Dr. Huo - SWE300-HW8 - Due: **Name:** Andrew Januszko

This homework is **individual**. Total points is 12.

1. (2 points) Explain the refactoring "move an expression inline" with an example.

Moving an expression inline is when you replace an intermediate variable that was assigned the result of an expression with the expression itself.
Instead of 'int i = a + b; int j = i + c;' you type 'int j = a + b + c;'

2. (3 points) There is another refactoring technique called "introduce an intermediate variable" which assigns an expression to an intermediate variable whose name summarizes the purpose of the expression. Give an example and explain when it does not contradict the "move an expression inline".

An example would be if you are using an equation with an expression in the numerator and the denominator.
so if the equation was (8x + 9y / 22x - 6y):

```
double numerator =(8 * x) + (9 * y);
double denominator = (22 * x) - (6 * y);
double quotient = numerator / denominator;
```

This better summarizes the purpose expression compared to:

```
double quotient = ((8 * x) + (9 * y)) / ((22 * x) - (6 * y));
```

3. (3 points) What's the benefit putting the most likely case as the true branch of an if statement? Give an example with code (other than the one in the slides).

The benefit of putting the most likely case as the true branch of an if-statement is a reduced runtime for most cases.

An example of this is:

```
public static Node append(Node lst1, Node2) {
   if(lst1 != null) {
     return append(lst1.next, lst2);
   } else {
     lst1 = lst2
     return lst1;
   }
}
```

Since lst1 is most likely not null, we put that as the true conditional.

4. (3 points) Unroll the loop:

```
int arr[10];
for (int i=0; i<10; i++) {
  arr[i] = i+2;
}
```

```
int arr[10];
arr[0] = 2;
arr[1] = 3;
arr[2] = 4;
arr[3] = 5;
arr[4] = 6;
arr[5] = 7;
arr[6] = 8;
arr[7] = 9;
arr[8] = 10;
arr[9] = 11;
```