Andrew DiBella
Hw0

1) The Runtime Stack and the Runtime Heap are both used together to allocate space for data in memory. The stack is used for storing local variables, primitive data types, and pointers to function calls or objects. The stack is used as a call stack and uses a Last in First out structure to organize the data. The heap is used for storing abstract data types(ADTs) like objects. These data types are called through the stack by using reference pointers that point to the location of the object in memory.
2) Using proof by induction, prove that the sum of integers 1, 2, ..., n is equal to n(n+1)/2:
Base Case:
For some integer n, $S(n) = n(n+1)/2$ if $S(2) = 2\,(2+1)/2 = 3 = 1+2$
Induction Step:
Assume for some integer k that $S(k) = k(k+1)/2$

$S(k+1) = 1+2+3+...+k + k+1$

$$= k(k+1) / 2 \quad + \quad k+1$$
$$= k(k+1)/2 \quad + \quad 2(k+1)/2$$
$$= k(k+1)+2(k+1)/2$$
$$= (k+1)(k+2)/2$$
$$= (k+1)\,((k+1)+1)/2$$

Therefore since the function is correct for the base case 2 and the function S works for k and k+1 we can conclude that the sum of integers 1,2,...,n is equal to the function n(n+1)/2.
3) The Four fundamental rules of recursion:
   a) You must always have a base case that will eventually stop the recursive call
   b) The recursive call must always make progress towards the base case
   c) Design rule: Assume all recursive calls work
   d) Never duplicate work by solving the same instance of a problem

A recursive function from above formula for sum of integers to n:

```
int sumN(int n){
  if (n<=1) return 1;
  return n + sumN(n-1);
}
```

a) This recursive call uses a base case that returns 1 if n<=1

b) The function starts at n and decrements down after every call making progress toward the basecase
c) Assuming the input of integer n is a positive integer then the program will work recursively, otherwise the output will be 1
d) The recursive function does not duplicate work

4) Will the following Java function terminate for all inputs? Prove your answer. You should assume that the int type does not wrap around. That is, it will not overflow; the int type can represent any integer, positive or negative.

```
void printToN(int n) {
    for (int i = 0; i != n + 1; i++) {
        System.out.println(i);
    }
}
```

The function will not be able to terminate for all inputs due to the possibility of n being a negative number less than -1. If n were to be less than -1 the the terminating int i will never not equal n+1. For example, if n = -2 then n+1= -1 and since i begins at 0 and increments the loop will never stop and will never satisfy the condition not equal n + 1.

5) What are the two things that define an Abstract Data Type (ADT)?

Two things that define an ADT are the data and the operations.