ASPC-A 2023 Day 5 (AM) Problem Set Solutions

This is the **solutions manual** for the **problem set for Day 5 (AM)**.

This solutions manual has **6 problems**. The items in the problem list below are links to each problem. You can click a problem name to jump to its page in the PDF file.

Solution for F91

We implement the given recurrence.

Solution 1:

```
def f91( n ):
      if n <= 100:
            return f91(f91(n+11))
      else:
            return n - 10

n = int(input())
print( f91(n) )
```

Incidentally, there is a shortcut. It turns out that the f91 function simplifies to the following:

Solution 2:

```
n = int(input())
if n <= 100:
      print(91)
else:
      print(n – 10)
```

Solution for Do Androids Dream?

We implement the given recurrence.

Solution:

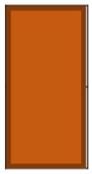
```
def check(n):
      if n==7 or n==9:
            return True
      elif n==5:
            return False
      elif n<=0:
            return False
      elif n%3==0:
            return check(n//3)
      elif n%8==0:
            return check(n//8)
      else:
            return check(n-20)

n = int(input())
if check(n):
      print( "YES" )
else:
      print( "NO" )
```

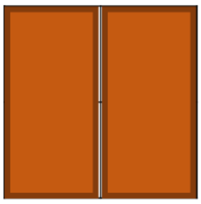
Solution for Chocolate Box

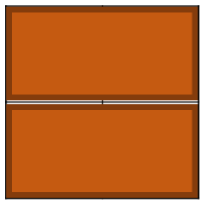
Let's consider small cases:

If $n = 1$, there is only one way to fill up the box.



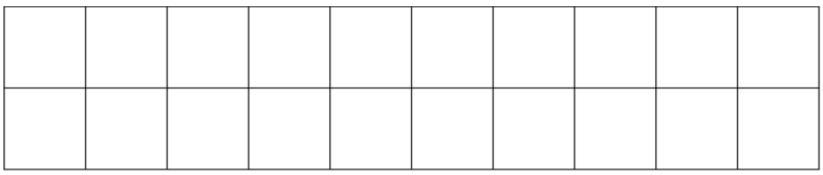
If $n = 2$, there are two ways to fill up the box.



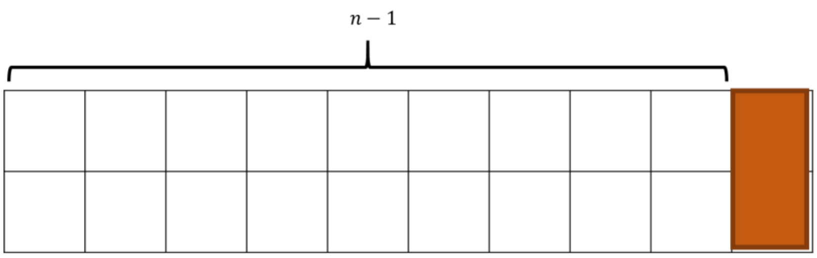


We then think about how to break down big cases.

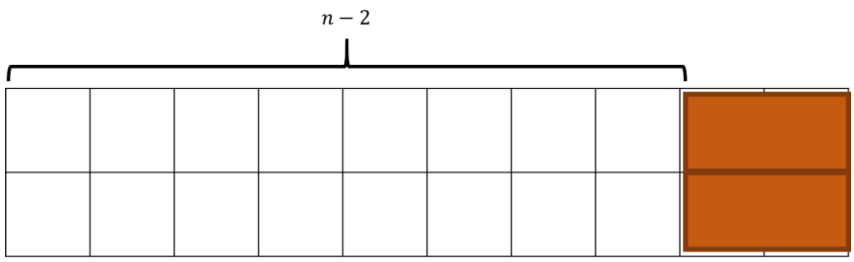
If we have a $2 \times n$ box:



We can either place a single chocolate like this and then fill up the remaining $2 \times n - 1$ box:



Or we can place two chocolates like this and then fill up the remaining $2 \times n - 2$ box:



We thus have the following recurrence:

Solution:

```
def fib( n ):
      if n == 1:
            return 1
      elif n == 2:
            return 2
      else:
            return fib(n-1) + fib(n-2)

n = int(input())
print( fib(n) )
```

Solution for Binary Tree

We start from the root, and print out its children.

Our base case is when the node we are printing is a leaf.

Solution:

```
def output(left, right, r ):
	if left[r] == -1 or right[r] == -1:
		# node is a leaf
		return str(r)
	else:
		return str(r) + "[" + output(left,right,left[r]) + "," + 
output(left,right,right[r]) + "]"

n = int(input())
left = [int(i) for i in input().split()]
right = [int(i) for i in input().split()]

print( output(left, right, 0) )
```

Solution for Sequence Verifier

You can solve this problem with 7 counters, or you can make the following observation:

If you sort a valid list, you should get [0,1,2,3,4,5,6].

Solution:

```
pieces = [int(i) for i in input().split()]
pieces.sort()
answer = True
for i in range(7):
	if 0 <= i <= 6:
		if pieces[i] != i:
			answer = False
	if not 0 <= i <= 6:
		answer = False
if answer:
	print("Sequence Valid")
else:
	print("Please debug")
```

Solution for Cipher Substitution

For every line of input, create a new line of text by searching for the letter in the plaintext.

Solution:

```
plaintext = input()
ciphertext = input()
print(ciphertext)
print(plaintext)
for t in range(8):
	line = input()
	decoded = ""
	for x in line:
		encodedchar = x
		for i in range(52):
			if plaintext[i] == x:
				encodedchar = ciphertext[i]
		decoded = decoded + encodedchar
	print(decoded)
```