1 Algorithm Analysis

Some Useful Formulas:

$$1 + 2 + 3 + 4... + N = \frac{N(N+1)}{2}$$

$$1 + 2 + 4 + 8 + ... + N = 2N - 1$$

$$2^0 + 2^1 + 2^2 + 2^3 + ... + 2^N = 2^{N+1} - 1$$

$$a + ar + ar^2 + ... + ar^{n-1} = \Sigma_{i=1}^{n} ar^{i-1} = a\frac{1-r^n}{1-r}$$

Give the worst-case order of growth of the runtime in $\Theta(\cdot)$ notation as a function of N. Your answer should be simple with no unnecessary leading constants or summations.

(a) Θ(_____)

```
static int recursion(int N) {
    int x = 0;
    if (N < 1000) {
        for (int i = 0; i <= N * N * N; i++) {
            x++;
        }
        return x;
    }
    for (int i = 1; i <= N / 2; i++) {
        x++;
    }
    return x + 2 * recursion(N / 2);
}
```

2 Array Resizing

We implement `Queue` ADT using **`ArrayQueue3`** that **`add`** and **`remove`** methods work as expected. However, when we try to **`add`** a new element into the full `Queue`, we will make a new array with size of `size+1` and copy every element over.

(a) What are the valid runtime bound of **`add`** in each case? (choose **ALL** that apply)

Best case:

☐ $\Omega(1)$	☐ $\Theta(1)$	☐ $O(1)$
☐ $\Omega(logN)$	☐ $\Theta(logN)$	☐ $O(logN)$
☐ $\Omega(N)$	☐ $\Theta(N)$	☐ $O(N)$

Worst Case:

☐ $\Omega(logN)$	☐ $\Theta(logN)$	☐ $O(logN)$
☐ $\Omega(N)$	☐ $\Theta(N)$	☐ $O(N)$
☐ $\Omega(N^2)$	☐ $\Theta(N^2)$	☐ $O(N^2)$

Overall:

☐ $\Omega(1)$	☐ $\Theta(1)$	☐ $O(1)$
☐ $\Omega(logN)$	☐ $\Theta(logN)$	☐ $O(logN)$
☐ $\Omega(N)$	☐ $\Theta(N)$	☐ $O(N)$
☐ $\Omega(N^2)$	☐ $\Theta(N^2)$	☐ $O(N^2)$

(b) If **`ArrayQueue3`** starts empty with an array of size 4, give a sequence of 6 operations that would perform the worst runtime (e.g., `add/remove/add/remove/add/remove` is a sequence of operations):

(c) What could be improved on **`ArrayQueue3`** to improve overall performances? (one sentence should be enough.)

Will you want to pick up your worksheet later? Circle one: Yes / No
How confident do you feel with the material this week? Circle one: 1 / 2 / 3 / 4 / 5

3 Algorithm Analysis (Optional)

(a) Θ(_____)

```
static void addItUp(int N) {
    int x = 0;
    for (int i = 1; i <= N; i *= 2) {
        for (int j = 0; j < i; j++) {
            x++;
        }
        if (i % 2 == 1) {
            for (int j = 0; j < N; j++) {
                x++;
            }
        }
    }
    System.out.println(x);
}
```