CHAPTER 1 – PROBLEMS

Problem 1. For each of the following algorithms, indicate (i) a natural size metric for its inputs, (ii) its basic operation, and (iii) whether the basic operation count can be different for inputs of the same size:

a. computing the sum of n numbers
   (i): n
   (ii): addition of two numbers
   (iii):it is the same for inputs of the same size

b. computing $n!$
   (i): n
   (ii): multiplication of two numbers
   (iii):it is the same for inputs of the same size

c. finding the largest element in a list of n numbers
   (i): n
   (ii): compare two numbers
   (iii):it is the same for inputs of the same size

d. Euclid's algorithm to find the GCD of two integers.
   (i): the magnitude of the input integers
   (ii): take the remainder of one number divided by the other.
   (iii):it is not the same for inputs of the same size. It depends on the size of the input integers and their divisibility. In the worst case, the number of basic operations is proportional to the number of digits in the smaller input integer.

Problem 2. Calculate the number of all assignment and comparison operations of the following algorithms, then show their order of growth in term of O-notation:

```
a. for (i = 0; i < n; i++) //n+1 assignments, n+1 comparisons
       for (j = 0; j < n; j++) //n(n+1) assignments, n(n+1) comparisons
         b[i][j] += c; //n² assignments
```

Assignment operations: $n + 1 + n(n + 1) + n^2 = 2n^2 + 2n + 1$
Comparison operations: $n + 1 + n(n + 1) = n^2 + 2n + 1$
All operations: $3n^2 + 4n + 2$
Order of growth: $O(n^2)$

```
b. for (i = 0; i < n; i++)//n+1 assignments, n+1 comparisons
       if (a[i] == k) //worst case: n comparisons
           return 1;
   return 0;
```

Assignment operations: $n + 1$
Comparison operations: $2n + 1$

All operations: $3n + 2$
Order of growth: $O(n)$

```
c. for (i = 0; i < n; i++)//n+1 assignments, n+1 comparisons
       for (j = i+1; j < n; j++)//n(n+1)/2 assignments, comparisons
           b[i][j] -= c; //n(n-1)/2 assignments
```

Assignment operations: $n + 1 + \frac{n(n+1)}{2} + \frac{n(n-1)}{2} = n^2 + n + 1$

Comparison operations: $n + 1 + \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{3n}{2} + 1$

All operations: $\frac{3n^2}{2} + \frac{5n}{2} + 2$
Order of growth: $O(n^2)$

Problem 3. For each of the following pairs of functions, indicate whether the first function of each of the following pairs has a lower, same, or higher order of growth (to within a constant multiple) than the second function.

a. $n(n + 1)$ and $2000n^2$ → same order of growth
b. $100n^2$ and $0.01n^3$ → lower
c. $\log_2 n$ and $\ln n$ → same Since changing a logarithm's base can be done by the formula $\log_a n = \log_a b \log_b n$, all logarithmic functions have the same order of growth to within a constant multiple
d. $\log^2{}_2 n$ and $\log_2 n^2$ → higher
e. $2^{n-1}$and 2^n → same since $2^{n-1} = \frac{1}{2}2^n$
f. $(n - 1)!$ and $n!$ → lower

Problem 4. List the following functions according to their order of growth from the lowest to the highest:

$$(n - 2)!, 5\lg(n + 100)^{10}, 2^{2n}, 0.001n^4 + 3n^3 + 1, \ln^2 n, \sqrt[3]{n}, 3^n$$

$$5\lg(n + 100)^{10}, \ln^2 n, \sqrt[3]{n}, 0.001n^4 + 3n^3 + 1, 3^n, 2^{2n}, (n - 2)!,$$

Problem 5. Express the function $\frac{n^3}{1000} - 100n^2 - 100n + 3$ in terms of O-notation.

$$f(x) = \frac{n^3}{1000} - 100n^2 - 100n + 3 \le n^3 + 3, \forall n > 0$$

$$f(x) \le n^3 + n^3 = 2n^3 \ , \forall n \ge 2.$$

We select $c = 2, n_0 = 2$. By definition of O-notation, $f(x) \in O(n^3)$

Problem 6. Explain why the statement, "The running time of algorithm A is at least $O(n^2)$," is meaningless.
The O–notation provides an upper bound. "At least" implies a lower bound.

Problem 7. Show that :

a. $f(x) = 4x^2 - 5x + 3 \in O(x^2)$
   $\rightarrow f(x) \le 4x^2 + 3 \le 4x^2 + x^2 = 5x^2, \forall x \ge 2$. Select $c = 5, n_0 = 2$

b. $f(x) = (x + 5)\, log_2(3x^2 + 7) \in O(x\, log_2\, x)$
   $f(x) \le (x + 5)\log_2(4x^2), \forall x \ge 3$
   $f(x) \le (x + 5)\log_2 x^3 = 3(x + 5)\log_2 x, \forall x \ge 4$

$f(x) \le 3(x + 5x)\log_2 x = 18x \log_2 x, \forall x \ge 4$
Select $c = 18, x_0 = 4$

c. $f(x) = (x^2 + 5\,log_2\,x)/(2x + 1) \in O(x)$

Since $\log_2 x < x, \forall x > 0$, we have:

$$5\log_2 x < 5x < 5x^2, \forall x > 1$$

Since $2x + 1 > 2x$, we have:

$$\frac{1}{2x+1} < \frac{1}{2x}, \forall x > 0$$

Therefore,

$$f(x) \le \frac{x^2 + 5x^2}{2x}, \forall x > 1$$

$$f(x) \le 3x, \forall x > 1$$

Select $c = 3, x_0 = 2$

Problem 8. Are the following functions $O(x)$?

a. $f(x) = 10$

Yes. $f(x) \le x, \forall x \ge 10. Select\ c = 1, n_0 = 10$

b. $f(x) = 3x + 7$

Yes, $f(x) \le 3x + x = 4x, \forall x \ge 7. Select\ c = 4, n_0 = 7$

c. $f(x) = 2x^2 + 2$

No,

Assume that $f(x)$ is expressed as $O(x)$, we have:

$$\exists c, x_0 > 0, f(x) \le c.x, \forall x \ge x_0$$

Select $x = cx_0 \ge x_0$ then

$f(cx_0) = 2(cx_0)^2 + 2 \le c^2 x_0$

$2c^2(x_0^2 - 2x_0 + 1) + 3c^2 x_0 - 2c^2 + 2 \le 0$

$2c^2(x_0 - 1)^2 + 3c^2(x_0 - 1) + c^2 + 2 \le 0$ (false)

Problem 9. Describe the running time of the following function using O-notation:

$$S = 1 + \frac{1}{2} + \frac{1}{6} + \cdots + \frac{1}{n!}$$

```
float CalculateS(int n)
{
	float s = 1;
	int fact = 1;
	for(int i=1; i<=n; i++)
	{
		fact = fact * i;		//main operation: n+1
		s = s + 1/(float)fact;
	}
	return s;
}
```

The total number of main operations: multiplications, additions, assignments: 3n+3

→ O(n)

Problem 10. Find $g(n)$ of the following $f(n)$ so that $f(n) \in O(g(n))$.

a. $f(n) = (2 + n) * (3 + \log_2 n)$

b. $f(n) = 11 * \log_2 n + \frac{n}{2} - 3542$

c. $f(n) = n * (3 + n) - 7n$

d. $f(n) = \log_2(n^2) + n$

Problem 11. Determine $O(g(n))$ of the following functions:

a. $f(n) = 10 \rightarrow O(1)$

b. $f(n) = 5n + 3 \rightarrow O(n)$

c. $f(n) = 10n^2 - 3n + 20 \rightarrow O(n^2)$

d. $f(n) = \log n + 100 \rightarrow O(\log n)$

e. $f(n) = n \log n + \log n + 5 \rightarrow O(n \log n)$

Problem 12. Which one is correct? Explain your answer.

a. $2^{n+1} = O(2^n)$?

Yes. We can select c = 3, $n_0 = 1$

b. $2^{2n} = O(2^n)$?

No. Assume $f(x) = 2^{2n}$ is expressed as $O(2^n)$, then:

$\exists c, n_0 > 0, f(x) \leq c2^n, \forall n \geq n_0$

$f(x) = 2^n . 2^n \leq c2^n, \forall n \geq n_0$

$2^n \leq c, \forall n \geq n_0$. Select $n = \lfloor \log_2(c+1) \rfloor + n_0 \geq n_0$

$2^{\lfloor \log_2(c+1) \rfloor + n_0} = (c+1)2^{n_0} \leq c$ (false)

Problem 13. Write the algorithms to solve the following problems using C++ and recursion. Find the Big-O of your algorithms.

a. Find the maximum of an array of *n* integers.

```
int FindMaximum (int a[], int n)
{
      if(n == 1)
            return a[0];
      int max = FindMaximum (a, n - 1);
      if (max < a[n - 1])
            max = a[n -1];
      return max;
}
```

$\rightarrow O(n)$

b. Find the minimum of an array of *n* integers.

```
int FindMinimum (int a[], int n)
{
	if (n == 1)
		return  a[0];
	int min = FindMinimum (a, n - 1);
	if (min > a[n - 1])
		min = a[n - 1];
	return min;
}
```

$\rightarrow O(n)$

c. Calculate the factorial of an integer *n.*

```
int Factorial (int n)
{
	if (n == 1)
		return 1;
	return n * Factorial(n-1);
}
```

$\rightarrow O(n)$

d. Calculate the sum of *n* integers.

```
Int Sum(int a[], int n)
{
	if (n == 1)
		return a[0];
	return a[n-1] + Sum(a, n - 1);
}
```

$\rightarrow O(n)$

e. Calculate the sum of 1 to *n*.

```
int Sum(int n)
{
	if (n == 1)
		return 1;
	return n + Sum(n - 1);
}
```

$\rightarrow O(n)$

f. Check if an array is symmetric or not.
(Example of a symmetric array: $< 12, 4, 3, 4, 12 >, < 5, 19, 19, 5 >$)

```
bool CheckSymmetry(int a[], int s, int n)
{
	if (s >= n)
		return true;
	if (a[s] == a[n - 1])
		return CheckSymmetry(a, s + 1, n - 1);
	return false;
}
```

$\rightarrow O(n)$