Lab 02

Problem 1

a. The General procedure of designing new algorithms

1. Clarify the meaning of the topic, list the input, output, and constraints of the topic
2. Optimize the time and space complexity of the algorithm as much as possible
3. Writing pseudo-code or code to implement the algorithm

b. The Algorithm analysis framework

1. Measuring an input's size
2. Measuring running time
3. Orders of growth (of the algorithm's efficiency function)
4. Worst-case, best-case and average-case efficiency

c. Asymptotic notations

- $g(n)$: growth of an algorithm's basic operation count
- $O(g(n))$: class of functions f(n) that grow no faster than g(n)
- $\Omega(g(n))$: class of functions f(n) that grow at least as fast as g(n)
- $\Theta(g(n))$: class of functions f(n) that grow at same rate as g(n)

Problem 2

$t(n) = 12n^2 + 17/3n + 7/5 \in O(n^2)$
because for all $n \geq 1, t(n) < cn^2, c \geq 14$
$log(3n + 4/n) \in O(n)$
because for all all $n \geq n_0, t(n) < cn^2$

Problem 3

```
for i := 1 to n do
    if key = item[i]
        then return i
end
exit
```

Complexity

The algorithm will check each element, the average time complexity is $O(n)$, it doesn't introduce new storage, the space complexity is $O(1)$

	Average	Worst-case	Best-case
Time complexity	$O(n)$	$O(n)$	$O(1)$
Space complexity	$O(1)$	$O(1)$	$O(1)$

Problem 4

```
upperbound := ⌊sqrt(n)⌋
for i := 2 to upperbound do
    if n mod i = 0
        then return true
end
return false
exit
```

The algorithm will check $[2, \lfloor\sqrt{n}\rfloor]$, $\lfloor\sqrt{n}\rfloor - 1$ elements in total, until it find one that is divisible into it, the average time complexity is $O(\sqrt{n})$, it doesn't introduce new storage, the space complexity is $O(1)$

	Average	Worst-case	Best-case
Time complexity	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(1)$
Space complexity	$O(1)$	$O(1)$	$O(1)$