Name: Willem Scott
ID: 108759616

CSCI 3104, Algorithms **Due January 22, 2021**
Problem Set 1 (50 points) **Spring 2021, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.
Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- The easiest way to access Gradescope is through our Canvas page. There is a Gradescope button in the left menu.
- Gradescope will only accept **.pdf** files.
- It is vital that you match each problem part with your work. Skip to 1:40 to just see the matching info.

Name: Willem Scott
ID: 108759616

CSCI 3104, Algorithms **Due January 22, 2021**
Problem Set 1 (50 points) **Spring 2021, CU-Boulder**

1. (a) Identify and describe the components of a loop invariant proof.
   (b) Identify and describe the components of a mathematical induction proof.

   Solution:

   a. A loop invariant proof is comprised of a loop invariant, which should hold true at the beginning of each iteration of our loop, as well as an initialization step, maintenance step, and termination step, which show that the invariant holds for our first iteration, each following iteration given the previous, and the final iteration respectively.

   b. A mathematical induction proof is comprised of a base case, $f(0)$ (or several $f(k)$ in a certain range), and an inductive step that operates off the understanding that if we can show $f(k-1)$ then we can show $f(k)$. From this we can show that $f(0)$ implies $f(1)$, and $f(1)$ implies $f(2)$ and so on.

2. Identify the loop invariant for the following algorithms.

(a)
```
function Sum(A)
    answer=0;
    n=length(A);
    for i=1 to n
        answer += A[i]
    end
    return answer
end
```

(b)
```
function Reverse(A)
    n=length(A)
    i=ceiling(n/2)
    j=ceiling(n/2) + (n+1) mod 2
    while i>0 and j<=n
        tmp=A[i]
        A[i]=A[j]
        A[j]=tmp
        i=i-1
        j=j+1
    end
end
```

(c) Assume that **A** is sorted such that the largest value is at **A**[n]. Assume **A** contains the value `target`.

```
function Search(A,target) //returns the index of the value target
    left=1
    right=length(A)
    while left <=right
        m=floor((left+right)/2)
        if A[m] < target
            left=m+1
        else if A[m]>target
            right=m-1
        else
            return m
        end
    end
end
```

Solution:

a. For each iteration i's start, answer will be equal to the sum of numbers of the elements of array $A[:i]$

b. For each iteration k's start, A[i-k:i+k] will be equal to the reverse of its original values

c. For each iteration, the target will be in the subarray A[left:right]

Name: Willem Scott
ID: 108759616

3. Prove the correctness of the following algorithm. (*Hint: You need to prove the correctness of the inner loop before you can prove the correctness of the outer loop.*)

```
function Sort(A)
    n=length(A)
    for i=1 to n
        for j=2 to n
            if A[j]<A[j-1]
                swap(A[j],A[j-1]) //a function that swaps the elements in the array
            end
        end
    end
end
```

Solution:

Inner Loop: Loop Invariant: At the beginning of every iteration step j, $A[j-1]$ is equal to the largest number $A[:j-1]$

Initialization: At the beginning of the first loop when j is 2, $A[j-1]$ is the first item, which is the largest and only item in the subarray $A[:j-1]$

Maintenance: Given that the loop invariant is true at the beginning of each iteration j, we verify whether $A[j] < A[j-1]$ and if it is then the two elements will be swapped. Swapping the elements will result in $A[j] >= A[j-1]$, or if the items were not swapped, this condition was already satisfied. Thus, when iteration $j+1$ starts, $A[j]$ will be larger than any item in $A[:j-1]$. If the items were not swapped, $A[j-1]$ is the largest item in $A[:j-1]$, which is implied by the loop invariant.

Termination: The loop will terminate at $j = n$. The loop invariant gives that at the end of the last step $j = n$, that $A[n]$ will be the maximum of $A[0:n]$. If the last k items of the list were already sorted, then by the loop invariant it is guaranteed that the last k+1 items will now be sorted.

Outer Loop:

Loop Invariant: At the beginning of every iteration step i, $A[n-i:n]$ is sorted.

Initialization: At the beginning of the first loop where $i = 1$, $A[n-1:n]$ contains no items and as such is sorted.

Maintenance: Given that the loop invariant is true at the beginning of each iteration j, we are guaranteed that a new value has found its spot in the last $i-1$ elements by the inner loop's termination, meaning now the last i elements are sorted.

Termination: The loop will terminate at $i = n$. The loop invariant gives that the last $i = n$ elements will be sorted, meaning $A[0:n]$ (being the entire array) will be sorted.

Name: Willem Scott
ID: 108759616

4. (a) Suppose you have a whole chocolate bar composed of $n \geq 1$ individual pieces. Prove that the minimum number of breaks to divide the chocolate bar into n pieces is $n - 1$.

   (b) Show that for fibonacci numbers $\sum_{i=1}^{n} f_i^2 = f_n f_{n+1}$
   Recall that the fibonacci numbers are defined as
   $f_0 = 0,\ f_1 = 1$
   $\forall n > 1,\ f_n = f_{n-1} + f_{n-2}$

   (c) For which nonnegative integers n is $3n + 2 \leq 2^n$? Prove your answer.

Solution:

a. Base Case: At $k = 1$ there have been no breaks and the bar is in one piece. $0 = 1 - 1$

Inductive Step: Each new break will increase the number of pieces by one, and will increase the number of breaks by 1. Given $n = k - 1$ and $n_{breaks} = n - 1$, one additional break yields one additional piece, and we are left with $n = k$ and $n_{breaks} = n - 1$

Therefore for any n pieces there can only have been $n - 1$ breaks.

b. Base Case: $k = 1$, $\sum_{i=1}^{1} f_1^2 = f_1^2 = f_1 f_2$. If $f_1^2 = 1$ and $f_2 = f_1 = 1$ then $f_1^2 = f_1 f_2 = 1 \times 1 = 1$

Inductive Step: Given $\sum_{i=1}^{k-1} f_i^2 = f_{k-1} f_k$ then $\sum_{i=1}^{k} f_i^2 = f_k^2 + \sum_{i=1}^{k-1} f_i^2 = f_k^2 + f_{k-1} f_k$. When f_k is factored out we are left with $f_k\,(f_k + f_{k-1})$, which is equal to $f_k f_{k+1}$

Therefore, for all $k \geq 1$, $\sum_{i=1}^{n} f_i^2 = f_n$

c. Let $f(n) = 3n + 2$ and $g(n) = 2^n$. The equation $f(n) \leq g(n)$ is not satisfied by any number less than 4, and is guaranteed to be satisfied by any number greater than 4. $f'(n) = 3$, $g'(n) = 2^n log(2)$ are true by definition.

Base Case: $3(4) + 2 \leq 2^4$, holds true when simplified to $14 \leq 16$

Inductive step: Given $f(n) \leq g(n)$, $f(n+1) = f(n) + 3$ and $g(n+1) = 2g(n)$ implies that $f(n+1) \leq g(n+1)$ since $g'(n)$ is also greater than $f'(n)$, meaning between n and n + 1, $g(n+1) - g(n)$ is greater than $f(n+1) - f(n)$ because $g'(n)$ is strictly increasing.

For all remaining numbers 0 through 3, it can be shown that none satisfy this condition.

$$f(0) = 2, g(0) = 1, 21$$

$$f(1) = 5, g(1) = 2, 52$$

$$f(2) = 8, g(2) = 4, 84$$

$$f(3) = 11, g(3) = 8, 118$$