COMS 311: Homework 1
Due: Sept 9, 11:59pm
Total Points: 100

Late submission policy. If you submit by Sept 10, 11:59PM, there will be 20% penalty. That is, if your score is x points, then your final score for this homework after the penalty will be $0.8 \times x$. Submission after Sept 10, 11:59PM will not be graded without explicit permission from the instructors.

Submission format. Your submission should be in pdf format. Name your submission file: `<Your-net-id>-311-hw1.pdf`. For instance, if your netid is `asterix`, then your submission file will be named `asterix-311-hw1.pdf`.

If you are planning to write your solution and scan+upload, please make sure the submission is readable (sometimes scanned versions are very difficult to read - poor scan quality to blame).

If you plan to snap pictures of your work and upload, please make sure the generated pdf is readable - in most cases, such submissions are difficult to read and more importantly, you may end up submitting parts of your work.

If you would like to type your work, then one of the options you can explore is latex (Overleaf). You can use the source file of this homework to get started.

Learning outcomes.

1. Determine whether or not a function is Big-O of another function
2. Analyze asymptotic worst-case time complexity of algorithms

---

1. Prove or disprove the following statements. Provide a proof for your answers. (40 Points)
   (a) $6n^2 - 41n + 2 \in O(n^2)$.
   (b) $\forall a \geq 1 : 2^n \in O(2^{n-a})$
   (c) $\forall a > 1 : O(\texttt{log}_2 n) \in O(\texttt{log}_a n))$
   (d) $\forall a > 1 : a^{a^{n+1}} \in O(a^{a^n})$.
   (e) If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then $f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$.

2. Derive the runtime of the following as a function of n and determine its Big-O upper bound. You must show the derivation of the end result. Simply stating the final answer without any derivation steps will result in 0 points. (60 Points)

   (a)
```
for (i = 1; i < n-8; i++) {
     for (j = i; j < i+8; j = j++) {
          <some-constant number of atomic/elementary operations>
}}
```
   (b)
```
for i in the range [1, n] {
    for j in the range [i, n] {
         for k in the range [1, j-i] {
          <some-constant number of atomic/elementary operations>
}}}
```

(c)
```
x = pow(2, n);
i = 1;
while i <= x {
  for j in range [1, i] {
        <some-constant number of atomic/elementary operations>
  }
  i = i * 2
}
```
Assume that `pow(2, n)` (i.e., 2^n) is computed *magically* in constant time.

3. **Extra Credit** Derive the runtime of the following in terms of n and determine its Big-O upper bound. You must show the derivation of the end result. Simply stating the final answer without any derivation steps will result in 0 points. (10pts)

```
function myf(integer a, integer n) {
    integer a1;
    while n >= 0 {
       if n==0 then return 1;
       a1 = myf(a, n/2);
       if n is even then  {
           return a1 * a1;
       }
       else {
            return a * a1 * a1;
       }
       n = n/2;
    }
}
```

Assume that $n/2 = 0$, when $n < 2$.