Codekata Report:

Name: Deepu Pandey

Email: deepu.24scse1011405@galgotiasuniversity.ac.in

Specialization: School of Computer Science & Engineering

Completion Year: 2028-3rd Sem

Section: Section-7

1. Problem Statement:Compute (a^b) mod m using binary exponentiation.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: C

Source Code:

```
#include <stdio.h>

long long modularExp(long long a, long long b, long long m) {
long long result = 1;
a = a % m;

while (b > 0) {
if (b % 2 == 1) {
result = (result * a) % m;
}
b = b / 2;
a = (a * a) % m;
}

return result;
}

int main() {
long long a, b, m;
scanf("a=%lld,b=%lld,m=%lld", &a, &b, &m);

long long result = modularExp(a, b, m);

printf("%lld\n", result);

return 0;
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
1
```

Compilation Status: Passed

Execution Time:

0.001s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
24
```

Compilation Status: Passed

Execution Time:

0.001s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5

Compilation Status: Passed

Execution Time:

0.001s

2. Problem Statement:Compute 2^n mod m for very large n efficiently.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
def power_mod(n, m):
# Fast exponentiation with modulo
result = 1
base = 2
while n > 0:
if n % 2 == 1:
result = (result * base) % m
n = n // 2
base = (base * base) % m
return result

input_str = input()
parts = input_str.split(',')
n = int(parts[0].split('=')[1])
m = int(parts[1].split('=')[1])
print(power_mod(n, m))
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
24
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
4
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
178116276
```

Compilation Status: Passed

Execution Time:

0.01s

3. Problem Statement:Given a,b,m compute (a^-b) mod m (use modular inverse).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
def power_mod(base, exp, mod):
result = 1
base = base % mod
while exp > 0:
if exp % 2 == 1:
result = (result * base) % mod
exp = exp >> 1
base = (base * base) % mod
return result

input_str = input()
parts = input_str.split(',')
a = int(parts[0].split('=')[1])
b = int(parts[1].split('=')[1])
m = int(parts[2].split('=')[1])

result = power_mod(a, b, m)
print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
9
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5

Compilation Status: Passed

Execution Time:

0.014s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

16

Compilation Status: Passed

Execution Time:

0.01s

4. Problem Statement:Answer q queries: for each (a,b,m) print (a^b) mod m.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
def power_mod(base, exp, mod):
result = 1
base = base % mod
while exp > 0:
if exp % 2 == 1:
result = (result * base) % mod
exp = exp >> 1
base = (base * base) % mod
return result

query = input().strip()
# Remove parentheses and split by comma
query = query.strip('()')
values = list(map(int, query.split(',')))
a, b, m = values[0], values[1], values[2]
result = power_mod(a, b, m)
print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
1
```

Compilation Status: Passed

Execution Time:

0.011s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
24
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5

Compilation Status: Passed

Execution Time:

0.01s

5. Problem Statement:Compute nth Fibonacci number using iterative method.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
n = int(input().split('=')[1])
if n <= 0:
print(0)
elif n == 1:
print(1)
else:
a, b = 0, 1
for _ in range(2, n + 1):
a, b = b, a + b
print(b)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
5
```

Compilation Status: Passed

Execution Time:

0.013s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
55
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
6765
```

Compilation Status: Passed

Execution Time:

0.01s

6. Problem Statement:Compute nth Fibonacci using recursion + memoization.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
memo = {}
def fib(n):
if n in memo:
return memo[n]
if n <= 1:
return n
memo[n] = fib(n-1) + fib(n-2)
return memo[n]

n = int(input().split('=')[1])
print(fib(n))
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
5
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
55
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
6765
```

Compilation Status: Passed

Execution Time:

0.014s

7. Problem Statement:Compute nth Fibonacci in O(log n) using fast doubling.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
def fib_fast_doubling(n):
if n == 0:
return 0
def fib_pair(k):
if k == 0:
return (0, 1)
m = k // 2
a, b = fib_pair(m)
c = a * (2 * b - a)
d = a * a + b * b
if k % 2 == 0:
return (c, d)
else:
return (d, c + d)
return fib_pair(n)[0]

n = int(input().split('=')[1])
print(fib_fast_doubling(n))
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
5
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
55
```

Compilation Status: Passed

Execution Time:

0.013s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

12586269025

Compilation Status: Passed

Execution Time:

0.014s

8. Problem Statement:Compute nth Fibonacci modulo m efficiently.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
line = input().split(',')
n = int(line[0].split('=')[1])
m = int(line[1].split('=')[1])

if n <= 1:
print(n % m)
else:
a, b = 0, 1
for _ in range(2, n + 1):
a, b = b, (a + b) % m
print(b)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

55

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

586268941

Compilation Status: Passed

Execution Time:

0.016s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

918091266

Compilation Status: Passed

Execution Time:

0.147s

9. Problem Statement:Given permutation p (1-indexed) of size n, compute p^k (apply permutation k times) and output resulting array.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: JAVA 8

Source Code:

```
import java.util.*;
class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String input = sc.nextLine();
// Parse input: p=[2,3,1],k=1
String[] parts = input.split(",k=");
String pStr = parts[0].substring(3, parts[0].length() - 1); // Remove "p=[" and "]"
int k = Integer.parseInt(parts[1]);
String[] pArr = pStr.split(",");
int n = pArr.length;
int[] p = new int[n + 1]; // 1-indexed
for (int i = 0; i < n; i++) {
p[i + 1] = Integer.parseInt(pArr[i]);
}
// Apply permutation k times
int[] result = new int[n + 1];
for (int i = 1; i <= n; i++) {
result[i] = i;
}
for (int iter = 0; iter < k; iter++) {
int[] temp = new int[n + 1];
for (int i = 1; i <= n; i++) {
temp[p[i]] = result[i];
}
result = temp;
}
// Output result
System.out.print("[");
for (int i = 1; i <= n; i++) {
System.out.print(result[i]);
if (i < n) System.out.print(",");
}
System.out.println("]");
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[3,1,2]
```

Compilation Status: Passed

Execution Time:

0.087s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[1,2,3,4]
```

Compilation Status: Passed

Execution Time:

0.089s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

[3,4,5,1,2]

Compilation Status: Passed

Execution Time:

0.091s

10. Problem Statement:Apply permutation k times but answer only where element i goes after k applications.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
line = input()
p_str = line[2:line.find('],i')+1]
i_str = line[line.find('i=') + 2:line.find(',k')]
k_str = line[line.find('k=') + 2:]
p = eval(p_str)
i = int(i_str)
k = int(k_str)

pos = i
for _ in range(k):
pos = p[pos - 1]
print(pos)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

3

Compilation Status: Passed

Execution Time:

0.011s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

3

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

4

Compilation Status: Passed

Execution Time:

0.01s

11. Problem Statement:Given permutation p and k up to 1e18, compute p^k efficiently for all positions.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: JAVA 8

Source Code:

```
import java.util.*;

public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String input = sc.nextLine();

// Parse input: p=[2,3,1],k=1
String[] parts = input.split(",k=");
String pStr = parts[0].substring(parts[0].indexOf('['));
long k = Long.parseLong(parts[1]);

// Parse permutation array
pStr = pStr.replaceAll("[\\[\\]]", "");
String[] pArr = pStr.split(",");
int n = pArr.length;
int[] p = new int[n];
for (int i = 0; i < n; i++) {
p[i] = Integer.parseInt(pArr[i].trim());
}

// Build binary lifting table
int maxLog = 60;
int[][] lift = new int[maxLog][n];

// lift[0][i] = p[i]-1 (0-indexed, jump to position p[i] once)
for (int i = 0; i < n; i++) {
lift[0][i] = p[i] - 1;
}

// lift[j][i] = position after 2^j jumps from i
for (int j = 1; j < maxLog; j++) {
for (int i = 0; i < n; i++) {
lift[j][i] = lift[j-1][lift[j-1][i]];
}
}

// For each position, find where we end up after k jumps
int[] result = new int[n];
for (int i = 0; i < n; i++) {
int pos = i;
long remaining = k;
for (int j = 0; j < maxLog && remaining > 0; j++) {
if ((remaining & 1) == 1) {
pos = lift[j][pos];
}
remaining >>= 1;
}
// The value at the ending position in the original array
result[i] = p[pos];
}

// Format output
System.out.print("[");
for (int i = 0; i < n; i++) {
System.out.print(result[i]);
if (i < n - 1) System.out.print(",");
}
System.out.println("]");
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[3,1,2]
```

Compilation Status: Passed

Execution Time:

0.089s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[3,1,2]
```

Compilation Status: Passed

Execution Time:

0.089s

12. Problem Statement:Given permutation p, find minimum k>0 such that p^k is identity (order of permutation).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
import math
from functools import reduce

def gcd(a, b):
while b:
a, b = b, a % b
return a

def lcm(a, b):
return abs(a * b) // gcd(a, b)

def find_order(p):
n = len(p)
visited = [False] * n
cycle_lengths = []

for i in range(n):
if not visited[i]:
cycle_length = 0
j = i
while not visited[j]:
visited[j] = True
j = p[j] - 1  # Convert to 0-indexed
cycle_length += 1
cycle_lengths.append(cycle_length)

# Find LCM of all cycle lengths
result = reduce(lcm, cycle_lengths)
return result

# Read input
input_str = input().strip()
# Parse the permutation from format 'p=[2,3,1]'
p = eval(input_str.split('=')[1])
```

```
# Calculate and print the order
print(find_order(p))
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
3
```

Compilation Status: Passed

Execution Time:

0.013s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
2
```

Compilation Status: Passed

Execution Time:

0.017s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

5

Compilation Status: Passed

Execution Time:

0.017s

13. Problem Statement:Compute gcd(a,b) using Euclidean algorithm.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
user_input = input()
parts = user_input.split(',')
a = int(parts[0].split('=')[1])
b = int(parts[1].split('=')[1])

def gcd(a, b):
while b != 0:
temp = b
b = a % b
a = temp
return a

result = gcd(a, b)
print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
6
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
1
```

Compilation Status: Passed

Execution Time:

0.012s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
2
```

Compilation Status: Passed

Execution Time:

0.015s

14. Problem Statement:Compute lcm(a,b) using gcd to avoid overflow.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
import math

# Read input
user_input = input()

# Parse the input 'a=12,b=15'
parts = user_input.split(',')
a = int(parts[0].split('=')[1])
b = int(parts[1].split('=')[1])

# Calculate GCD
gcd = math.gcd(a, b)

# Calculate LCM using formula: lcm(a,b) = (a*b) / gcd(a,b)
lcm = (a * b) // gcd

# Print result
print(lcm)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
60
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
35
```

Compilation Status: Passed

Execution Time:

0.015s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
500000
```

Compilation Status: Passed

Execution Time:

0.014s

15. Problem Statement:Compute gcd for many pairs quickly (q up to 1e5).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
import math
import ast
```

```
# Read input and parse as list of tuples
userInput = input()
pairs = ast.literal_eval(userInput)

# Compute GCD for each pair and print results
for pair in pairs:
result = math.gcd(pair[0], pair[1])
print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
6
```

Compilation Status: Passed

Execution Time:

0.011s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
1
```

Compilation Status: Passed

Execution Time:

0.013s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
25
```

Compilation Status: Passed

Execution Time:

0.012s

16. Problem Statement:Find x,y such that ax+by=gcd(a,b) (Extended Euclidean).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
def extended_gcd(a, b):
if b == 0:
return a, 1, 0
else:
gcd, x1, y1 = extended_gcd(b, a % b)
x = y1
y = x1 - (a // b) * y1
return gcd, x, y

line = input().split(',')
a = int(line[0].split('=')[1])
b = int(line[1].split('=')[1])
gcd, x, y = extended_gcd(a, b)
print(x, y)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
1 -1
```

Compilation Status: Passed

Execution Time:

0.015s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
4 -1
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
1 -10
```

Compilation Status: Passed

Execution Time:

0.01s

17. Problem Statement:Solve ax+by=c: determine if integer solutions exist and output any one solution.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
import math

# Parse input
input_str = input().strip()
parts = input_str.split(',')
a = int(parts[0].split('=')[1])
b = int(parts[1].split('=')[1])
c = int(parts[2].split('=')[1])

# Extended Euclidean Algorithm
def extended_gcd(a, b):
if b == 0:
return a, 1, 0
gcd, x1, y1 = extended_gcd(b, a % b)
x = y1
y = x1 - (a // b) * y1
return gcd, x, y

# Find GCD and check if solution exists
gcd = math.gcd(a, b)

if c % gcd != 0:
print("No solution")
else:
# Get a particular solution using extended GCD
_, x0, y0 = extended_gcd(a, b)

# Scale the solution
x0 = x0 * (c // gcd)
y0 = y0 * (c // gcd)

# The general solution is: x = x0 + (b/gcd)*k, y = y0 - (a/gcd)*k
# We want to find k such that both x and y have appropriate values
b_step = b // gcd
a_step = a // gcd

# Find k to minimize |x| + |y| or make them positive
# Try to make x positive and small
```

```
if b_step != 0:
k = -x0 // b_step
x = x0 + b_step * k
y = y0 - a_step * k

# Check both k and k+1 to find the best solution
candidates = []
for offset in [-1, 0, 1]:
k_try = k + offset
x_try = x0 + b_step * k_try
y_try = y0 - a_step * k_try
# Verify the solution
if a * x_try + b * y_try == c:
candidates.append((x_try, y_try, abs(x_try) + abs(y_try)))

# Choose the solution with smallest sum of absolute values
# Or prefer positive x
best = min(candidates, key=lambda t: (t[2], -t[0] if t[0] > 0 else float('inf')))
print(best[0], best[1])
else:
print(x0, y0)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2 1

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1 1

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

No solution

Compilation Status: Passed

Execution Time:

0.01s

18. Problem Statement:Find number of integer solutions (x,y) in bounds 0<=x<=X, 0<=y<=Y for ax+by=c.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
# Read input
input_str = input()
parts = input_str.split(',')

# Parse values
a = int(parts[0].split('=')[1])
b = int(parts[1].split('=')[1])
c = int(parts[2].split('=')[1])
X = int(parts[3].split('=')[1])
Y = int(parts[4].split('=')[1])
```

```
count = 0

# Check all possible values of x
for x in range(X + 1):
# For given x, check if y is valid
# ax + by = c => by = c - ax => y = (c - ax) / b
remainder = c - a * x

# Check if y is an integer and within bounds
if remainder >= 0 and remainder % b == 0:
y = remainder // b
if 0 <= y <= Y:
count += 1

print(count)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1

Compilation Status: Passed

Execution Time:

0.014s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1

Compilation Status: Passed

Execution Time:

0.014s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

0

Compilation Status: Passed

Execution Time:

0.011s

19. Problem Statement:Generate all solutions for ax+by=c within given x-range [L,R].

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: JAVA 8

Source Code:

```
import java.util.Scanner;
class Main {   // <-- Rename class to Main
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String input = sc.nextLine().replace(",", " ").trim();
String[] parts = input.split("\\s+");
int a = Integer.parseInt(parts[0].split("=")[1]);
int b = Integer.parseInt(parts[1].split("=")[1]);
int c = Integer.parseInt(parts[2].split("=")[1]);
int L = Integer.parseInt(parts[3].split("=")[1]);
int R = Integer.parseInt(parts[4].split("=")[1]);
boolean found = false;
for (int x = L; x <= R; x++) {
if ((c - a * x) % b == 0) {
```

```
int y = (c - a * x) / b;
System.out.println(x + " " + y);
found = true;
break;
}
}
if (!found) System.out.println("No solution");
sc.close();
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
2 1
```

Compilation Status: Passed

Execution Time:

0.109s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
1 1
```

Compilation Status: Passed

Execution Time:

0.108s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
0 2
```

Compilation Status: Passed

Execution Time:

0.113s

20. Problem Statement:Find solution (x,y) to ax+by=c that minimizes x+y (non-negative solutions).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: C

Source Code:

```
#include <stdio.h>
long long extended_gcd(long long a, long long b, long long *x, long long *y) {
if (b == 0) { *x = 1; *y = 0; return a; }
long long x1, y1;
long long g = extended_gcd(b, a % b, &x1, &y1);
*x = y1;
*y = x1 - (a / b) * y1;
return g;
}
long long ceil_div(long long a, long long b) {
if (b < 0) { a = -a; b = -b; }
return (a + b - 1) / b;
}
long long floor_div(long long a, long long b) {
if (b < 0) { a = -a; b = -b; }
return a / b;
}
int main() {
long long a, b, c;
scanf("a=%lld,b=%lld,c=%lld", &a, &b, &c);
long long x0, y0;
long long g = extended_gcd(a, b, &x0, &y0);
if (c % g != 0) {
printf("No solution\n");
return 0;
}
x0 *= c / g;
y0 *= c / g;
long long step_x = b / g;
long long step_y = a / g;
long long t_min = ceil_div(-x0, step_x);
long long t_max = floor_div(y0, step_y);
if (t_min > t_max) {
printf("No non-negative solution\n");
return 0;
}
long long t;
if (step_x - step_y > 0) t = t_min;
else t = t_max;
long long x = x0 + step_x * t;
long long y = y0 - step_y * t;
printf("%lld %lld\n", x, y);
return 0;
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
2 1
```

Compilation Status: Passed

Execution Time:

0.001s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

0 4

Compilation Status: Passed

Execution Time:

0.001s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

No solution

Compilation Status: Passed

Execution Time:

0.001s

21. Problem Statement:Encode a positive integer N using Fibonacci coding (Zeckendorf representation).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
# Read input
n = int(input().split('=')[1])

# Generate Fibonacci numbers up to n
fib = [1, 2]
while fib[-1] < n:
fib.append(fib[-1] + fib[-2])
```

```
# Build Zeckendorf representation
result = []
for i in range(len(fib) - 1, -1, -1):
if fib[i] <= n:
result.append('1')
n -= fib[i]
elif result:  # Only add 0 if we've already added a 1
result.append('0')

# Add trailing 1 for Fibonacci coding
result.append('1')
print(''.join(result))
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
100101
```

Compilation Status: Passed

Execution Time:

0.009s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
11
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
10000101001
```

Compilation Status: Passed

Execution Time:

0.01s

22. Problem Statement:Given n, compute nth Fibonacci using closed-form (double) for small n.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
import math

# Read input
input_str = input()
n = int(input_str.split('=')[1])

# Binet's formula
phi = (1 + math.sqrt(5)) / 2
psi = (1 - math.sqrt(5)) / 2

fib = (phi**n - psi**n) / math.sqrt(5)

# Round to nearest integer
result = round(fib)
print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
5
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
55
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
6765
```

Compilation Status: Passed

Execution Time:

0.011s

23. Problem Statement:Compute nth Fibonacci using matrix exponentiation (O(log n)).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
def matrix_mult(A, B, m):
result = [[0, 0], [0, 0]]
for i in range(2):
for j in range(2):
for k in range(2):
result[i][j] = (result[i][j] + A[i][k] * B[k][j]) % m
return result

def matrix_power(matrix, n, m):
if n == 0:
return [[1, 0], [0, 1]]
if n == 1:
return matrix

if n % 2 == 0:
half = matrix_power(matrix, n // 2, m)
return matrix_mult(half, half, m)
else:
return matrix_mult(matrix, matrix_power(matrix, n - 1, m), m)

def fibonacci(n, m):
if n == 0:
return 0
if n == 1:
return 1

base_matrix = [[1, 1], [1, 0]]
result_matrix = matrix_power(base_matrix, n, m)
return result_matrix[0][1]

userInput = input()
parts = userInput.split(',')
n = int(parts[0].split('=')[1])
m = int(parts[1].split('=')[1])
```

```
print(fibonacci(n, m))
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
55
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
586268941
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

918091266

Compilation Status: Passed

Execution Time:

0.01s

24. Problem Statement:Find Pisano period (period of Fibonacci modulo m) for given m.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
# Read input
user_input = input()
m = int(user_input.split('=')[1])

# Generate Fibonacci sequence modulo m until it repeats
fib_mod = [0, 1]
while True:
next_val = (fib_mod[-1] + fib_mod[-2]) % m
fib_mod.append(next_val)

# Check if we found the period (sequence starts repeating with 0, 1)
if len(fib_mod) > 2 and fib_mod[-2] == 0 and fib_mod[-1] == 1:
fib_mod = fib_mod[:-2]  # Remove the repeated 0, 1
break

# Output the result
period = len(fib_mod)
sequence = ' '.join(map(str, fib_mod))
print(f"Period: {period}")
print(f"Sequence: {sequence}")
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
Period: 20
Sequence: 0 1 1 2 3 0 3 3 1 4 0 4 4 3 2 0 2 2 4 1
```

Compilation Status: Passed

Execution Time:

0.013s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
Period: 8
Sequence: 0 1 1 2 0 2 2 1
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
Period: 60
Sequence: 0 1 1 2 3 5 8 3 1 4 5 9 4 3 7 0 7 7 4 1 5 6 1 7 8 5 3 8 1 9 0 9 9 8 7 5 2 7 9 6 5
1 6 7 3 0 3 3 6 9 5 4 9 3 2 5 7 2 9 1
```

Compilation Status: Passed

Execution Time:

0.01s

25. Problem Statement:Sieve of Eratosthenes: list primes ≤ n.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
# Read input
user_input = input()
n = int(user_input.split('=')[1])

# Sieve of Eratosthenes
def sieve_of_eratosthenes(n):
if n < 2:
return []

# Create a boolean array and initialize all entries as true
prime = [True] * (n + 1)
prime[0] = prime[1] = False

p = 2
while p * p <= n:
# If prime[p] is not changed, then it is a prime
if prime[p]:
# Update all multiples of p
for i in range(p * p, n + 1, p):
prime[i] = False
p += 1

# Collect all prime numbers
primes = [i for i in range(2, n + 1) if prime[i]]
return primes

# Get primes and print as list without spaces
primes = sieve_of_eratosthenes(n)
print('[' + ','.join(map(str, primes)) + ']')
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[2,3,5,7]
```

Compilation Status: Passed

Execution Time:

0.011s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[2,3,5,7,11,13,17,19]
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[]
```

Compilation Status: Passed

Execution Time:

0.01s

26. Problem Statement:Linear sieve (Euler sieve) to compute

smallest prime factor and primes ≤ n.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
s = input().strip()  # e.g. n=10
n = int(s.split('=')[1])

if n < 2:
print("[]")
else:
primes = []
lp = [0] * (n + 1)  # lowest prime factor

for i in range(2, n + 1):
if lp[i] == 0:
lp[i] = i
primes.append(i)
for p in primes:
if p > lp[i] or i * p > n:
break
lp[i * p] = p

print("[" + ",".join(map(str, primes)) + "]")
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[2,3,5,7]
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[2,3,5,7,11,13,17,19]
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]
```

Compilation Status: Passed

Execution Time:

0.014s

27. Problem Statement:Primality test: implement Miller-Rabin deterministic for 64-bit numbers.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: JAVA 8

Source Code:

```
import java.util.*;

public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String input = sc.nextLine();

// Parse input: n=2
long n = Long.parseLong(input.substring(input.indexOf('=') + 1));

boolean result = isPrime(n);
System.out.println(result);
}

static boolean isPrime(long n) {
if (n < 2) return false;
if (n == 2 || n == 3) return true;
if (n % 2 == 0) return false;

// Write n-1 as d * 2^r
long d = n - 1;
int r = 0;
while (d % 2 == 0) {
d /= 2;
r++;
}

// Deterministic witnesses for 64-bit numbers
long[] witnesses = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

for (long a : witnesses) {
if (a >= n) continue;
if (!millerTest(n, d, r, a)) {
return false;
}
}
return true;
}

static boolean millerTest(long n, long d, int r, long a) {
long x = modPow(a, d, n);

if (x == 1 || x == n - 1) return true;

for (int i = 0; i < r - 1; i++) {
x = modMul(x, x, n);
if (x == n - 1) return true;
}
return false;
}

static long modPow(long base, long exp, long mod) {
```

```
long result = 1;
base %= mod;
while (exp > 0) {
if (exp % 2 == 1) {
result = modMul(result, base, mod);
}
base = modMul(base, base, mod);
exp /= 2;
}
return result;
}

static long modMul(long a, long b, long mod) {
return ((a % mod) * (b % mod)) % mod;
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
true
```

Compilation Status: Passed

Execution Time:

0.096s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
true
```

Compilation Status: Passed

Execution Time:

0.097s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

true

Compilation Status: Passed

Execution Time:

0.092s

28. Problem Statement:Integer factorization (pollard rho simple) for composite n up to 64-bit.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: JAVA 8

Source Code:

```
import java.util.*;
import java.math.BigInteger;
public class Main {  // Rename to Main for online judge
static final Random rnd = new Random();
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String s = sc.nextLine().trim();
sc.close();
// Parse input like n=12345 or just 12345
if (s.startsWith("n=")) s = s.substring(2);
if (s.isEmpty()) return;
BigInteger n = new BigInteger(s);
if (n.compareTo(BigInteger.ONE) <= 0) {
System.out.println("[]");
```

```
return;
}
List<BigInteger> primes = factorAll(n);
Collections.sort(primes);
StringBuilder sb = new StringBuilder();
sb.append("[");
for (int i = 0; i < primes.size(); i++) {
if (i > 0) sb.append(",");
sb.append(primes.get(i).toString());
}
sb.append("]");
System.out.println(sb.toString());
}
// Factor n into prime factors using Pollard's Rho
static List<BigInteger> factorAll(BigInteger n) {
List<BigInteger> found = new ArrayList<>();
Deque<BigInteger> stack = new ArrayDeque<>();
stack.push(n);
while (!stack.isEmpty()) {
BigInteger m = stack.pop();
if (m.equals(BigInteger.ONE)) continue;
if (m.isProbablePrime(40)) {
if (!found.contains(m)) found.add(m);
continue;
}
BigInteger d = rhoFactor(m);
if (d == null || d.equals(m)) {
BigInteger small = smallDivisor(m);
if (small == null) {  // Prime
found.add(m);
continue;
}
d = small;
}
stack.push(d);
stack.push(m.divide(d));
}
return found;
}
// Check small divisors up to 100000
static BigInteger smallDivisor(BigInteger n) {
BigInteger two = BigInteger.valueOf(2);
if (n.mod(two).equals(BigInteger.ZERO)) return two;
BigInteger limit = BigInteger.valueOf(100000);
for (BigInteger i = BigInteger.valueOf(3); i.compareTo(limit) <= 0; i = i.add(two)) {
if (n.mod(i).equals(BigInteger.ZERO)) return i;
}
return null;
}
// Pollard's Rho factorization
static BigInteger rhoFactor(BigInteger n) {
if (n.mod(BigInteger.valueOf(2)).equals(BigInteger.ZERO)) return
BigInteger.valueOf(2);
BigInteger c = new BigInteger(n.bitLength(),
rnd).mod(n.subtract(BigInteger.ONE)).add(BigInteger.ONE);
BigInteger x = new BigInteger(n.bitLength(),
rnd).mod(n.subtract(BigInteger.ONE)).add(BigInteger.ONE);
BigInteger y = x;
BigInteger d = BigInteger.ONE;
while (d.equals(BigInteger.ONE)) {
x = x.multiply(x).mod(n).add(c).mod(n);
y = y.multiply(y).mod(n).add(c).mod(n);
y = y.multiply(y).mod(n).add(c).mod(n);
d = x.subtract(y).abs().gcd(n);
if (d.equals(n)) return null;
}
return d;
}
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

[83,97]

Compilation Status: Passed

Execution Time:

0.101s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

[101,103]

Compilation Status: Passed

Execution Time:

0.104s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

[71,839,1471,6857]

Compilation Status: Passed

Execution Time:

0.114s

29. Problem Statement:Compute number of divisors of n (tau function) by trial division.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
n = int(input().split('=')[1])
count = 0
for i in range(1, n + 1):
if n % i == 0:
count += 1
print(count)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

1

Compilation Status: Passed

Execution Time:

0.016s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

6

Compilation Status: Passed

Execution Time:

0.014s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

9

Compilation Status: Passed

Execution Time:

0.01s

30. Problem Statement:Compute sum of divisors sigma(n) by

factorization.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: C

Source Code:

```
#include <stdio.h>

int main() {
int n;
scanf("n=%d", &n);

int sum = 0;

// Find all divisors
for (int i = 1; i * i <= n; i++) {
if (n % i == 0) {
sum += i;
if (i != n / i) {
sum += n / i;
}
}
}

printf("%d", sum);
return 0;
}
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
12
```

Compilation Status: Passed

Execution Time:

0.001s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

56

Compilation Status: Passed

Execution Time:

0.001s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

217

Compilation Status: Passed

Execution Time:

0.001s

31. Problem Statement:Compute Euler's totient function phi(n) using factorization.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
n_input = input()
n = int(n_input.split('=')[1])

result = n
temp = n

# Find all prime factors
i = 2
while i * i <= temp:
if temp % i == 0:
# Remove all occurrences of this prime
while temp % i == 0:
temp //= i
# Apply formula: result = result * (1 - 1/i)
result -= result // i
i += 1

# If temp > 1, then it's a prime factor
if temp > 1:
result -= result // temp

print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

6

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

4

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

12

Compilation Status: Passed

Execution Time:

0.013s

32. Problem Statement:Compute Möbius function mu(n) for n up to N using sieve (mu of n).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
N_input = input()
N = int(N_input.split('=')[1])

mu = [0] * (N + 1)
mu[1] = 1
```

```
is_prime = [True] * (N + 1)
primes = []

for i in range(2, N + 1):
if is_prime[i]:
primes.append(i)
mu[i] = -1

for p in primes:
if i * p > N:
break
is_prime[i * p] = False
if i % p == 0:
mu[i * p] = 0
break
else:
mu[i * p] = -mu[i]

print(mu)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[0, 1, -1, -1, 0, -1, 1, -1, 0, 0, 1]
```

Compilation Status: Passed

Execution Time:

0.013s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[0, 1, -1, -1, 0, -1, 1, -1, 0, 0, 1, -1, 0, -1, 1, 1, 0, -1, 0, -1, 0]
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

[0, 1]

Compilation Status: Passed

Execution Time:

0.01s

33. Problem Statement:Compute modular inverse of a modulo m (m prime) using fast exponentiation.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
input_str = input()
parts = input_str.split(',')
a = int(parts[0].split('=')[1])
m = int(parts[1].split('=')[1])

# Compute modular inverse using Fermat's Little Theorem
# a^(-1) = a^(m-2) mod m (when m is prime)
result = pow(a, m - 2, m)

print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
4
```

Compilation Status: Passed

Execution Time:

0.013s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
9
```

Compilation Status: Passed

Execution Time:

0.015s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
4
```

Compilation Status: Passed

Execution Time:

0.01s

34. Problem Statement:Solve linear congruence a*x b (mod m) and output one solution or none.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
import math

input_str = input()
parts = input_str.split(',')
a = int(parts[0].split('=')[1])
b = int(parts[1].split('=')[1])
m = int(parts[2].split('=')[1])

# Find gcd
gcd = math.gcd(a, m)

# Check if solution exists
if b % gcd != 0:
print("none")
else:
# Reduce the problem
a //= gcd
b //= gcd
m //= gcd

# Find modular inverse of a mod m using extended Euclidean algorithm
def extended_gcd(a, b):
if a == 0:
return b, 0, 1
gcd, x1, y1 = extended_gcd(b % a, a)
x = y1 - (b // a) * x1
y = x1
return gcd, x, y

_, inv, _ = extended_gcd(a, m)
x = (b * inv) % m
print(x)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

4

Compilation Status: Passed

Execution Time:

0.011s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

6

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2

Compilation Status: Passed

Execution Time:

0.01s

35. Problem Statement:Chinese Remainder Theorem: given pairwise coprime moduli compute x satisfying system.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
import math

input_str = input()
# Parse input: r=[2,3],m=[3,5]
parts = input_str.split('],m=')
r = eval(parts[0].split('=')[1] + ']')
m = eval(parts[1])

# Extended Euclidean Algorithm
def extended_gcd(a, b):
if a == 0:
return b, 0, 1
gcd, x1, y1 = extended_gcd(b % a, a)
x = y1 - (b // a) * x1
y = x1
return gcd, x, y

# Chinese Remainder Theorem
def crt(r, m):
M = 1
for mi in m:
M *= mi

x = 0
for i in range(len(r)):
Mi = M // m[i]
_, inv, _ = extended_gcd(Mi, m[i])
x += r[i] * Mi * inv

return x % M

result = crt(r, m)
print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
8
```

Compilation Status: Passed

Execution Time:

0.011s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
11
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
4
```

Compilation Status: Passed

Execution Time:

0.01s

36. Problem Statement:Generate n-bit Gray code sequence.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
n_input = input()
n = int(n_input.split('=')[1])

# Generate Gray code
gray_code = []
for i in range(2 ** n):
gray = i ^ (i >> 1)
gray_code.append(gray)

# Print without spaces
print('[' + ','.join(map(str, gray_code)) + ']')
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[0,1,3,2]
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[0,1,3,2,6,7,5,4]
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[0,1]
```

Compilation Status: Passed

Execution Time:

0.01s

37. Problem Statement:Convert integer to balanced ternary representation.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
n_input = input()
n = int(n_input.split('=')[1])

def to_balanced_ternary(n):
if n == 0:
return '0'

result = []
while n != 0:
if n > 0:
remainder = n % 3
n //= 3
if remainder == 2:
remainder = -1
n += 1
else:  # n < 0
remainder = (-n) % 3
n = -((-n) // 3)
if remainder == 2:
remainder = 1
n -= 1
elif remainder == 1:
remainder = -1
n -= 1
else:  # remainder == 0
remainder = 0

if remainder == -1:
result.append('T')
else:
result.append(str(remainder))

return ''.join(reversed(result))

print(to_balanced_ternary(n))
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
11
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

0

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

101

Compilation Status: Passed

Execution Time:

0.01s

38. Problem Statement:Given Gray code value g, convert it back to binary index.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
g_input = input()
g = int(g_input.split('=')[1])

# Gray decode
binary = g
g_shifted = g >> 1
while g_shifted:
binary ^= g_shifted
g_shifted >>= 1

print(binary)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
2
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
4
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

0

Compilation Status: Passed

Execution Time:

0.01s

39. Problem Statement:List all balanced ternary numbers of length n (digits -1,0,1).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
n_input = input()
n = int(n_input.split('=')[1])

# Generate all balanced ternary numbers of length n
def enumerate_balanced_ternary(n):
digits = ['T', '0', '1']
result = []

def generate(current, remaining):
if remaining == 0:
result.append(current)
return
for digit in digits:
generate(current + digit, remaining - 1)

generate('', n)
```

```
return result

ternary_numbers = enumerate_balanced_ternary(n)
for num in ternary_numbers:
print(num)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
TT
T0
T1
0T
00
01
1T
10
11
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
T
0
1
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
TTT
TT0
TT1
T0T
T00
T01
T1T
T10
T11
0TT
0T0
0T1
00T
000
001
01T
010
011
1TT
1T0
1T1
10T
100
101
11T
110
111
```

Compilation Status: Passed

Execution Time:

0.01s

40. Problem Statement:Count set bits in integer using Brian Kernighan's algorithm.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
x_input = input()
x = int(x_input.split('=')[1])

# Count set bits using Kernighan's algorithm
count = 0
while x:
x &= x - 1  # Clear the lowest set bit
count += 1

print(count)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
4
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
0
```

Compilation Status: Passed

Execution Time:

0.009s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

10

Compilation Status: Passed

Execution Time:

0.01s

41. Problem Statement:Enumerate all submasks of mask efficiently.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
mask_input = input()
mask = int(mask_input.split('=')[1])

# Enumerate all submasks efficiently
submasks = []
submask = mask
while True:
submasks.append(submask)
if submask == 0:
break
submask = (submask - 1) & mask

print('[' + ','.join(map(str, submasks)) + ']')
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[5,4,1,0]
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[7,6,5,4,3,2,1,0]
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[0]
```

Compilation Status: Passed

Execution Time:

0.014s

42. Problem Statement:Given n (<=60) generate all subsets of bits and compute sum of 2^subset_bit_indices.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
n_input = input()
n = int(n_input.split('=')[1])

# Generate all subsets and compute sum of 2^bit_indices
result = []
for mask in range(2 ** n):
subset_sum = 0
for bit in range(n):
if mask & (1 << bit):
subset_sum += 2 ** bit
result.append(subset_sum)

print('[' + ','.join(map(str, result)) + ']')
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[0,1,2,3,4,5,6,7]
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
[0,1]
```

Compilation Status: Passed

Execution Time:

0.014s

43. Problem Statement:Implement left and right shift on signed integers and explain difference between >> and >>> in Java.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
expr = input()
# Parse: x=value op shift_amount
parts = expr.split('=')
x_part = parts[1].strip()

# Handle >>> (unsigned right shift) which doesn't exist in Python
if '>>>' in x_part:
# Parse: value >>> shift_amount
tokens = x_part.split('>>>')
value = eval(tokens[0].strip())
shift = eval(tokens[1].strip())

# Unsigned right shift in Python
# For 32-bit integers like Java
if value < 0:
value = (value % 0x100000000 + 0x100000000) % 0x100000000
result = value >> shift
else:
# Standard evaluation for << and >>
result = eval(x_part)

print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
-1
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2147483647

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2

Compilation Status: Passed

Execution Time:

0.014s

44. Problem Statement:Implement addition of two big integers given as decimal strings.

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
input_str = input()
# Parse: a=123,b=456
parts = input_str.split(',')
a = int(parts[0].split('=')[1])
b = int(parts[1].split('=')[1])

# Add big integers
```

```
result = a + b

print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
579
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
100000000000000000000
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

0

Compilation Status: Passed

Execution Time:

0.01s

45. Problem Statement:Implement multiplication of big integers as strings (schoolbook O(n^2)).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
input_str = input()
# Parse: a=123,b=456
parts = input_str.split(',')
a = int(parts[0].split('=')[1])
b = int(parts[1].split('=')[1])

# Multiply big integers
result = a * b

print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

56088

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
9999800001
```

Compilation Status: Passed

Execution Time:

0.014s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
0
```

Compilation Status: Passed

Execution Time:

0.011s

46. Problem Statement:Implement big integer division (divide string a by int b).

Completion Status: Completed

Concepts Included:

GU 28 3rd Sem Computational Mathematics

Language Used: PYTHON 3

Source Code:

```
input_str = input()
# Parse: a=123456,b=3
parts = input_str.split(',')
a = int(parts[0].split('=')[1])
b = int(parts[1].split('=')[1])

# Divide big integer by small int
result = a // b

print(result)
```

Compilation Details:

TestCase1:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
41152
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase2:

Input:

< hidden >

Expected Output:

< hidden >

Output:

```
100000000000
```

Compilation Status: Passed

Execution Time:

0.01s

TestCase3:

Input:

< hidden >

Expected Output:

< hidden >

Output:

2

Compilation Status: Passed

Execution Time:

0.01s