o(f), g ∈ Ω(f), or g ∈ ω(f).

(a) (2 points) f(n) = 9n and g(n) = 5n 3

using computation with limits:

$$\lim_{n \to +\infty} \frac{9n}{5n^3} = \lim_{n \to +\infty} \frac{9}{5n^2} = 0$$

f ∈ Θ(g)	No
f ∈ O(g),	Yes
f ∈ o(g),	Yes
f∈ Ω(g)	No
f ∈ ω(g)	No

now, the other way around:

$$\lim_{n \to +\infty} \frac{5n^3}{9n} = \lim_{n \to +\infty} \frac{5n^2}{9} = \infty$$

g∈ Θ(f)	No
g ∈ O(f),	No
g ∈ o(f),	No
g∈ Ω(f)	Yes
g ∈ ω(f)	Yes

(b) (2 points) f(n) = 9n 0.8 + 2n 0.3 + 14 log n and g(n) = √ n,

using computation with limits:

$$\lim_{n \to +\infty} \frac{9n^{0.8} + 2n^{0.3} + 14logn}{n^{0.5}} = \lim_{n \to +\infty} \frac{9n^{0.8}}{n^{0.5}} + \lim_{n \to +\infty} \frac{2n^{0.3}}{n^{0.5}} + \lim_{n \to +\infty} \frac{14logn}{n^{0.5}}$$

Now, evaluating each term separately:

$$\lim_{n \to +\infty} \frac{14logn}{n^{0.5}} = 0 \text{ ,}$$

Since : $\sqrt{n}$ grows faster than $logn$. To "formally" prove this, L'Hopital's rule can be used: Notice that $14log_2(n) = \frac{14ln(n)}{ln(2)}$, so that $f'(n) = \frac{1}{nln(2)}$. Also, $g'(n) = \frac{1}{2\sqrt{n}}$, such that:

$$\lim_{n \to +\infty} \frac{14logn}{n^{0.5}} = \lim_{n \to +\infty} \frac{2\sqrt{n}}{nln(2)}$$. linear functions grow faster than square roots, so this evaluates to zero. L'Hopital's rule can be applied again to show this:

Derivative of top expression = $\frac{1}{\sqrt{n}}$

Derivative of bottom expression: ln(2)

thus,

$$\lim_{n\to+\infty} \frac{2\sqrt{n}}{n ln(2)} = \lim_{n\to+\infty} \frac{\frac{1}{\sqrt{n}}}{\ln(2)} = \lim_{n\to+\infty} \frac{1}{\sqrt{n}\ln(2)} = 0$$

Next term:

$$\lim_{n\to+\infty} \frac{2n^{0.3}}{n^{0.5}} = \lim_{n\to+\infty} \frac{2}{n^{0.2}} = 0$$

$$\lim_{n\to+\infty} \frac{9n^{0.8}}{n^{0.5}} = \lim_{n\to+\infty} 9n^{0.3} = \infty$$

thus,

$$\lim_{n\to+\infty} \frac{9n^{0.8} + 2n^{0.3} + 14logn}{n^{0.5}} = \infty$$

f ∈ Θ(g)	No
f ∈ O(g),	No
f ∈ o(g),	No
f∈ Ω(g)	Yes
f ∈ ω(g)	Yes

now, the other way around

$$\lim_{n\to+\infty} \frac{n^{0.5}}{9n^{0.8} + 2n^{0.3} + 14logn} = 0$$ as shown above, the term with the greatest order of growth is $9n^{0.8}$, which evaluates the limit to zero as this term is in the denominator.

g∈ Θ(f)	No
g ∈ O(f),	Yes
g ∈ o(f),	Yes
g∈ Ω(f)	No
g ∈ ω(f)	No

(c) (2 points) f(n) = n 2/ log n and g(n) = n log n,

$$\lim_{n\to+\infty} \frac{\frac{n^2}{\log n}}{n \log n} = \lim_{n\to+\infty} \frac{n}{(\log n)^2}$$

again, we apply L'Hopital's rule to compute this limit

derivative of bottom expression = $\dfrac{2\ln n}{n \ln 2}$

$$\lim_{n\to+\infty} \frac{n}{(\log n)^2} = \lim_{n\to+\infty} \frac{n \ln 2}{2 \ln n}$$

now we use L'Hopital's rule to prove that linear growth is greater than logarithmic as n approaches infinity:

derivative of bottom expression $= \dfrac{2}{n}$

derivative of top expression = ln 2

$$\lim_{n\to+\infty} \frac{n \ln 2}{2 \ln n} = \lim_{n\to+\infty} \frac{n \ln 2}{2} = \infty$$

f ∈ Θ(g)	No
f ∈ O(g),	No
f ∈ o(g),	No
f∈ Ω(g)	Yes
f ∈ ω(g)	Yes

Now, the other way around:

$\lim_{n\to+\infty} \dfrac{n(\log n)^2}{n^2} = 0$**,** above, it is already shown that n^2 grows at a greater rate than $n(\log n)^2$ as $n \to \infty$.

g∈ Θ(f)	No
g ∈ O(f),	Yes
g ∈ o(f),	Yes
g∈ Ω(f)	No
g ∈ ω(f)	No

(d) (2 points) f(n) = (log(3n))3 and g(n) = 9 log n.

$$\lim_{n\to+\infty} \frac{(\frac{\ln(3n)}{\ln 2})^3}{9\frac{\ln(n)}{\ln(2)}}$$

again, we apply L'Hopital's rule:

Derivative of top expression:

$$f'(x) = \frac{3(\frac{\ln(3n)}{\ln 2})^2}{n}$$

Derivative of bottom expression:

$$g'(x) = \frac{9}{n \ln(2)}$$

Dividing these terms, we get:

$$\lim_{n\to+\infty} \frac{(\frac{\ln(3n)}{\ln 2})^3}{9\frac{\ln(n)}{\ln(2)}} = \lim_{n\to+\infty} \frac{3n\ln(2)(\frac{\ln(3n)}{\ln(2)})^2}{9n} = \lim_{n\to+\infty} \frac{3\ln(2)(\frac{\ln(3n)}{\ln(2)})^2}{9} = \infty$$

From this, we can also derive that $(\log(3N))^3$ grows at a greater rate than $9\log n$ as $n \to \infty$

f ∈ Θ(g)	No
f ∈ O(g),	No
f ∈ o(g),	No
f∈ Ω(g)	Yes
f ∈ ω(g)	Yes

Now, the other way around:

$\lim_{n\to+\infty} \frac{9\frac{\ln(n)}{\ln(2)}}{(\frac{\ln(3n)}{\ln 2})^3} = 0$, the derivations above showed that $(\log(3n))^3$ grows at a greater rate than $9\log n$ as $n \to \infty$, so the expression evaluates to zero.

g∈ Θ(f)	No
g ∈ O(f),	Yes
g ∈ o(f),	Yes
g∈ Ω(f)	No
g ∈ ω(f)	No

Problem 3.2

b) Loop invariant: at any time, the sub-array to the left of the current element is sorted. Consider the following trace of an execution of Selection Sort on the array {8,6,7,0,4,9,2,1,5,3}

*0	6	7	8	4	9	2	1	5	3
0	*1	7	8	4	9	2	6	5	3
0	1	*2	8	4	9	7	6	5	3
0	1	2	*3	4	9	7	6	5	8
0	1	2	3	*4	9	7	6	5	8
0	1	2	3	4	*5	7	6	9	8
0	1	2	3	4	5	*6	7	9	8
0	1	2	3	4	5	6	*7	9	8
0	1	2	3	4	5	6	7	*8	9

Where the * asterisk denotes the beginning of the leftmost sub-array. As It can be seen, the left-most sub-array is always sorted in each iteration of the outermost for loop of Selection Sort. Some special considerations are that:

- At iteration zero, the leftmost sub-array is of size 1, so it is trivially already sorted
- During any iteration, the leftmost sub-array is sorted by swapping positions with the smallest element in the right sub-array. This means that for the final iteration, since all of the left sub-array is sorted, then the final element of the complete array is also sorted, so the array is sorted.

c) For full code, see: testSelectionSort.cpp

The case with the least swaps is an array that is already sorted in increasing order: This is because no swaps need to be done: the following c++ code shows how to make an n-sized array with the least swaps for Selection Sort. (Note that the functions are void since they are snippets of the source code)

```
void best_Case(int n){
        int best_case[n];
        for (int i =0; i < n; i++)
                best_case[i] = i;
}
```

The case with the most swaps is an array with the greatest element in the first position, followed by the rest of the elements in increasing order. This gives a total of n-1 swaps for any input size n. So for an input of size 10, the worst case is {10, 1, 2, 3, 4, 5, 6, 7, 8, 9}. This is the worst case because the greatest number will always be the comparison index for swapping, and thus will always swap until the algorithm terminates.

```
void worst_Case(int n){
        int worst_case[n];
        worst_case[0] = n;
        for (int i = 1; i < n; i++)
                worst_case[i] = i;
}
```

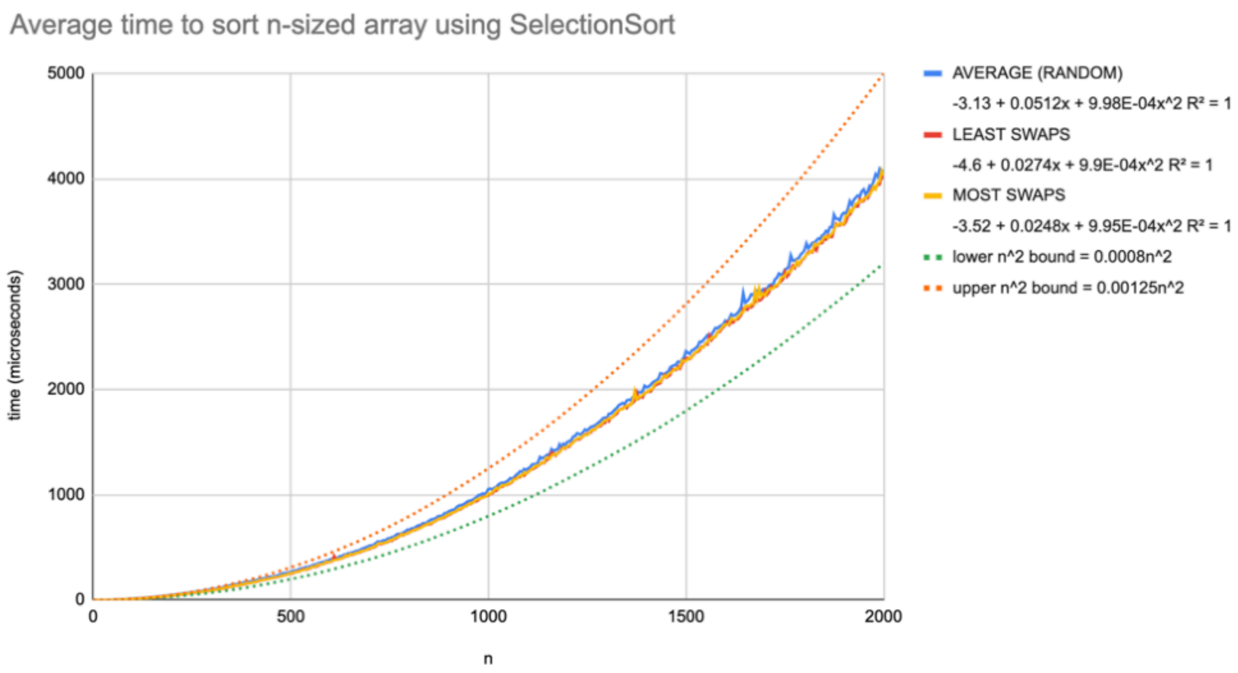
The “average case” is made using a random number generator to make the numbers.

```
# include <ctime>
void average_Case(int n){
        srand(time(NULL));
        int average_case[n];
        for(int i =0; i < n; i++)
                average_case[i] = rand();
}
```

***d)** The following results were obtained using the code from testSelectionSort.cpp, running the following parameters:*

- Maximum number of elements (n) = 2000
- Increasing intervals of n from 0 to maximum = 5
- Number of repetitions of the experiment to calculate the mean: 100

The mean results (called final_x.txt) were then passed to google sheets and graphed.



e) from the visual perspective it is easy to see that cases A, B, and the average case are all are all elements of O(n^2). Furthermore, all three of the cases seem to have a very similar time complexity.

To be more concise, let cases A, B, and the average case be defined as a function $f(n)$. In more mathematical terms:

$$0 \leq 0.0008n^2 \leq f(n) \leq 0.00125n^2, \forall n > n_0$$

Therefore, we can also conclude:

$f(n) = \Theta(n^2)$