Practice Exercise 7: O-notation

Exercises a-b. Suppose that each of the following expressions represents the number of logical operations in an algorithm as a function of *n*, the size of the list being manipulated. For each expression, determine the ***dominant term*** and then classify the algorithm in ***simplified O-notation***.

(a). $1000n^3 + n^2\log_{10}n + 2n^3 \log_2 n + 20200^7$

1a. What is the dominant term?
1b. What is the O-notation?

(b). $8n^2 + 7^{89}\log_8 n + 2^n + 56n^4$

2a. What is the dominant term?
2b. What is the O-notation?

Exercises c-k. Find the running time (O-notation) in terms of ***n*** for each section. Do *not* try to understand what the functions do, but simply look at their execution times.

Exercise (c)

```
int function computation(int result, int n)
{
      for (int i = 3; i <= n; ++i)
            result += i * n;
      return result;
}
```

Exercise (d)

```
void function tables(int k)
{
      for (int i = 0; i < k; ++i)
            for (int j = 0; j <= k; ++j)
                  a[i] += a[j] + i + j;
}
```

Exercise (e)

```
long factorial (int n)
{
      if (n < = 1)
            return 1;
      else
            return n * factorial (n - 1);
}
```

Exercise (f)

```
bool DoublyList::search(const int& searchData) const
{
	bool found = false;
	Node *current = first;

	while (current != NULL && !found)
	{
		if (current->getData() == searchData)
			found = true;
		else
			current = current->getNextLink();
	}
	return found;
}
```

Exercise (g)

```
void modifyArray(int a[], int size, int item)
{
	int max = a[0];

	for (int i = 1; i < size / 2; ++i)
	{
		if (max < a[i])
			max = a[i];
	}
	for (int j = 1; j <= size; ++j)
	{
		++max;
		cout << max;
	}
}
```

Exercise (h)

```
void doSomething(int n)
{
	for (int k = 1; k <= n / 2; ++k)
	{
		cout << (k * k) << endl;
		for (int j = 1; j <= n; ++j)
			cout << (n * n * n * n) << " ";
	}
}
```

Exercise (i)

```
bool myFunction (int k)
{
      int x = k + 2;
      while (x > 1) x /= 2;
      return (k > x);
}
```

Exercise (j)

```
bool anotherFunction(const vector<int>& v1)
{
      if (v1.size() == 0)return false;
      else
      {
            int size = static_cast<int>(v1.size());
            for (unsigned int i = 0; i < size - 1; ++ i)
                  for (unsigned int j = size/2; j < size - 1; ++j)
                        if (v1[i] != v1[j+1]) return true;
      }
      return false;
}
```

Exercise (k)

```
void thisFunction(int n)
{
      for (int k = 0; k < n; ++k)
      {
            j = n;
            while (j > 0)
            {
                  cout << j << " ";
                  j /= 2;
            }
      }
}
```