Assignment 2

Diana Lin 260625972

Question One

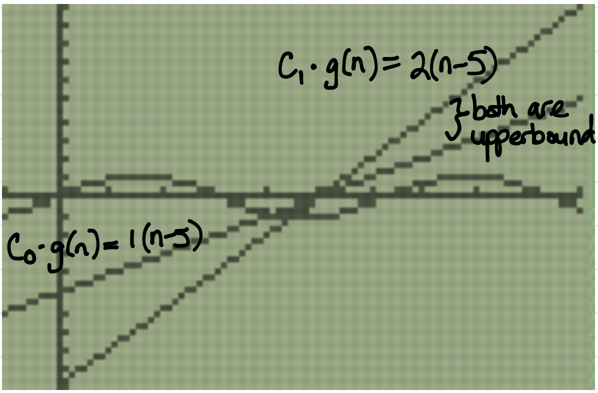
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \longrightarrow O(g(n)) \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty \to \Omega(g(n))$$

let $f(n) = \sin(n)$
$g(n) = n-5 \qquad c_0 = 1 \qquad c_1 = 2$

$$\left.\begin{aligned} &\lim_{n\to\infty} \frac{\sin(n)}{n-5} \Rightarrow \lim_{n\to\infty} \frac{\pm 1}{\infty} = 0 \\ &\lim_{n\to\infty} \frac{\sin(n)}{2n-10} \Rightarrow \lim_{n\to\infty} \frac{\pm 1}{\infty} = 0 \end{aligned}\right\} \begin{aligned} &g(n) = n-5 \\ &f(n) \in O(g(n)) \end{aligned}$$

$$\lim_{n\to\infty} \frac{\sin(n)}{n-5} \neq 0 < L < \infty \to f(n) \notin \Theta(g(n))$$
$$\neq \pm\infty \to f(n) \notin \Omega(g(n))$$
$$= 0 \to f(n) \in O(g(n))$$

$$\boxed{\begin{aligned} &\therefore f(n) \in O(g(n)) \\ &\therefore f(n) \notin \Theta(g(n)) \end{aligned}}$$



Question Two

a) i) $99n \rightarrow 99(2n)$

$99(2n) = 198n$

$\frac{198n}{99n} = 2$

$\boxed{2x}$

ii) $99n \rightarrow 99(n+1)$

$99(n+1) = 99n + 99$

$\frac{99n+99}{99n} = \frac{99n}{99n} + \frac{99}{99n}$

$\hookrightarrow 1 + \frac{1}{n} = \frac{n}{n} + \frac{1}{n}$

$\hookrightarrow \boxed{\frac{n+1}{n} x}$

b) i) $n^2 \rightarrow (2n)^2$

$(2n)^2 = 4n^2$

$\frac{4n^2}{n^2} = \boxed{4x}$

ii) $n^2 \rightarrow (n+1)^2$

$(n+1)^2 = n^2 + 2n + 1$

$\hookrightarrow \boxed{\frac{n^2+2n+1}{n^2} x}$

c) i) $n^4 \rightarrow (2n)^4$

$(2n)^4 = 16n^4$

$\frac{16n^4}{n^4} = \boxed{16x}$

ii) $n^4 \rightarrow (n+1)^4$

$(n+1)^4 = n^4 + 4n^3 + 6n^2 + 4n + 1$

$\boxed{\hookrightarrow \frac{n^4+4n^3+6n^2+4n+1}{n^4} x}$

d) i) $n2^n \rightarrow (2n)2^{(2n)}$

$(2n)2^{(2n)} = (2n)(2^n)^2$

$\frac{2\cancel{n}(2^n)^{\cancel{2}}}{\cancel{n}\cancel{2^n}} = 2(2^n)$

$\hookrightarrow 2^1 \cdot 2^n$

$\hookrightarrow \boxed{2^{n+1} x}$

ii) $n2^n \rightarrow (n+1)2^{(n+1)}$

$(n+1)2^{(n+1)} = (n+1)(2)(2^n)$

$\frac{(n+1)(2)(\cancel{2^n})}{n\cancel{2^n}} = \frac{2n+2}{n} = \boxed{\frac{2(n+1)}{n} x}$

e) i) $3^n \rightarrow 3^{(2n)}$

$3^{(2n)} \rightarrow (3^n)^2$

$\frac{(3^n)^{\cancel{2}}}{\cancel{3^n}} = \boxed{3^n x}$

ii) $3^n \rightarrow 3^{(n+1)}$

$3^{(n+1)} \rightarrow 3^n \cdot 3^1$

$\frac{3(\cancel{3^n})}{\cancel{3^n}} = \boxed{3x}$

ascending order of growth rate
↳ differentiate each function

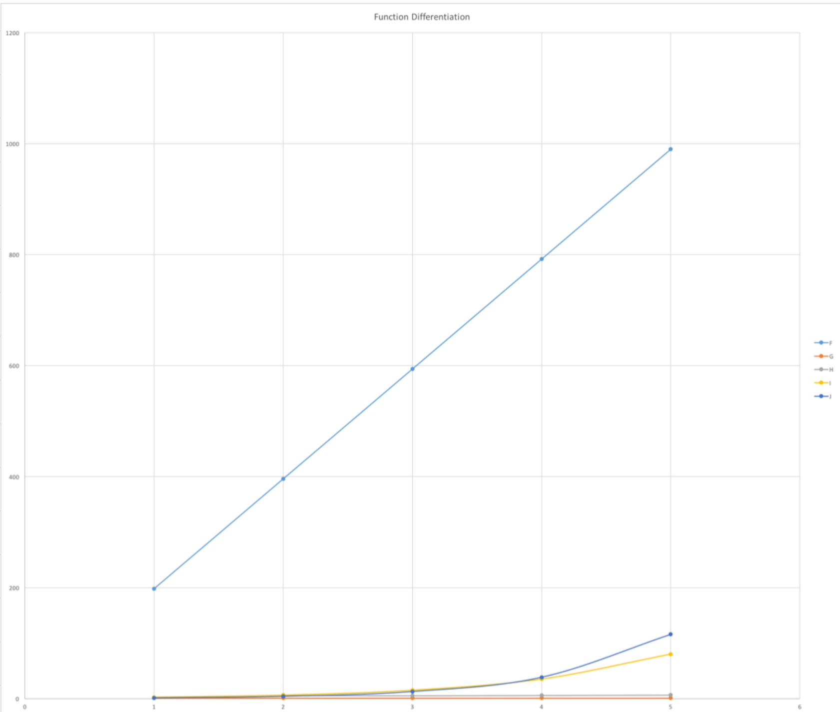
f) $f_1(n) = 99n^2$
$f_1'(n) = 2(99)n$
$f_1'(n) = 198n$

h) $f_3(n) = n^2 \log(\log(n))$
$$f_3'(n) = n\left(\frac{2\log(n)\log\left(\frac{\log(n)}{\log(10)}\right)+1}{\log(10)\log(n)}\right)$$

g) $f_2(n) = 2^{\lg n}$
$f_2(n) = \cancel{2^{\log_2}}n$
$f_2(n) = n$
$f_2'(n) = 1$

i) $f_4(n) = n2^n$
$f_4'(n) = 2^n(n\log(2)+1)$

j) $f_5(n) = 3^n$
$f_5'(n) = 3^n \log(3)$



$\boxed{G, H, I, J, F}$

Question Three

Assume $f(n)$ is $O(g(n))$. Refer to slide 74 (L9)

a) $\log_2 f(n)$ is $O(\log_2 g(n))$

$0 \leq f(n) \leq c \cdot g(n)$

$\log_2 f(n) \leq \log_2 c \cdot g(n) \rightarrow$ logs are directly proportional

$\hookrightarrow \log_2 c + \log g(n) \rightarrow$ law of logs

need a case where $\rightarrow \log_2 f(n) \leq d \log_2 g(n)$

$$\hookrightarrow \frac{\log_2 c + \log_2 g(n)}{\log_2 g(n)} \leq \frac{d \log_2 g(n)}{\log_2 g(n)}$$

$\frac{\log_2 c}{\log_2 g(n)} + 1 \leq d \rightarrow$ if $n \rightarrow \infty$, $\log_2 g(n) \rightarrow 0$

so: $2\left(\frac{1}{n} + 1\right) \in O\left(1 + \frac{1}{n}\right)$

counterexample:

$$\log_2 2 + \log_2(1 + \tfrac{1}{n}) \notin O(\log_2(1 + \tfrac{1}{n}))$$

$$\boxed{\text{FALSE}}$$

b) $2^{f(n)}$ is $O(2^{g(n)})$

c) $f(n)^2$ is $O(g(n)^2)$

b) $2^{f(n)}$ is $O(2^{g(n)})$

$$0 \leq f(n) \leq c\,g(n)$$

$$2^{f(n)} \leq 2^{c\,g(n)}$$

$$2^{f(n)} \leq \left(2^{g(n)}\right)^{c}$$

$$\log 2^{f(n)} \leq c \log 2^{g(n)}$$

need: $\log 2^{f(n)} \leq \log_c C + \log 2^{g(n)}$

$$\frac{\log_c C + \log 2^{g(n)}}{\log 2^{g(n)}} \leq d \, \frac{\log 2^{g(n)}}{\log 2^{g(n)}}$$

$$\frac{\log_c C}{\log 2^{g(n)}} + 1 \leq d$$

$\boxed{\text{FALSE}}$

$$2^{2n} \notin O(2^{n})$$

c) $f(n)^2$ is $O(g(n)^2)$

$$0 \leq f(n) \leq c \cdot g(n)$$

c^2 just a factor like c

$$0^2 \leq f(n)^2 \leq (c \cdot g(n))^2$$

$f(n)^2 \in O(g(n))^2$

$$0^2 \leq f(n)^2 \leq c^2 \cdot g(n)^2$$

$\boxed{\text{TRUE}}$

Question Four

a)

Algorithm algo1(n)

c $i \leftarrow 1$

lin **while** $i < n$ **do**

. $\quad i \leftarrow i + 100$

$\boxed{O(n)}$

b)

Algorithm algo2(n)

c $x \leftarrow 0$

quad **for** $i \leftarrow 1$ **to** n **do**

lin **for** $j \leftarrow 1$ **to** i **do**

. $\quad x \leftarrow x + 1$

$\boxed{O(n^2)}$

c)

Algorithm algo3(n)

c $i \leftarrow n$

log **while** $(i > 1)$ **do**

. $\quad i \leftarrow i/2$

$\boxed{O(\log(n))}$

d)

Algorithm algo4(n)

c $k \leftarrow 1$

c **for** $i \leftarrow 1$ **to** 1000

c **for** $j \leftarrow 1$ to i

. $\quad k \leftarrow (k + i - j) * (2 + i + j)$

$\boxed{O(1)}$

Question Five

Prove $\sum_{i=1}^{n} i^k$ is $\Theta(n^{k+1})$ for any positive integer k.

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ $\quad$ $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ $\quad$ $\sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$

$\sum_{i=1}^{n} i^k \rightarrow \Theta(n^{k+1})$ $\quad$ use this one as example

$$D(n^3) = n^3 - (n-1)^3$$
$$n^3 - (n^3 - 3n^2 + 3n - 1)$$
$$\cancel{n^3} - \cancel{n^3} + 3n^2 - 3n + 1 \rightarrow \frac{3n^2}{3} - \frac{3n}{3} + \frac{1}{3}$$
$$= n^2 \boxed{-n} + 1/3$$

$$D(n^2) = n^2 - (n-1)^2$$
$$n^2 - (n^2 - 2n + 1)$$
$$\cancel{n^2} - \cancel{n^2} + 2n - 1$$
$$2n - 1 \rightarrow \frac{2n}{2} - \frac{1}{2} = \boxed{n} - 1/2$$

$$D\left(\frac{n^3}{3} + \frac{n^2}{2}\right) = n^2 + 1/3 - 1/2$$
$$= n^2 - 1/6$$
$$D\left(\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}\right) = n^2$$

$$\sum_{i=1}^{n} i^2 = \frac{n(2n+1)(n+1)}{6}$$

$$\frac{2n^3 + 3n^2 + n}{6} = \frac{n(2n^2 + 3n + 1)}{6} = \frac{n(2n+1)(n+1)}{6}$$

Similar proof can be done with k.

$$D\left(\frac{n^{k+1}}{x(k+1)} + \frac{x(\ldots)}{x(\ldots)} + \frac{n}{x}\right) = n^k \quad k \in \oplus \text{integer}$$

$$\therefore \lim_{n \to \infty} \frac{f(n)}{g(n)} \neq \infty \text{ or } 0$$

Question Six

$f(n) \notin O(g(n))$ and $g(n) \notin O(f(n))$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$ and $\lim_{n\to\infty} \frac{g(n)}{f(n)} \neq 0$

$$\lim_{n\to\infty} \frac{n^2-1}{n^2+2n+1} \xrightarrow[\text{Rule}]{\text{l'Hopitals}} \frac{2n}{2n+2} \xrightarrow{\text{l'H}} \frac{2}{2} = 1$$

$$\lim_{n\to\infty} \frac{n^2+2n+1}{n^2-1} \xrightarrow{\text{L'H}} \frac{2n+2}{2n} \xrightarrow{\text{l'H}} \frac{2}{2} = 1$$

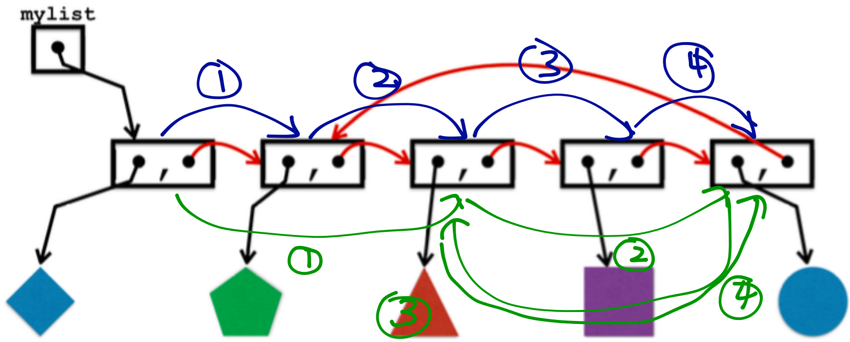
$$\boxed{\therefore\ f(n) = n^2-1 \quad g(n) = n^2+2n+1}$$

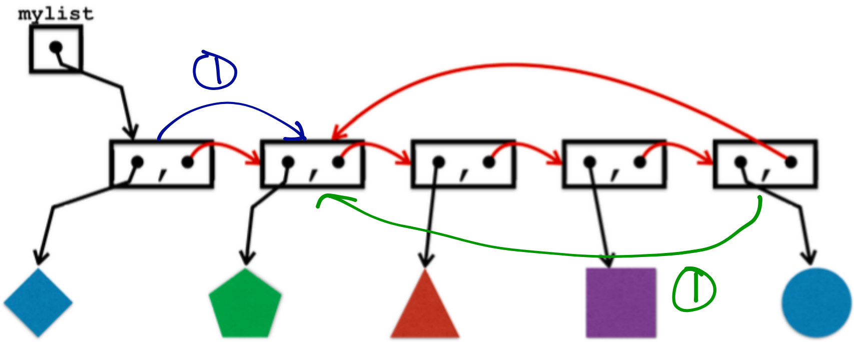
Question Seven

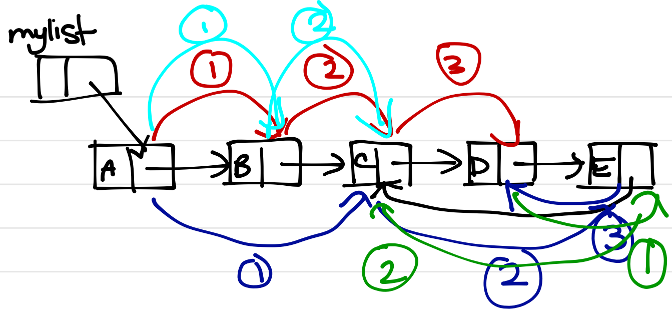
a) Step 1: Tortoise = ⬠
Hare = △
Step 2: Tortoise = △
Hare = ○
Step 3: Tortoise = □
Hare = △
Step 4: Tortoise = ○ } tortoise == hare on
Hare = ○ } the 4th step

b) Step 1: Tortoise = ⬠ } tortoise == hare on
Hare = ⬠ } the 1st step

c)



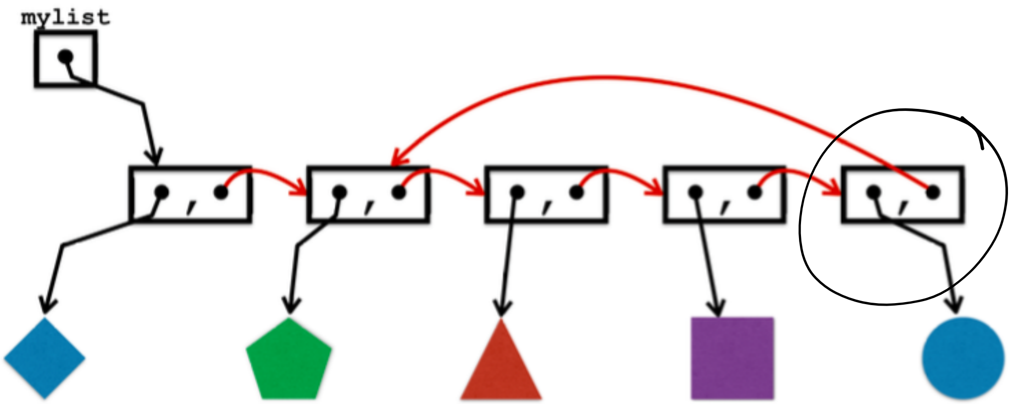




Step	Tortoise	Hare
1	B	C
2	C	E
3	D	D
Reset	A	D
1	B	E
2	C	C

x++ and y=y+2 until x==y. → reset x to 0, leave y in same position

x++ and y++ until x==y → loop entered



Explanation: Keeping the hare at its first cycle spot and moving the tortoise back to the beginning of the list sets the two apart at a certain distance.
By making the two move at equal pace they will either meet once again at the beginning of the loop, or will never meet again.

The initial set up ensures that the hare ends up somewhere within the loop equal to the distance between the hare and the tortoise. This is a critical set up. if there is one at double the pace of the tortoise. Then when the tortoise is reset, and the hare's pace is reduced The two are now running at the same speed. If they start at different places of the same route heading in the same direction, one will never catch up to the other since there is no "acceleration". However, if the route loops itself, then eventually the two will meet again, even if travelling at the same speed.

d) public static void main (String[] args) {

```
tortoise = tortoise.next;
hare = hare.next.next;
while (tortoise != hare) {
    tortoise = tortoise.next;
    hare = hare.next.next;
}

tortoise = mylist;
int counter = 0;

while (tortoise != hare) {
    tortoise = tortoise.next;
    counter++;
    hare = hare.next.next;
}
```

/*wherever tortoise and hare are, that is wher
the list enters the loop*/

```
int x = counter - 1; //counter is where
                     //loop starts
// x is length

SLinkedList NR = new SLinkedList();
pointer1 = NR;
pointer2 = mylist;
```

```
for (int i=0; i<x; i++) {
    pointer1 = pointer1.next;
    pointer 2 = pointer.next;
    pointer 2 = pointer1;
}

if (x != mylist.size()) {
    SLinkedList R = new LinkedList();
    pointer 3 = pointer 1.next;
    pointer4 = R;

    for(int i=0; i<(mylist.size()-x); i++) {
        pointer3 = pointer3.next;
        pointer 4 = pointer 4.next;
        pointer4 = pointer3;
    }

    pointer2.next = null;
    pointer4. next = null;

} else {
    SLinkedList R = new SlinkedList();
}

}

//Sorry have not mastered the skill of writing
//pseudocode
```

e) Runtime should be linear → no nested for loops → no division → $O(n)$
1st while loop runs for at least mylist.size() times → for loops go up to maximum of mylist.size()
↳ $S(n) = n$ — n → number of cells, S → space
↓
directly proportional

f) identify for each x the smallest n and k such that $f^{(n)}(x) = f^{(n+k)}(x)$

$$f^{(0)}(s) = s \qquad f^{(n)}(s) = f(f^{(n-1)}(s)) \qquad n>0$$

$$\left.\begin{aligned}
n=1\quad & f^{(1)}(s) = f(f^{(0)}(s)) \\
& f^{(1)}(s) = f(s) \\
n=2\quad & f^{(2)}(s) = f(f^{(1)}(s)) \\
& f^{(2)}(s) = f(f(s)) \\
n=3\quad & f^{(3)}(s) = f(f^{(2)}(s)) \\
& f^{(3)}(s) = f(f(f(s)))
\end{aligned}\right\} \text{composite } f(s)$$

$$f^{(n)}(x) = f^{(n+k)}(x)$$

	$n=0$	$n=1$	$n=2$	$n=3$
	step 1	step 2	step 3	step 4
A	s	$f(s)$	$f(f(s))$	$f(f(f(s)))$
B	s	$f(f(s))$	$f(f(f(f(s))))$	$f(f(f(f(f(f(s))))))$
	$n=0$	$n=2$	$n=4$	$n=6$

$n=4$ step 5 | $n=5$ step 6 | $n=6$ step 7 | $n=7$ step 8

A $f(f(f(f(f(s)))))$ ....

B ....

$n=8$ $n=10$ $n=12$ $n=14$

↳ finite set should eventually loop.

$f^{(n)}(x) = f^{(0)}(x)$;
$f^{(n+1)}(x) = f^{(0)}(x)$;
$f^{(n)}(x) = f^{(n-1)}(x)$;
while ($f^{(n)}(x)$!= $f^{(n-1)}(x)$) {
  $f^{(n)}(x) = f^{(n-1)}(x)$;
  $f^{(n+1)}(x) = f^{(n-1)}(x)$;
}

int i = 0;
$f^{(n)}(x) = f^{(0)}(x)$;
while ($f^{(n)}(x)$!= $f^{(n-1)}(x)$) {
  $f^{(n)}(x) = f^{(n-1)}(x)$;
  $f^{(n-1)}(x) = f^{(n-2)}(x)$;
  i++;
}
int j = 1;
$f^{(n+1)}(x) = f^{(n)}(x)$;
j++;
}

$\boxed{\rightarrow n = i \quad k = j}$

g) $O(n)$ and $O(k)$ → linear/directly proportional
↳ $S(n)$ and $S(k)$

h) - runtime to make our division algorithm more efficient
- assignment and this probem are equally hard
- being able to find a loop in a linked list ⇒ finding where the decimal repeated itself in Rep array
- breaking the linked list into NR and R can be used to break decimal part of assignment 1 into NonRep and Rep