11/25 # Dynamic Programming

1) compute n^{th} fibonacci number

Fib(n) = sum of two preceding numbers

```
int fib(int n) {
    if (n==0) return 0;
    if (n==1) return 1;
    int a=0; b=0;
    for(int i=2; i< n; i++){
        int c = a+b;
        a = b;
        b = c;
    }
    return a + b;
}
```

2) Child can climb stairs/steps in 1, 2, or 3 steps at a time. Count possible ways to climb the stairs.

Given: n stairs & 1, 2, 3 steps Need: # of ways to climb.

Time complexity: $O(3^n)$

```
int countWays(int n){
    if(n==0) return 1;
    if(n<0) return 0;
    return countWays(n-1) + countWays(n-2) + countWays(n-3);
}
```

DP Technique:

```
int countWays(int n) { int[] cache = new int[n+1];
    if (n<0) return 0;
    if (n==0) return 1;
    if (cache[n]>-1) { return cache[n]; }
        cache[n] = countWays(n-1,cache) + countWay(n-2, cache)
                 + countWays(n-3, cache);
    return cache[n];
}
```

Runtime: $O(n)$