Natalie Craun
February 1st, 2017
Problem 3 (Runtime Analysis)

Part (a)

```
for(int i=n-1; i >=0; i--){
  for(int k=0; k < i*n; k++){
    // do something that takes O(1) time
  }
}
```

$$\sum_{i=0}^{n}\sum_{k=0}^{i*n} O(1)$$

If $i = n-1$, the inner loop will take $n(n-1) + n(n-2)... O(n^2)$ time to execute.

$\sum_{i=0}^{n} O(n^2)$ The overall function will then take $n(n^2) + (n-1)n^2 + (n-2)n^2...\Theta(n^3)$ time to execute.

Part (b)

```
for(int i=1; i < n; i++){
  for(int k=1; k <= n; k++){
    if( A[k] == i){
      for(int m=1; m <= n; m=m+m){
        // do something that takes O(1) time
        // Assume the contents of the A[] array are not changed
      }
    }
  }
}
```

The if statement will only execute once for every iteration of the outer loop, it makes the inner loop essentially run in O(1) time because it will only perform the if statement operations once for every k through n loops. The if statement will only be satisfied n times at most, making the entirety of the inner loop and if statement contents $\Theta(n)$.

So we have $\sum_{i=1}^{n-1} \Theta(n)$ which resolves to $\Theta(n^2)$

Part (c)

```
void f3(int* A, int n)
{
  if(n <= 1) return;
  else {
    f3(A, n-2);
    // do something that takes O(1) time
    f3(A, n-2);
  }
}
```

Say n = 10. 1st: n=10, 2nd: n=8, 3rd: n=6, 4th: n=4, 5th: n=2, 6th n=0, STOP

The first recursive call has executed $\frac{n}{2}$ times.

If this recursive function only had ONE recursive call, the runtime would be $\Theta(n)$ because the function would call itself $\frac{n}{2}$ times. BUT because there are two recursive calls, each call results in 2 more calls. This means as n increases, the number of recursive calls grows exponentially: 2^n

Therefore, the runtime of this function is $\Theta(2^n)$

Part (d)

```
int *a = new int [10]; // new is O(1)
int size = 10;
for (int i = 0; i < n; i ++)
  {
    if (i == size)
      {
        int newsize = 3*size/2;
        int *b = new int [newsize];   // new is O(1)
        for (int j = 0; j < size; j ++) b[j] = a[j];
        delete [] a;   // delete is O(1)
        a = b;
        size = newsize;
```

```
        }
    a[i] = i*i;
  }
```

The if statement will only execute once, (when i == 10). It will not continue to be called because the updated size only exists within the if statement. The operations inside the if statement take $O(1)$ time because the for loop will only run 10 times.

Therefore, we have

$$\sum_{i=0}^{n} O(1)$$

which makes the overall runtime $\Theta(n)$.