Module 9: Recurrence Relations

1 Given the recurrence relation $a_{n+1} = 3a_n - 2a_{n-1}$ and initial values $a_0 = 0$ and $a_1 = 1$, what is a_5?

Solution:

$a_2 = 3a_1 - 2a_0 = 3(1) - 2(0) = 3$
$a_3 = 3a_2 - 2a_1 = 3(3) - 2(1) = 7$
$a_4 = 3a_3 - 2a_2 = 3(7) - 2(3) = 15$
$a_5 = 3a_4 - 2a_3 = 3(15) - 2(7) = 31$

2 Find p_6 for $p_n = 4p_{n-1} - 2n - 3$ and $p_0 = 2$.

Solution:

$p_1 = 4p_0 - 2(1) - 3 = 4(2) - 2 - 3 = 3$
$p_2 = 4p_1 - 2(2) - 3 = 4(3) - 4 - 3 = 5$
$p_3 = 4p_2 - 2(3) - 3 = 4(5) - 6 - 3 = 11$
$p_4 = 4p_3 - 2(4) - 3 = 4(11) - 8 - 3 = 33$
$p_5 = 4p_4 - 2(5) - 3 = 4(33) - 10 - 3 = 119$
$p_6 = 4p_5 - 2(6) - 3 = 4(119) - 12 - 3 = 461$

Module 10: Induction Methods

For questions 3 through 5, provide the following information:

- What is the base case? If the problem requires strong induction, you might need several base cases.
- What is the induction hypothesis? If the problem requires strong induction make sure you include that in your induction hypothesis.
- What needs to be proved to complete the induction step?
- For an extra challenge, complete the proof.

3. For all natural numbers n, the following equality holds:

$$11 + 19 + 27 + \cdots + (8n + 3) = n(4n + 7).$$

Solution:

Base Case: $n = 1$, $11 = 1(4(1) + 7)$ ✓

Inductive Hypothesis: Assume $11 + 19 + 27 + \cdots + (8k + 3) = k(4k + 7)$ for positive integer k.

Need to Prove: Show $11+19+27+\cdots+(8k+3)+(8(k+1)+3) = (k+1)(4(k+1)+7)$.

4. For all natural number n, 3 evenly divides $7^n - 4^n$.

Solution:

Base Case: $n = 1$, 3 divides $7^1 - 4^1 = 3$ ✓

Inductive Hypothesis: Assume 3 evenly divides $7^k - 4^k$ for positive integer k.

Need to Prove: Show 3 evenly divides $7^{k+1} - 4^{k+1}$

5. For all integers n larger than 2 the following inequality holds: $2^n > 2n$

Solution:

Base Case: $n = 3$, $2^3 > 2(3)$ ✓

Inductive Hypothesis: Assume $2^k > 2k$ for integer $k \geq 3$

Need to Prove: Show $2^{k+1} > 2(k + 1)$.

Module 11: Recursive Structures

6 The following algorithm computes the GCD of two integers. The function `remainder(n,m)` calculates the remainder of the division n divided by m ($n = qm + r$).

Algorithm 1 GCD

```
procedure GCD(n,m)
r := remainder(n,m)
if r == 0
  return m
else
  GCD(m,r)
```

(a) If you execute the command GCD(21,99) how many times, in total, will the function GCD be called?

(b) What are the inputs of each of those calls?

(c) Will GCD(n,m) always equal GCD(m,n)?

(d) If $n > m$ which will call GCD more times, GCD(n,m) or GCD(m,n)?

Solution:

(a) GCD will be called a total of 5 times. The initial call and then it will call itself an additional 4 times.

(b) GCD(21,99) GCD(99,21), GCD(21,15), GCD(15,6), GCD(6,3).

(c) Yes.

(d) The first call of GCD(21,99) is GCD(99,21) since $21 = 0 \cdot 99 + 21$. A similar thing will happen whenever the first entry is smaller than the second. Therefore, there is always exactly one more call in GCD(m,n) than GCD(n,m).

7 Evaluate the following recursive algorithm at n = 6 and n = 7. Explain, in your own words, what this algorithm does.

Algorithm 2 Double Factorial

```
procedure DFactorial(n)
input:  a non-negative integer n

if n = 1 or 0 return 1
r := DFactorial(n - 2)
  return (r*n)
```

Solution: `DFactorial(6)` calls on the function for $n = 6, 4, 2, 0$. `DFactorial(0)` $= 1$, then
`DFactorial(2)` $= 2 *$ `Dfactorial(0)` $= 2$, `DFactorial(4)` $= 4 *$ `Dfactorial(2)` $= 8$, finally
`DFactorial(6)` $= 6 *$ `Dfactorial(4)` $= 48$.

`DFactorial(7)` calls on the function for $n = 7, 5, 3, 1$. `DFactorial(1)` $= 1$, then `DFactorial(3)` $= 3 *$ `Dfactorial(1)` $= 3$, `DFactorial(5)` $= 5 *$ `Dfactorial(3)` $= 15$, finally `DFactorial(7)` $= 7 *$ `Dfactorial(5)` $= 105$.

In general, `Dfactorial(n)` is the product of every other positive integer between 1 and n, starting with n and going down. Alternatively it is the product of every integer of the same parity (even or odd) as n between 1 and n.

8 Given the recursive algorithm in this pseudocode.

Algorithm 3 a wild function

```
procedure WILD(n)
input:  a positive integer n.

if n < 3
  return 1
if n ≥ 3
    t:  = wild (n - 1)
    r:  = wild (n - 2)
    if n is odd
      return r+t
    if n is even
      return r*t
```

What is the output of `wild(8)`? How many times, in total, is `wild` called?

Solution: We inductively calculate `wild(n)` starting from 1 moving up to 8.

n	1	2	3	4	5	6	7	8
step	1	1	$1+1$	2 * 1	$2+2$	4 * 2	$8+4$	12*8
`wild(n)`	1	1	2	2	4	8	12	96

As we go through the algorithm, we see that each call of `wild(n)` with $n \geq 3$ calls `wild` twice. Starting from the initial call, `wild(8)` calls `wild(7)` and `wild(6)` and each of those call wild twice again. Also notice that `wild(n)` is called once each time `wild(n + 1)` or `wild(n + 2)` is called. it is helpful to make a binary tree here. In total we count:

`wild(n)`	8	7	6	5	4	3	2	1	total
number of calls	1	1	2	3	5	8	13	8	41

Module 12: Simple Recurrence Relations

9 For the recurrence relation, $d_n = -d_{n-1} + 6d_{n-2}$, answer the following questions.

a. What is the characteristic equation?

b. What are the roots of the characteristic equation?

c. If $d_0 = 1$ and $d_1 = 4$, find the closed form expression for d_n. (Solve the recurrence relation.)

d. What is d_{10}? What is d_{14}?

Solution: Refer to Lesson 3.21 to work through this whole process.

a. $x^2 + x - 6 = 0$

b. $x = 2$ and $x = -3$

c. $d_n = \frac{7}{5}(2)^n - \frac{2}{5}(-3)^n$

d. $d_{10} = \frac{7}{5}(2)^{10} - \frac{2}{5}(-3)^{10} = -22,186$ and $d_{14} = \frac{7}{5}(2)^{14} - \frac{2}{5}(-3)^{14} = -1,890,250$