ECE 404 Homework 3

Elias Talcott

February 6, 2020

Contents

1 Theory Problems

1.1 Problem 1

Show whether or not the set of remainders Z_{12} forms a group with either one of the modulo addition or modulo multiplication operations.

Solution

To show that the set Z_{12} is a group with modulo addition, I will show that there is closure, associativity, an identity element, and an inverse for each element.

Closure means that the result of operations on any set elements is also an element. For example, the result of $(2 + 4)\ \%\ 12 = 6$ and $(6 + 8)\ \%\ 12 = 2$. The result of each operation is in Z_{12}. This holds for every pair of elements in the set.

Associativity means that $(((a + b)\ \%\ 12) + c)\ \%\ 12 = (a + ((b + c)\ \%\ 12)\ \%\ 12$. This also holds for every trio of elements in the set. The order in which modulo addition is applied to the elements has no effect since both orders produce the same result as $(a + b + c)\ \%\ 12$.

The identity element for modulo addition is the same as the identity element for regular addition. If you add 0 to any element in the set, you get the same element back.

Finally, each element in the set has an inverse element. This means for any element in the set, there is an element that can be added to it to produce the identity element. In this case, the inverse for an element n is the element $12 - n$, as the result adds to 12 which is equivalent to 0.

1.2 Problem 2

Compute $gcd(29495, 16983)$ using Euclid's Algorithm. Show all the steps.

Solution

Modular division is used to find $gcd(a, b)$. First, $a\ \%\ b$ is calculated, and then this result becomes the b value for the next round. The previous b value becomes the a value for the next round.

Round	a	b	$a\ \%\ b$
0	29495	16983	12512
1	16983	12512	4471
2	12512	4471	3570
3	4471	3570	901
4	3570	901	867
5	901	867	34
6	867	34	17
7	34	17	0

Based on the Euclid's Algorithm table above, $gcd(29495, 16983) = 17$

1.3 Problem 3

With the help of Bezout's identity, show that if c is a common divisor of two integers $a, b > 0$, then $c \mid gcd(a, b)$ (i.e. c is a divisor of $gcd(a, b)$)

Solution

Bezout's identity says that for any positive integers a and n, $gcd(a, n) = xa + yn$ where x and y are integers. This is to say that the greatest common divisor of two numbers can be written as a linear combination of the numbers.

If an integer c is a divisor of both a and b, then it must also be a divisor or a linear combination of a and b. This means that $c \mid xa + yn$. Since $gcd(a, n) = xa + yn$, it follows that $c \mid gcd(a, n)$.

1.4 Problem 4

Use the Extended Euclid's Algorithm to compute by hand the multiplicative inverse of 25 in Z_{28}. List all of the steps.

Solution

The multiplicative inverse of 25 modulo 28 can be found using the Extended Euclid's Algorithm. The steps of Euclid's Algorithm are rewritten to produce a linear combination of the original two numbers. The coefficient for 25 in the end is its multiplicative inverse

$$\begin{aligned}
\text{gcd}(25, 28) &= \text{gcd}(28, 25) & \text{residue } 25 &= (0 * 28) + (1 * 25) \\
&= \text{gcd}(25, 3) & \text{residue } 3 &= (1 * 28) + (-1 * 25) \\
&= \text{gcd}(3, 1) & \text{residue } 1 &= (1 * 28) + (-9 * 3) \\
&& &= (1 * 28) + (-9 * ((1 * 28) + (-1 * 25))) \\
&& &= (-8 * 28) + (9 * 25)
\end{aligned}$$

Based on the Extended Euclid's Algorithm above, the multiplicative inverse of 25 in Z_{28} is 9. This is confirmed by the result $(25 * 9)$ % $28 = 225$ % $28 = 1$.

1.5 Problem 5

In the following, find the smallest possible integer x. Briefly explain how you found the answer to each. You should solve them without using brute-force methods.

(a) $8x \equiv 11 \pmod{13}$

(b) $5x \equiv 3 \pmod{21}$

(c) $8x \equiv 9 \pmod{7}$

Solution

The solution to each congruence equation can be found by multiplying both sides by the multiplicative inverse of the coefficient on x.

(a) $8x \equiv 11 \pmod{13}$

$$\begin{aligned}
\gcd(8,13) &= \gcd(13,8) \text{ residue } 8 = (0*13)+(1*8)\\
&= \gcd(8,5) \quad \text{residue } 5 = (1*13)+(-1*8)\\
&= \gcd(5,3) \quad \text{residue } 3 = (1*13)+(-2*5)\\
&\qquad\qquad\qquad\qquad\quad = (-1*13)+(2*8)\\
&= \gcd(3,2) \quad \text{residue } 2 = (1*5)+(-1*3)\\
&\qquad\qquad\qquad\qquad\quad = (2*13)+(-3*8)\\
&= \gcd(2,1) \quad \text{residue } 1 = (1*3)+(-1*2)\\
&\qquad\qquad\qquad\qquad\quad = (-3*13)+(5*8)
\end{aligned}$$

Since the multiplicative inverse of 8 modulo 13 is 5, we multiply both sides by 5.

$$\begin{aligned}
x &\equiv 55 \pmod{13}\\
x &\equiv 3 \pmod{13}\\
x &= 3
\end{aligned}$$

(b) $5x \equiv 3 \pmod{21}$

Since the multiplicative inverse of 5 modulo 21 is 17, we multiply both sides by 17.

$$\begin{aligned}
x &\equiv 51 \pmod{21}\\
x &\equiv 9 \pmod{21}\\
x &= 9
\end{aligned}$$

(c) $8x \equiv 9 \pmod{7}$

Since 8 and 9 are greater than 7, they can be replaced with their equivalents modulo 7.

$$\begin{aligned}
x &\equiv 2 \pmod{7}\\
x &= 2
\end{aligned}$$

2 Programming Problem

Write a program that takes as input a small integer n (say, smaller than 50) and determines if Zn is a field or only a commutative ring. Assume that the operators are modulo n addition and modulo n multiplication. The program should prompt the user to enter the number. Depending upon the input n, it should correctly print out either “field” or “ring”.

2.1 Python Code

```
#!/usr/bin/env python3

# Homework Number: 3
# Name: Elias Talcott
# ECN Login: etalcott
# Due Date: February 6, 2020

# Input integer n
n = "-1"
while not n.isdigit() or eval(n) < 1:
    n = input("Enter a positive integer: ")
    if not n.isdigit() or eval(n) < 1:
        print("{} is not a positive integer.".format(n))
n = eval(n)

# Check if n is prime
if n < 2:
    prime = False
elif n == 2:
    prime = True
elif n > 2:
    prime = True
    for i in range(2, n // 2):
        if n % i == 0:
            prime = False
            break

# If n is prime, then Zn is a finite field, else it is a ring
if prime:
    print("field")
else:
    print("ring")
```

2.2 Code Explanation

If an integer n is prime, then its set of residues Z_n along with the addition and multiplication operators form a finite field. This is the case because a prime modulus is relatively prime with each member of its set of residues. If a number is relatively prime with the modulus, then it has a multiplicative inverse (a requirement for a ring to become a field).

My solution simply checks whether the modulus is prime or not. If it is prime, the program prints "field". If it is not prime, the program prints "ring".