Name: Jaryd Meek
ID:
Collaborators: Noah Nguyen

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: Jaryd Meek
ID:
Collaborators: Noah Nguyen
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)
   i. **Initialization -** Describes what is true prior to the first iteration of the loop.
   ii. **Maintenance -** Describes how if the loop invariant is true before the iteration of the loop, it is also true after an iteration of the loop.
   iii. **Termination -** When the loop hits it's termination point, the invariant gives us information to prove that the algorithm is correct.

(b)
   i. **Base Case -** In the base case you verify that the given statement or equation holds for the first item in the set.
   ii. **Induction Step -** In the induction step, you prove that since it worked for the n case, it also works for the $n+1$ case.
   iii. **Conclusion -** State that since the induction step proved true, the statement is true for all n.

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) At the start of iteration i, the variable `answer` will have the sum of the values in the subarry $A[1 : (i - 1)]$.

(b) At the start of each iteration, the subarray $A[i+1 : j-1]$ will contain the items from the original array, in reverse order.

(c) At the start of each iteration, the subarray $A[left : right]$ will contain the `target`.

Name: Jaryd Meek
ID:
Collaborators: Noah Nguyen

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

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 - Prior to iteration j, the value of $A[j-1]$ is the largest in the subarray $A[1:j-1]$

Initialization - The loop initializes at $j = 2$. This tells us that $A[2-1] = A[1]$ is the largest value in the subarray $A[1:2-1] = A[1:1]$. Since $A[1:1]$ is just a single value, with that value being $A[1]$, $A[1]$ must be the largest value.

Maintenence - At the start of the j^{th} iteration, the subarray $A[1:j-1]$ has the largest value at $A[j-1]$. We examine $A[j]$ to determine if it is less than $A[j-1]$. If it is, we swap the two. This maintains that the largest value will be at the position of $A[j-1]$, thus the loop invariant is maintained.

Termination - At the start of the last loop $j = n$, we assume that the loop invariant holds before the last iteration and that $A[n-1]$ is the largest value in the subarray $A[1:n-1]$. After using the maintenance step, we get that $A[j-1] = A[n]$ is the largest value in the array $A[1:n]$ and we terminate.

Outer Loop -
Loop Invariant - Prior to iteration i, the subarray $A[n-(i-1):n]$ is sorted.

Initialization - The loop initializes at $i = 1$. This tells us that $A[n-(i-1):n] = A[n-(1-1):n] = A[n:n]$ which is an array of one item and is therefore sorted.

Maintenence - At the start of the i^{th} iteration, $A[n-(i-1):n]$ is sorted. Thus when iteration i+1 starts, the inner for loop moves the greatest value in the subarray $A[1:n-i]$ to the rightmost position. This means that the subarray $A[n-i:n]$ will be sorted. Therefore the loop invariant is maintained.

Termination - At the start of the last loop, $i = n$. We assume that the loop invariant holds before the last iteration, meaning that $A[2:n]$ is sorted. Using the maintainence step, we get that $A[1:n]$ is sorted. i then incriments and we terminate with A sorted.

Name: $\boxed{\text{Jaryd Meek}}$
ID: $\square$
Collaborators: Noah Nguyen

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

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) **Proof by weak induction -**
Base Case -
If there is 1 piece of chocolate, there are 0 breaks since there is only a single square of chocolate. (like you just got it out of the wrapper)

$$\text{number of breaks} = n - 1 = 1 - 1 = 0$$
$$0 \stackrel{\checkmark}{=} 0$$

Inductive Step -
Inductive Hypothesis - Assume that every chocolate bar with n pieces has $n - 1$ breaks.
Let C be a chocolate bar with $n + 1$ pieces.
Let C' be a chocolate bar with one less piece.
C' has n pieces, By the inductive hypothesis, C' has $n - 1$ breaks.
Thus, C has $n + 1$ pieces and n breaks because it has one more break than C'

Conclusion -
Since n was arbitrary, we've shown that all chocolate bars with with n pieces have $n - 1$ breaks

(b) **Proof by weak induction -**
Base Case -

$$n = 2$$
$$\sum_{i=1}^{2} f_i^2 = f_1^2 + f_2^2 = 1^2 + 1^2 = 2$$
$$f_2 \cdot f_3 = 1 \cdot 2 = 2$$
$$2 \stackrel{\checkmark}{=} 2$$

Inductive Step -
Assume that $\sum_{i=1}^{n} f_i^2 = f_i \cdot f_{i+1}$ for some $n \geq 2$

$$\begin{aligned}
\sum_{i=1}^{n+1} f_i^2 &= \sum_{i=1}^{n} f_i^2 + f_{n+1}^2 && \text{(By properties of sums)} \\
&= f_n \cdot f_{n+1} + f_{n+1}^2 && \text{(By inductive hypothesis)} \\
&= f_{n+1}\,(f_n + f_{n+1}) && \text{(Factor out a } f_{n+1}\text{)} \\
&= f_{n+1} \cdot f_{n+2} && \text{(Simplify } f_n + f_{n+1} \text{ to } f_{n+2} \text{ using definiton of fibonacci numbers)}
\end{aligned}$$

Conclusion -
Therefore by weak induction, we've shown that $\forall n > 1, \sum_{i=1}^{n} f_i^2 = f_n \cdot f_{n+1}$

(c) **Proof by weak induction -**
Base Case -

$$n = 4$$
$$(3 \cdot 4) + 2 = 14$$
$$2^4 = 16$$
$$14 \overset{\checkmark}{\leq} 16$$

Inductive Step -
Assume that $3k + 2 \leq 2^k$ for all $k \geq 4$

Substitute k+1 in and simplify

$$3(k + 1) + 2 \leq 2^{k+1}$$
$$3k + 5 \leq 2^{k+1}$$

Solve the original equation for k+1

$$3k + 2 \leq 2^k \qquad \text{(start with the inductive hypothesis)}$$
$$2(3k + 2) \leq 2^k \cdot 2 \qquad \text{(multiply both sides by 2)}$$
$$6k + 4 \leq 2^{k+1} \qquad \text{(distribute the left, simplify the right)}$$
$$\text{Ok, since we have that } 6k + 4 \leq 2^{k+1} \text{ Ok, since we have that}$$
$$6k + 4 \leq 2^{k+1} \qquad \text{(distribute the left, simplify the right)}$$

Properties of inequalities

$$\text{since } 3k + 5 \leq 2^{k+1}$$
$$\text{and } 6k + 4 \leq 2^{k+1}$$
$$\text{and } 3k + 5 \leq 6k + 5 \quad \forall k \geq 4$$
$$\therefore 3k + 5 \leq 6k + 4 \leq 2^{k+1}$$

Conclusion - Since we have shown above that when you substitute $k+1$ into the original equation, you get that 2^{k+1} must be greater than $3k + 5$ and below we've shown that 2^{k+1} must also be greater than $6k + 4$, and $6k + 4$ must always be greater than $3k + 5$ (since we're working with all numbers greater than 4), we've shown that $\forall n \geq 4, 3n + 2 \leq 2^n$