CSE 311 Section 08

Induction, Recursively Defined Sets, and One-to-One and Onto

Administrivia

Announcements & Reminders

- Midterm
  - Please don’t talk about the midterm!! Not everyone has taken it yet ☺
- HW6 Regrade Requests
  - Regrade request window open as usual
  - If something was graded incorrectly, submit a regrade request
- HW7
  - Due **Friday** 11/22 @ 11:59pm (note the unusual day!)
  - Late due date Monday 11/25 @ 11:59 PM

Recursively Defined Sets

Recursive Definition of Sets

Define a set S as follows:

Basis Step:
Describe the basic starting elements in your set
ex: $0 \in S$

Recursive Step:
Describe how to derive new elements of the set from previous elements
ex: If $x \in S$ then $x + 2 \in S$.

Exclusion Rule: Every element of S is in S from the basis step (alone) or a finite number of recursive steps starting from a basis step.

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

b) Binary strings not containing 10.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

d) Binary strings containing at most two 0s and at most two 1s.

Work on this problem with the people around you.

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Generate accepted and rejected strings first!

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Generate accepted and rejected strings first!

Step 1: Write out basic cases and more intricate cases

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Generate accepted and rejected strings first!

Step 1: Write out basic cases and more intricate cases

Accepted Strings	Rejected Strings
ε	0
11	1
10101010	1010101
10101011	010101011

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Generate accepted and rejected strings first!

Step 1: Write out basic cases and more intricate cases

Step 2: Find a pattern!

Accepted Strings	Rejected Strings
ε	0
11	1
10101010	1010101
10101011	010101011

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Generate accepted and rejected strings first!

Step 1: Write out basic cases and more intricate cases

Accepted Strings	Rejected Strings
ε	0
11	1
10101010	1010101
10101011	010101011

Step 2: Find a pattern!

All even-length strings can be generated from **a series of substrings** of **length 2!**

All possible substrings of length 2 are:
10, 01, 11, 00

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Step 3: Write out Basis and Recursive step

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Step 3: Write out Basis and Recursive step

Basis: $\varepsilon \in S$

Recursive Step: If $x \in S$, then $x00, x01, x10, x11 \in S$

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Step 3: Write out Basis and Recursive step

Basis: $\varepsilon \in S$

Recursive Step: If $x \in S$, then $x00, x01, x10, x11 \in S$

Step 4: check that you cannot build the rejected strings and only build accepted strings with the recursive step

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

a) Binary strings of even length.

Step 3: Write out Basis and Recursive step

Basis: $\varepsilon \in S$

Recursive Step: If $x \in S$, then $x00, x01, x10, x11 \in S$

Step 4: check that you cannot build the rejected strings and only build accepted strings with the recursive step

Accepted Strings	Rejected Strings
ε	0
11	1
10101010	1010101
10101011	010101011

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

b) Binary strings not containing 10.

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

b) Binary strings not containing 10.

Step 1: Write out basic cases and more intricate cases

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

b) Binary strings not containing 10.

Step 1: Write out basic cases and more intricate cases

Accepted Strings	Rejected Strings
1	01**0**
0	**1**0
ε	10**0**
111	111**0**
00001	**1**00001

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

b) Binary strings not containing 10.

Step 1: Write out basic cases and more intricate cases

Step 2: Find a pattern!

Accepted Strings	Rejected Strings
1	010
0	10
ε	100
111	1110
00001	100001

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

b) Binary strings not containing 10.

Step 1: Write out basic cases and more intricate cases

Accepted Strings	Rejected Strings
1	01**0**
0	**1**0
ε	10**0**
111	111**0**
00001	**1**00001

Step 2: Find a pattern!

0's and 1's cannot be in the same string **unless** 0's **come first** and 1's **come second**

0's should be **built from the left** (0x)
1's should be **built from the right** (x1)

such strings that have 1's and 0's can only look like: 000…1111

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

b) Binary strings not containing 10.

Step 3: Write out Basis and Recursive step

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

b) Binary strings not containing 10.

Step 3: Write out Basis and Recursive step

If the string does not contain 10, then the first 1 in the string can only be followed by more 1s. Hence, it must be of the form 0^m1^n for some $m, n \in \mathbb{N}$.

Basis: $\varepsilon \in S$

Recursive Step: If $x \in S$, then $0x \in S$ and $x1 \in S$

Step 4: check that you cannot build the rejected strings and only build accepted strings with the recursive step :)

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

Step 1: Write out basic cases and more intricate cases

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

Step 1: Write out basic cases and more intricate cases

Accepted Strings	Rejected Strings
1	010
01	**0**
ε	100
111	1110
00001111	**00001**

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

Step 1: Write out basic cases and more intricate cases

Step 2: Find a pattern!

Accepted Strings	Rejected Strings
1	010
01	**0**
ε	100
111	1110
00001111	**00001**

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

Step 1: Write out basic cases and more intricate cases

Accepted Strings	Rejected Strings
1	010
01	**0**
ε	100
111	1110
00001111	**00001**

Step 2: Find a pattern!

From part (b) we know:
0's should be **built from the left** (0x)
1's should be **built from the right** (x1)

New restriction for adding a 0:
for every 0 we add, there must be **at least** an additional 1 accompanying it so we always have **# 1's ≥ # 0's**

So lets change: 0x to 0x1

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

Step 3: Write out Basis and Recursive step

Problem 1 – Recursively Defined Sets

For each of the following, write a recursive definition of the sets satisfying the following properties. Briefly justify that your solution is correct.

c) Binary strings not containing 10 as a substring and having at least as many 1s as 0s.

Step 3: Write out Basis and Recursive step

These must be of the form 0^m1^n for some $m, n \in \mathbb{N}$ with $m \leq n$. We can ensure that by pairing up the 0s with 1s as they are added:

Basis: $\varepsilon \in S$.

Recursive Step: If $x \in S$, then $0x1 \in S$ and $x1 \in S$.

Step 4: check that you cannot build the rejected strings and only build accepted strings with the recursive step :)

Structural Induction

Idea of Structural Induction

Every element is built up recursively…

So to show $P(s)$ for all $s \in S$…

Show $P(b)$ for all base case elements b.

Show for an arbitrary element not in the base case, if $P()$ holds for every named element in the recursive rule, then $P()$ holds for the new element (each recursive rule will be a case of this proof).

Structural Induction Template

Let $P(x)$ be “<predicate>”. We show $P(x)$ holds for all $x \in S$ by structural induction.

Base Case: Show $P(x)$
[Do that for every base cases x in S.]

Inductive Hypothesis: Suppose $P(x)$ for an arbitrary x
[Do that for every x listed as in S in the recursive rules.]

Inductive Step: Show $P()$ holds for y.
[You will need a separate case/step for every recursive rule.]

Therefore $P(x)$ holds for all $x \in S$ by the principle of induction.

Problem 2b – Structural Induction on Trees

Definition of Tree:
Basis Step: • is a Tree.
Recursive Step: If L is a Tree and R is a Tree then Tree(•, L, R) is a Tree

Definition of leaves():
leaves(•) = 1
leaves(Tree(•, L, R)) = leaves(L) + leaves(R)

Definition of size():
size(•) = 1
size(Tree(•, L, R)) =1 + size(L) + size(R)

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

Work on this problem with the people around you.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For $x \in S$, let P(x) be "".
We show P(x) holds for all $x \in S$ by structural induction on x.

Base Case: Show P(x) (for all x in the basis rules)

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules), i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(x) holds for all $x \in S$ by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(x) holds for all x ∈ S by structural induction on x.

Base Case: Show P(x) (for all x in the basis rules)

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules), i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(x) holds for all x ∈ S by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: Show P(x) (for all x in the basis rules)

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules),
i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b - Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules),
i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b - Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R, i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R, i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R, i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be “leaves(T) ≥ size(T)/2 + 1/2”.
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R, i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R, i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

Again, as long as you can get this far, you will get the majority of points on the problem! Go for this skeleton first, and then think about what you need to do to complete the proof.

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2
leaves(Tree(•, L, R)) =

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R) definition of leaves

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R) — definition of leaves

≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2) — by Inductive Hypothesis

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R) — definition of leaves
≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2) — by Inductive Hypothesis
= (1/2 + size(L)/2 + size(R)/2) + 1/2

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R) — definition of leaves
≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2) — by Inductive Hypothesis
= (1/2 + size(L)/2 + size(R)/2) + 1/2
= (1 + size(L) + size(R)) / 2 + 1/2

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R)	definition of leaves
≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2)	by Inductive Hypothesis
= (1/2 + size(L)/2 + size(R)/2) + 1/2	
= (1 + size(L) + size(R)) / 2 + 1/2	
= size(T)/2 + 1/2	definition of size

Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

For a tree T, let P(T) be "leaves(T) ≥ size(T)/2 + 1/2".
We show P(T) holds for all trees T by structural induction on T.

Base Case: P(•): By definition of leaves(•), leaves(•) = 1 and size(•) = 1.
So, leaves(•) = 1 ≥ 1/2 + 1/2 = size(•)/2 + 1/2, so P(•) holds.

Inductive Hypothesis: Suppose P(L) and P(R) hold for some arbitrary trees L and R,
i.e., leaves(L) ≥ size(L)/2 + 1/2, leaves(R) ≥ size(R)/2 + 1/2

Inductive Step: Goal: Show P(Tree(•, L, R)): leaves(Tree(•, L, R)) ≥ size(Tree(•, L, R))/2 + 1/2

leaves(Tree(•, L, R)) = leaves(L) + leaves(R)	definition of leaves
≥ (size(L)/2 + 1/2) + (size(R)/2 + 1/2)	by Inductive Hypothesis
= (1/2 + size(L)/2 + size(R)/2) + 1/2	
= (1 + size(L) + size(R)) / 2 + 1/2	
= size(T)/2 + 1/2	definition of size

So, P(Tree(•, L, R)) holds!
Conclusion: Therefore P(T) holds for all trees T by the principle of induction.

Problem 2a - Structural Induction on Strings

Definition of string:
Basis Step: "" is a string.
Recursive Step: If X is a string and c is a character then append(c, X) is a string.

Definition of **len**():
len("") = 0
len(append(c, X)) = 1 + len(X)

Definition of **double**():
double("") = ""
double(append(c, X)) = append(c, append(c, double(X)))

Prove that for any string X, len(double(X)) = 2len(X).

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For $x \in S$, let P(x) be “something”.
We show P(x) holds for all $x \in S$ by structural induction on x.

Base Case: Show P(x) (for all x in the basis rules)

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules),
i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(x) holds for all $x \in S$ by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We show P(x) holds for all x ∈ S by structural induction on x.

Base Case: Show P(x) (for all x in the basis rules)

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules),
i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: Show P(x) (for all x in the basis rules)

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules),
i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(x) (for all x in the recursive rules),
i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. (IH in terms of P(x))

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show that P(?) holds. (IS goal in terms of P(?))

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show P(append(c, X)) for any c: len(double(append(c, X))) = 2(len(append(c, X)))

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show P(append(c, X)) for any c: len(double(append(c, X))) = 2(len(append(c, X)))
len(double(append(c, X))) =

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show P(append(c, X)) for any c: len(double(append(c, X))) = 2(len(append(c, X)))

len(double(append(c, X))) = len(append(c, append(c, double(X)))) definition of double

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show P(append(c, X)) for any c: len(double(append(c, X))) = 2(len(append(c, X)))

len(double(append(c, X))) = len(append(c, append(c, double(X)))) definition of double
= 1 + len(append(c, double(X))) definition of len

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show P(append(c, X)) for any c: len(double(append(c, X))) = 2(len(append(c, X)))

len(double(append(c, X))) = len(append(c, append(c, double(X)))) definition of double
= 1 + len(append(c, double(X))) definition of len
= 1 + 1 + len(double(X)) definition of len

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show P(append(c, X)) for any c: len(double(append(c, X))) = 2(len(append(c, X)))

len(double(append(c, X))) = len(append(c, append(c, double(X)))) definition of double
= 1 + len(append(c, double(X))) definition of len
= 1 + 1 + len(double(X)) definition of len
= 2 + 2len(X) by I.H.

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show P(append(c, X)) for any c: len(double(append(c, X))) = 2(len(append(c, X)))

len(double(append(c, X))) = len(append(c, append(c, double(X))))	definition of double
= 1 + len(append(c, double(X)))	definition of len
= 1 + 1 + len(double(X))	definition of len
= 2 + 2len(X)	by I.H.
= 2(1 + len(X))	

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show P(append(c, X)) for any c: len(double(append(c, X))) = 2(len(append(c, X)))

len(double(append(c, X))) = len(append(c, append(c, double(X))))	definition of double
= 1 + len(append(c, double(X)))	definition of len
= 1 + 1 + len(double(X))	definition of len
= 2 + 2len(X)	by I.H.
= 2(1 + len(X))	
= 2(len(append(c, X)))	definition of len

Conclusion: Therefore P(X) holds for all strings X by structural induction.

Problem 2a - Structural Induction on Strings

Prove that for any string X,
len(double(X)) = 2len(X)

For a string X, let P(X) be “len(double(X)) = 2len(X)”.
We prove P(X) for all strings X by structural induction on X

Base Case: P(""): By definition, len(double("")) = len("") = 0 = 2 · 0 = 2len(""), so P("") holds

Inductive Hypothesis: Suppose P(X) holds for some arbitrary string X,
i.e. len(double(X)) = 2len(X)

Inductive Step: Goal: Show P(append(c, X)) for any c: len(double(append(c, X))) = 2(len(append(c, X)))

len(double(append(c, X))) = len(append(c, append(c, double(X))))	definition of double
= 1 + len(append(c, double(X)))	definition of len
= 1 + 1 + len(double(X))	definition of len
= 2 + 2len(X)	by I.H.
= 2(1 + len(X))	
= 2(len(append(c, X)))	definition of len

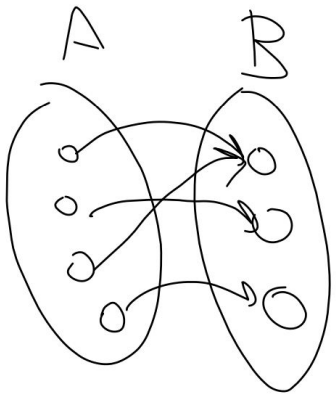
So, P(append(c, X)) holds!

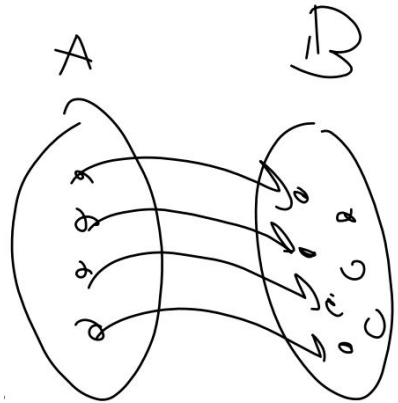
Conclusion: Therefore P(X) holds for all strings X by structural induction.

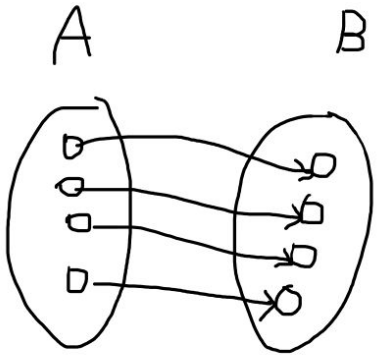
One-to-One and Onto

Review of function definitions

Match the definition to the description! (One to one, onto or bijective?)



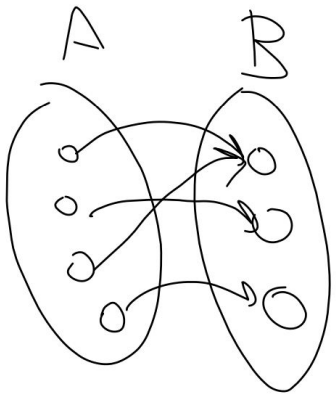


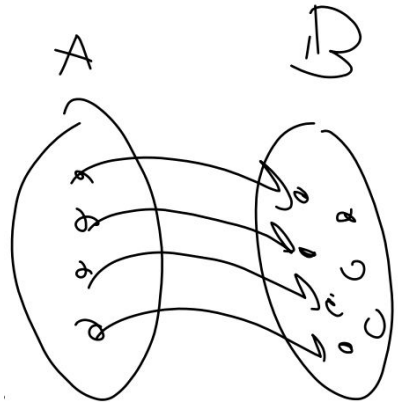


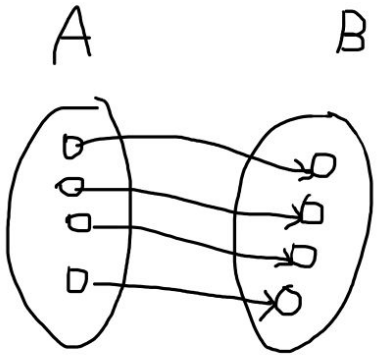
Review of function definitions

Match the definition to the description! (One to one, onto or bijective?)

Onto function
Every element of the codomain has <u>at least one element</u> in the domain mapping to it



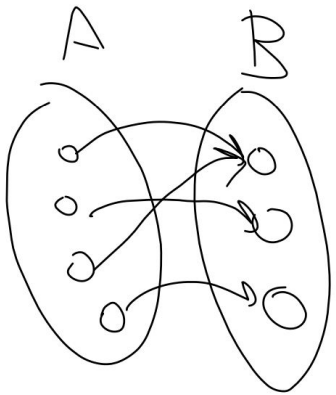




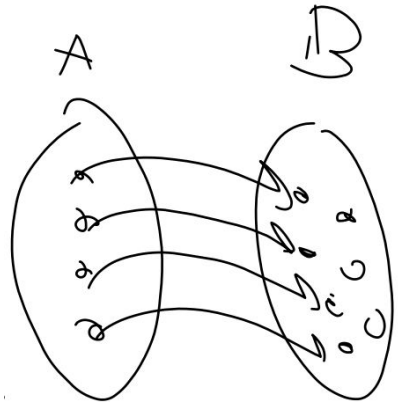
Review of function definitions

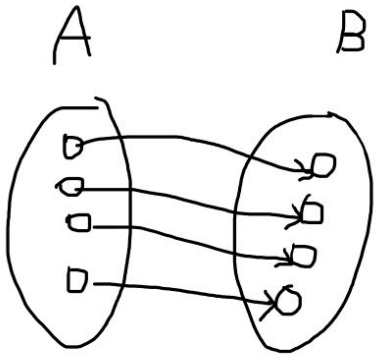
Match the definition to the description! (One to one, onto or bijective?)

Onto function
Every element of the codomain has at least one element in the domain mapping to it



One to one function
Every element of the codomain has at most one element in the domain mapping to it

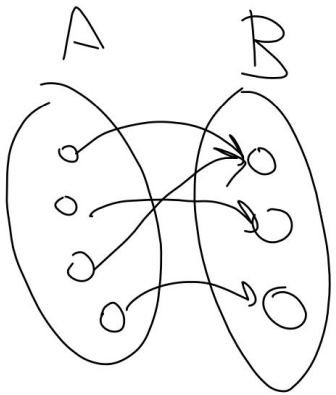




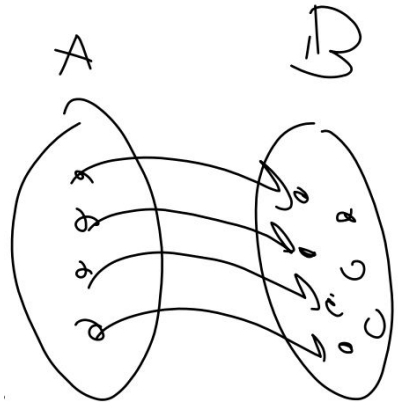
Review of function definitions

Match the definition to the description! (One to one, onto or bijective?)

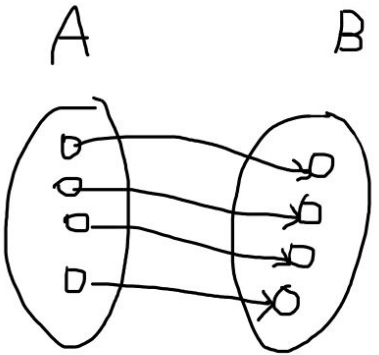
Onto function
Every element of the codomain has at least one element in the domain mapping to it



One to one function
Every element of the codomain has at most one element in the domain mapping to it



Bijection
Every element of the codomain has exactly one element in the domain mapping to it



Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

a) $f : \mathbb{N} \to \mathbb{N}, f(x) = x^2$

b) $f : \mathbb{R} \to \mathbb{R}, f(x) = x^2$

c) $f : \mathbb{R}^+ \to \mathbb{R}^+, f(x) = x^2$

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

a) $f : \mathbb{N} \to \mathbb{N}, f(x) = x^2$

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

a) $f : \mathbb{N} \to \mathbb{N}, f(x) = x^2$

For this domain and co-domain, the function is one-to-one, but not onto.

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

a) $f : \mathbb{N} \to \mathbb{N}, f(x) = x^2$

For this domain and co-domain, the function is one-to-one, but not onto.

It is one-to-one: For a natural number output (i.e., x^2), the possible inputs are x, $-x$, but only one of those can be a natural number (since natural numbers are all positive).

It is not onto: for example, $5 \in \mathbb{N}$ (i.e., the codomain) but no natural number can be put into the function to produce 5.

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

$b)\ f : \mathbb{R} \to \mathbb{R}, f(x) = x^2$

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

$b)\ f : \mathbb{R} \to \mathbb{R}, f(x) = x^2$

For this domain and co-domain, the function is neither one-to-one nor onto.

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

b) $f : \mathbb{R} \to \mathbb{R}, f(x) = x^2$

For this domain and co-domain, the function is neither one-to-one nor onto.

It is not one-to-one: 16 can be produced by both 4, −4 as inputs.

It is not onto, −5 (for example) cannot be produced as output, since real-numbers when squared are always non-negative.

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

$c)\ f : \mathbb{R}^+ \to \mathbb{R}^+, f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

$c)\ f : \mathbb{R}^+ \to \mathbb{R}^+, f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$

For this domain and co-domain, the function is both one-to-one and onto.

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

$c)\ f : \mathbb{R}^+ \to \mathbb{R}^+, f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$

For this domain and co-domain, the function is both one-to-one and onto.

It is one-to-one, if $x^2 = y^2$, then $\pm x = \pm y$, but since all inputs to f are non-negative, we must have that $x = y$.

It is onto, for an arbitrary output z, by def of f, $z = x^2$, that x is a real number, choosing x to be positive will give us a valid element of the domain.

That's All, Folks!

Thanks for coming to section this week!
Any questions?

Problem 8 – One-to-One and Onto

For each of these functions, state whether it is one-to-one, onto, both, or neither.

$c)\ f : \mathbb{R}^+ \to \mathbb{R}^+, f(x) = x^2$, where $\mathbb{R}^+ = \{x : x \in \mathbb{R} \wedge x \geq 0\}$

For this domain and co-domain, the function is both one-to-one and onto.

Structural Induction

Idea of Structural Induction

Every element is built up recursively…

So to show $P(s)$ for all $s \in S$…

Show $P(b)$ for all base case elements b.

Show for an arbitrary element not in the base case, if $P()$ holds for every named element in the recursive rule, then $P()$ holds for the new element (each recursive rule will be a case of this proof).

Structural Induction Template

Let $P(x)$ be “<predicate>”. We show $P(x)$ holds for all $x \in S$ by structural induction.

Base Case: Show $P(x)$
[Do that for every base cases x in S.]

Inductive Hypothesis: Suppose $P(x)$
[Do that for every x listed as in S in the recursive rules.]

Inductive Step: Show $P()$ holds for y.
[You will need a separate case/step for every recursive rule.]

Therefore $P(x)$ holds for all $x \in S$ by the principle of induction.

Problem 2b – Structural Induction on Trees

Definition of Tree:
Basis Step: • is a Tree.
Recursive Step: If L is a Tree and R is a Tree then Tree(•, L, R) is a Tree

Definition of leaves():
leaves(•) = 1
leaves(Tree(•, L, R)) = leaves(L) + leaves(R)

Definition of size():
size(•) = 1
size(Tree(•, L, R)) =1 + size(L) + size(R)

Prove that leaves(T) ≥ size(T)/2 + 1/2 for all Trees T

Work on this problem with the people around you.

Problem 2b – Structural Induction on Trees

Let $P(x)$ be "" for all elements $x \in S$.
We show $P(x)$ holds for all elements $x \in S$ by structural induction.
Base Case: (x= <basis>):
Let y be an arbitrary element not covered by the base cases. By the exclusion rule, y=<recursive rule> for <building blocks of y>.
Inductive Hypothesis: Suppose P(<building blocks of y>) holds for <building blocks>
Inductive Step: Goal: Show $P(y)$ holds:

Conclusion: Therefore $P(x)$ holds for all elements $x \in S$ by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: (x = <basis>):
Let y be an arbitrary element not covered by the base cases. By the exclusion rule, y=<recursive rule> for <building blocks of y>.
Inductive Hypothesis: Suppose P(<building blocks of y>) holds for <building blocks>
Inductive Step: Goal: Show P(y) holds:

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: ($T = \bullet$): By definition of $\text{leaves}(\bullet)$, $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$, so $P(\bullet)$ holds.
Let y be an arbitrary element not covered by the base cases. By the exclusion rule, y=<recursive rule> for <building blocks of y>.
Inductive Hypothesis: Suppose P(<building blocks of y>) holds for <building blocks>
Inductive Step: Goal: Show P(y) holds:

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: ($T = \bullet$): By definition of $\text{leaves}(\bullet)$, $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$, so $P(\bullet)$ holds.
Let Y be an arbitrary tree not covered by the base cases. By the exclusion rule, $Y = \text{Tree}(\bullet, L, R)$ for some trees L and R.
Inductive Hypothesis: Suppose $P(L)$ and $P(R)$ hold for some arbitrary trees L and R.
Inductive Step: Goal: Show P(y) holds:

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: ($T = \bullet$): By definition of $\text{leaves}(\bullet)$, $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$, so $P(\bullet)$ holds.
Let Y be an arbitrary tree not covered by the base cases. By the exclusion rule, $Y = \text{Tree}(\bullet, L, R)$ for some trees L and R.
Inductive Hypothesis: Suppose $P(L)$ and $P(R)$ hold for some arbitrary trees L and R.
Inductive Step: Goal: Show $P(\text{Tree}(\bullet, L, R))$ holds: $\text{leaves}(\text{Tree}(\bullet,L,R)) \geq \text{size}(\text{Tree}(\bullet,L,R))/2 + 1/2$

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: ($T = \bullet$): By definition of $\text{leaves}(\bullet)$, $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$, so $P(\bullet)$ holds.
Let Y be an arbitrary tree not covered by the base cases. By the exclusion rule, $Y = \text{Tree}(\bullet, L, R)$ for some trees L and R.
Inductive Hypothesis: Suppose $P(L)$ and $P(R)$ hold for some arbitrary trees L and R.
Inductive Step: Goal: Show $P(\text{Tree}(\bullet, L, R))$ holds: $\text{leaves}(\text{Tree}(\bullet,L,R)) \geq \text{size}(\text{Tree}(\bullet,L,R))/2 + 1/2$

$$\text{leaves}(Tree(\bullet, L, R)) = \\ ??? \\ = \text{size}(\text{Tree}(\bullet, L, R))/2 + 1/2 \qquad ???$$

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: ($T = \bullet$): By definition of $\text{leaves}(\bullet)$, $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$, so $P(\bullet)$ holds.
Let Y be an arbitrary tree not covered by the base cases. By the exclusion rule, $Y = \text{Tree}(\bullet, L, R)$ for some trees L and R.
Inductive Hypothesis: Suppose $P(L)$ and $P(R)$ hold for some arbitrary trees L and R.
Inductive Step: Goal: Show $P(\text{Tree}(\bullet, L, R))$ holds: $\text{leaves}(\text{Tree}(\bullet,L,R)) \geq \text{size}(\text{Tree}(\bullet,L,R))/2 + 1/2$

$$\begin{aligned} \text{leaves}(Tree(\bullet, L, R)) &= \text{leaves}(L) + \text{leaves}(R) && \text{definition of leaves} \\ &??? \\ &= \text{size}(\text{Tree}(\bullet, L, R))/2 + 1/2 && ??? \end{aligned}$$

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: ($T = \bullet$): By definition of leaves($\bullet$), $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$, so $P(\bullet)$ holds.
Let Y be an arbitrary tree not covered by the base cases. By the exclusion rule, $Y = \text{Tree}(\bullet, L, R)$ for some trees L and R.
Inductive Hypothesis: Suppose $P(L)$ and $P(R)$ hold for some arbitrary trees L and R.
Inductive Step: Goal: Show $P(\text{Tree}(\bullet, L, R))$ holds: $\text{leaves}(\text{Tree}(\bullet,L,R)) \geq \text{size}(\text{Tree}(\bullet,L,R))/2 + 1/2$

$$\begin{aligned} \text{leaves}(Tree(\bullet, L, R)) &= \text{leaves}(L) + \text{leaves}(R) && \text{definition of leaves} \\ &\geq (\text{size}(L)/2 + 1/2) + (\text{size}(R)/2 + 1/2) && \text{by IH} \\ &??? \\ &= \text{size}(\text{Tree}(\bullet, L, R))/2 + 1/2 && ??? \end{aligned}$$

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: ($T = \bullet$): By definition of $\text{leaves}(\bullet)$, $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$, so $P(\bullet)$ holds.
Let Y be an arbitrary tree not covered by the base cases. By the exclusion rule, $Y = \text{Tree}(\bullet, L, R)$ for some trees L and R.
Inductive Hypothesis: Suppose $P(L)$ and $P(R)$ hold for some arbitrary trees L and R.
Inductive Step: Goal: Show $P(\text{Tree}(\bullet, L, R))$ holds: $\text{leaves}(\text{Tree}(\bullet,L,R)) \geq \text{size}(\text{Tree}(\bullet,L,R))/2 + 1/2$

$$
\begin{aligned}
\text{leaves}(Tree(\bullet, L, R)) &= \text{leaves}(L) + \text{leaves}(R) && \text{definition of leaves} \\
&\geq (\text{size}(L)/2 + 1/2) + (\text{size}(R)/2 + 1/2) && \text{by IH} \\
&= (1/2 + \text{size}(L)/2 + \text{size}(R)/2) + 1/2 \\
&??? \\
&= \text{size}(\text{Tree}(\bullet, L, R))/2 + 1/2 && ???
\end{aligned}
$$

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: ($T = \bullet$): By definition of leaves($\bullet$), $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$, so $P(\bullet)$ holds.
Let Y be an arbitrary tree not covered by the base cases. By the exclusion rule, $Y = \text{Tree}(\bullet, L, R)$ for some trees L and R.
Inductive Hypothesis: Suppose $P(L)$ and $P(R)$ hold for some arbitrary trees L and R.
Inductive Step: Goal: Show $P(\text{Tree}(\bullet, L, R))$ holds: $\text{leaves}(\text{Tree}(\bullet,L,R)) \geq \text{size}(\text{Tree}(\bullet,L,R))/2 + 1/2$

$$
\begin{aligned}
\text{leaves}(Tree(\bullet, L, R)) &= \text{leaves}(L) + \text{leaves}(R) && \text{definition of leaves} \\
&\geq (\text{size}(L)/2 + 1/2) + (\text{size}(R)/2 + 1/2) && \text{by IH} \\
&= (1/2 + \text{size}(L)/2 + \text{size}(R)/2) + 1/2 \\
&= (1 + \text{size}(L) + \text{size}(R))/2 + 1/2 \\
&= \text{size}(\text{Tree}(\bullet, L, R))/2 + 1/2 && ???
\end{aligned}
$$

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2b – Structural Induction on Trees

Let $P(T)$ be "$\text{leaves}(T) \geq \text{size}(T)/2 + 1/2$" for all trees T.
We show $P(T)$ holds for all trees T by structural induction.
Base Case: ($T = \bullet$): By definition of $\text{leaves}(\bullet)$, $\text{leaves}(\bullet) = 1$ and $\text{size}(\bullet) = 1$. So, $\text{leaves}(\bullet) = 1 \geq 1/2 + 1/2 = \text{size}(\bullet)/2 + 1/2$, so $P(\bullet)$ holds.
Let Y be an arbitrary tree not covered by the base cases. By the exclusion rule, $Y = \text{Tree}(\bullet, L, R)$ for some trees L and R.
Inductive Hypothesis: Suppose $P(L)$ and $P(R)$ hold for some arbitrary trees L and R.
Inductive Step: Goal: Show $P(\text{Tree}(\bullet, L, R))$ holds: $\text{leaves}(\text{Tree}(\bullet,L,R)) \geq \text{size}(\text{Tree}(\bullet,L,R))/2 + 1/2$

$$
\begin{aligned}
\text{leaves}(Tree(\bullet, L, R)) &= \text{leaves}(L) + \text{leaves}(R) && \text{definition of leaves} \\
&\geq (\text{size}(L)/2 + 1/2) + (\text{size}(R)/2 + 1/2) && \text{by IH} \\
&= (1/2 + \text{size}(L)/2 + \text{size}(R)/2) + 1/2 \\
&= (1 + \text{size}(L) + \text{size}(R))/2 + 1/2 \\
&= \text{size}(\text{Tree}(\bullet, L, R))/2 + 1/2 && \text{definition of size}
\end{aligned}
$$

Conclusion: Therefore $P(T)$ holds for all trees T by the principle of induction.

Problem 2a – Structural Induction on Strings

Definition of string:
Basis Step: "" is a string.
Recursive Step: If X is a string and c is a character then append(c, X) is a string.

Definition of len():
len("") = 0
len(append(c, X)) = 1 + len(X)

Definition of double():
double("") = ""
double(append(c, X)) = append(c, append(c, double(X)))

Prove that for any string X, len(double(X)) = 2len(X).

Work on this problem with the people around you.

Problem 2a – Structural Induction on Strings

Let $P(x)$ be "" for all elements $x \in S$.
We show $P(x)$ holds for all elements $x \in S$ by structural induction.
Base Case: (x= <basis>):
Let y be an arbitrary element not covered by the base cases. By the exclusion rule, y=<recursive rule> for <building blocks of y>.
Inductive Hypothesis: Suppose P(<building blocks of y>) holds for <building blocks>
Inductive Step: Goal: Show $P(y)$ holds:

Conclusion: Therefore $P(x)$ holds for all elements $x \in S$ by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: (x= <basis>):
Let y be an arbitrary element not covered by the base cases. By the exclusion rule, y=<recursive rule> for <building blocks of y>.
Inductive Hypothesis: Suppose P(<building blocks of y>) holds for <building blocks>
Inductive Step: Goal: Show P(y) holds:

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}("")) = \text{len}("") = 0 = 2 \cdot 0 = 2\text{len}("")$, so $P("")$ holds
Let y be an arbitrary element not covered by the base cases. By the exclusion rule, y=<recursive rule> for <building blocks of y>.
Inductive Hypothesis: Suppose P(<building blocks of y>) holds for <building blocks>
Inductive Step: Goal: Show P(y) holds:

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}("")) = \text{len}("") = 0 = 2 \cdot 0 = 2\text{len}("")$, so $P("")$ holds
Let Y be an arbitrary string not covered by the base cases. By the exclusion rule, $Y = \text{append}(c, Z)$ for some character c and string Z.
Inductive Hypothesis: Suppose $P(Z)$ holds for some arbitrary string Z.
Inductive Step: Goal: Show P(y) holds:

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}("")) = \text{len}("") = 0 = 2 \cdot 0 = 2\text{len}("")$, so $P("")$ holds
Let Y be an arbitrary string not covered by the base cases. By the exclusion rule, $Y = \text{append}(c, Z)$ for some character c and string Z.
Inductive Hypothesis: Suppose $P(Z)$ holds for some arbitrary string Z.
Inductive Step: Goal: Show $P(\text{append}(c, Z))$ holds: $\text{len}(\text{double}(\text{append}(c, Z))) = 2\text{len}(\text{append}(c, Z))$

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}(""")) = \text{len}(""") = 0 = 2 \cdot 0 = 2\text{len}(""")$, so $P(""")$ holds
Let Y be an arbitrary string not covered by the base cases. By the exclusion rule, $Y = \text{append}(c, Z)$ for some character c and string Z.
Inductive Hypothesis: Suppose $P(Z)$ holds for some arbitrary string Z.
Inductive Step: Goal: Show $P(\text{append}(c, Z))$ holds: $\text{len}(\text{double}(\text{append}(c, Z))) = 2\text{len}(\text{append}(c, Z))$

$$\text{len}(\text{double}(\text{append}(c, Z))) =$$
$$???$$
$$= 2(\text{len}(\text{append}(c, Z))) \qquad ???$$

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}("")) = \text{len}("") = 0 = 2 \cdot 0 = 2\text{len}("")$, so $P("")$ holds
Let Y be an arbitrary string not covered by the base cases. By the exclusion rule, $Y = \text{append}(c, Z)$ for some character c and string Z.
Inductive Hypothesis: Suppose $P(Z)$ holds for some arbitrary string Z.
Inductive Step: Goal: Show $P(\text{append}(c, Z))$ holds: $\text{len}(\text{double}(\text{append}(c, Z))) = 2\text{len}(\text{append}(c, Z))$

$$\begin{aligned} \text{len}(\text{double}(\text{append}(c, Z))) &= \text{len}(\text{append}(c, \text{append}(c, \text{double}(Z)))) && \text{def. of double} \\ &\overset{???}{=} 2(\text{len}(\text{append}(c, Z))) && ??? \end{aligned}$$

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}(""")) = \text{len}(""") = 0 = 2 \cdot 0 = 2\text{len}(""")$, so $P(""")$ holds
Let Y be an arbitrary string not covered by the base cases. By the exclusion rule, $Y = \text{append}(c, Z)$ for some character c and string Z.
Inductive Hypothesis: Suppose $P(Z)$ holds for some arbitrary string Z.
Inductive Step: Goal: Show $P(\text{append}(c, Z))$ holds: $\text{len}(\text{double}(\text{append})(c, Z)) = 2\text{len}(\text{append}(c, Z))$

$$\begin{aligned} \text{len}(\text{double}(\text{append}(c, Z))) &= \text{len}(\text{append}(c, \text{append}(c, \text{double}(Z)))) && \text{def. of double} \\ &= 1 + \text{len}(\text{append}(c, \text{double}(Z))) && \text{def. of len} \\ &??? \\ &= 2(\text{len}(\text{append}(c, Z))) && ??? \end{aligned}$$

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}(""")) = \text{len}(""") = 0 = 2 \cdot 0 = 2\text{len}(""")$, so $P(""")$ holds
Let Y be an arbitrary string not covered by the base cases. By the exclusion rule, $Y = \text{append}(c, Z)$ for some character c and string Z.
Inductive Hypothesis: Suppose $P(Z)$ holds for some arbitrary string Z.
Inductive Step: Goal: Show $P(\text{append}(c, Z))$ holds: $\text{len}(\text{double}(\text{append}(c, Z))) = 2\text{len}(\text{append}(c, Z))$

$$
\begin{aligned}
\text{len}(\text{double}(\text{append}(c, Z))) &= \text{len}(\text{append}(c, \text{append}(c, \text{double}(Z)))) && \text{def. of double} \\
&= 1 + \text{len}(\text{append}(c, \text{double}(Z))) && \text{def. of len} \\
&= 1 + 1 + \text{len}(\text{double}(Z)) && \text{def. of len} \\
&??? \\
&= 2(\text{len}(\text{append}(c, Z))) && ???
\end{aligned}
$$

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}("")) = \text{len}("") = 0 = 2 \cdot 0 = 2\text{len}("")$, so $P("")$ holds
Let Y be an arbitrary string not covered by the base cases. By the exclusion rule, $Y = \text{append}(c, Z)$ for some character c and string Z.
Inductive Hypothesis: Suppose $P(Z)$ holds for some arbitrary string Z.
Inductive Step: Goal: Show $P(\text{append}(c, Z))$ holds: $\text{len}(\text{double}(\text{append}(c, Z))) = 2\text{len}(\text{append}(c, Z))$

$$
\begin{aligned}
\text{len}(\text{double}(\text{append}(c, Z))) &= \text{len}(\text{append}(c, \text{append}(c, \text{double}(Z)))) && \text{def. of double} \\
&= 1 + \text{len}(\text{append}(c, \text{double}(Z))) && \text{def. of len} \\
&= 1 + 1 + \text{len}(\text{double}(Z)) && \text{def. of len} \\
&= 2 + 2\text{len}(Z) && \text{by I.H.} \\
&??? \\
&= 2(\text{len}(\text{append}(c, Z))) && ???
\end{aligned}
$$

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}(""")) = \text{len}(""") = 0 = 2 \cdot 0 = 2\text{len}(""")$, so $P(""")$ holds
Let Y be an arbitrary string not covered by the base cases. By the exclusion rule, $Y = \text{append}(c, Z)$ for some character c and string Z.
Inductive Hypothesis: Suppose $P(Z)$ holds for some arbitrary string Z.
Inductive Step: Goal: Show $P(\text{append}(c, Z))$ holds: $\text{len}(\text{double}(\text{append}(c, Z))) = 2\text{len}(\text{append}(c, Z))$

$$
\begin{aligned}
\text{len}(\text{double}(\text{append}(c, Z))) &= \text{len}(\text{append}(c, \text{append}(c, \text{double}(Z)))) && \text{def. of double} \\
&= 1 + \text{len}(\text{append}(c, \text{double}(Z))) && \text{def. of len} \\
&= 1 + 1 + \text{len}(\text{double}(Z)) && \text{def. of len} \\
&= 2 + 2\text{len}(Z) && \text{by I.H.} \\
&= 2(1 + \text{len}(Z)) \\
&= 2(\text{len}(\text{append}(c, Z))) && ???
\end{aligned}
$$

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Problem 2a – Structural Induction on Strings

Let $P(X)$ be "$\text{len}(\text{double}(X)) = 2\text{len}(X)$" for all strings X.
We show $P(X)$ holds for all strings X by structural induction.
Base Case: ($X = ""$): By definition, $\text{len}(\text{double}(""")) = \text{len}(""") = 0 = 2 \cdot 0 = 2\text{len}(""")$, so $P(""")$ holds
Let Y be an arbitrary string not covered by the base cases. By the exclusion rule, $Y = \text{append}(c, Z)$ for some character c and string Z.
Inductive Hypothesis: Suppose $P(Z)$ holds for some arbitrary string Z.
Inductive Step: Goal: Show $P(\text{append}(c, Z))$ holds: $\text{len}(\text{double}(\text{append}(c, Z))) = 2\text{len}(\text{append}(c, Z))$

$$
\begin{aligned}
\text{len}(\text{double}(\text{append}(c, Z))) &= \text{len}(\text{append}(c, \text{append}(c, \text{double}(Z)))) && \text{def. of double} \\
&= 1 + \text{len}(\text{append}(c, \text{double}(Z))) && \text{def. of len} \\
&= 1 + 1 + \text{len}(\text{double}(Z)) && \text{def. of len} \\
&= 2 + 2\text{len}(Z) && \text{by I.H.} \\
&= 2(1 + \text{len}(Z)) \\
&= 2(\text{len}(\text{append}(c, Z))) && \text{def. of len}
\end{aligned}
$$

Conclusion: Therefore $P(X)$ holds for all strings X by the principle of induction.

Regular Expressions

Regular Expressions

Basis:

- ε is a regular expression. The empty string itself matches the pattern (and nothing else does).
- $\varnothing$ is a regular expression. No strings match this pattern.
- a is a regular expression, for any $a \in \Sigma$ (i.e. any character). The character itself matching this pattern.

Recursive:

- If A, B are regular expressions then $(A \cup B)$ is a regular expression. matched by any string that matches A or that matches B [or both]).
- If A, B are regular expressions then AB is a regular expression. matched by any string x such that $x = yz$, y matches A and z matches B.
- If A is a regular expression, then $A*$ is a regular expression. matched by any string that can be divided into 0 or more strings that match A.

Regular Expressions

A regular expression is a recursively defined set of strings that form a language.

A regular expression will generate all strings in a language, and won't generate any strings that ARE NOT in the language

Hints:

- Come up with a few examples of strings that ARE and ARE NOT in your language
- Then, after you write your regex, check to make sure that it CAN generate all of your examples that are in the language, and it CAN'T generate those that are not

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

e) Write a regular expression that matches all binary strings of the form 1^{ky}, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1's.

Work on this problem with the people around you.

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

base-10 numbers:

Our everyday numbers! Notice we have 10 symbols (0-9) to represent numbers.

256: $(\mathbf{2} * 10^2) + (\mathbf{5} * 10^1) + (\mathbf{6} * 10^0)$

base-2 numbers: (binary)

10: $(\mathbf{1} * 2^1) + (\mathbf{0} * 2^0)$

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)*

⚠️ “0101” or “091” **are not** Base-10 numbers

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$$

⚠️ “0” **is** a Base-10 number not considered

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0” **is** a Base-10 number not considered

All possible *strings* using numbers 0-9 that never start with 0 *or* is 0

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0” **is** a Base-10 number not considered

All possible *strings* using numbers 0-9 that never start with 0 *or* is 0

$0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*)$

Problem 1 – Regular Expressions

a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Representing numbers all possible *strings* using numbers 0-9:

$(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0101” or “091” **are not** Base-10 numbers

All possible *strings* using numbers 0-9 that never start with 0

$(1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*$

⚠️ “0” **is** a Base-10 number not considered

All possible *strings* using numbers 0-9 that never start with 0 *or* is 0

$0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)*)$

✅Generates only all possible Base-10 numbers

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

✅Generates only all possible Base-3 numbers

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$$

Generates only all possible Base-3 numbers

…divisible by 3

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

Generates only all possible Base-3 numbers

…divisible by 3

Hint: you know that Base-10 numbers are divisible by 10 when they end in 0 (10, 20, 30, 40…)

Problem 1 – Regular Expressions

b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Write a regular expression that matches all base-3 numbers

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*)$

Generates only all possible Base-3 numbers

…divisible by 3

Hint: you know that Base-10 numbers are divisible by 10 when they end in 0 (10, 20, 30, 40…)

$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)*\mathbf{0})$

✅all possible Base-3 numbers divisible by 3

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

$(0 \cup 1)^* \; 111 \; (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

$(0 \cup 1)^* 111 (0 \cup 1)^*$

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠ Generates “000” like “00 01 111”

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠️ Generates “000” like “00 01 111”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)

✅ all binary strings with “111” and without “000”

Problem 1 – Regular Expressions

c) Write a regular expression that matches all binary strings that contain the substring “111”, but not the substring “000”.

all binary strings that contain the substring “111”

(0 ∪ 1)* 111 (0 ∪ 1)*

⚠️ The Kleene-star has us generating any number of 0’s

…without the substring “000”

Use careful case-work to modify this and produce only 0,1,or two 0’s

(0 ∪ 00 ∪ ε) (1)* 111 **(0 ∪ 00 ∪ ε)** (1)*

⚠️ Cannot produce 1’s with “0” or “00” like “1011101”

(0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)*** 111 (0 ∪ 00 ∪ ε) **(01 ∪ 001 ∪ 1)***

⚠️ Generates “000” like “00 01 111”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)

✅ all binary strings with “111” and without “000”

(01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε) 111 (01 ∪ 001 ∪ 1)* (0 ∪ 00 ∪ ε)

Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 1: Write out basic and more intricate cases

Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 1: Write out basic and more intricate cases

Accepted Strings	Rejected Strings
ε	00
1	11
10101	101011
0101	0100

Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 1: Write out basic and more intricate cases

Accepted Strings	Rejected Strings
ε	00
1	11
10101	101011
0101	0100

Step 2: Find a pattern!

Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 1: Write out basic and more intricate cases

Accepted Strings	Rejected Strings
ε	00
1	11
10101	10101**1**
0101	010**0**

Step 2: Find a pattern!

strings can be generated from **either a series** of **"01" or "10" substrings**

(1) Using the "01" substring, one additional 0 can be added

(1) Using the "10" substring, one additional 1 can be added

Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 3: Write out the expression with the two cases we found

Problem 1 – Regular Expressions

d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Step 3: Write out the expression with the two cases we found

$((01)* (0 \cup \varepsilon)) \cup ((10)* (1 \cup \varepsilon))$

Problem 1 – Regular Expressions

e) Write a regular expression that matches all binary strings of the form 1^{ky}, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1's.

Problem 1 – Regular Expressions

e) Write a regular expression that matches all binary strings of the form 1^{ky}, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1's.

$1(0 \cup 1)* 1(0 \cup 1)*$

Explanation: While it may seem like we need to keep track of how many 1's there are, it turns out that we don't. Convince yourself that strings in the language are exactly those of the form 1x, where x is any binary string with at least one 1. Hence, x is matched by the regular expression $(0 \cup 1)*1(0 \cup 1)*$