print
Solution to Project Euler problem 2

Computers are fast, so we can implement this solution directly without any clever math.

```
def compute():
	ans = 0
	x = 1  Represents the current Fibonacci number being processed
	y = 2  Represents the next Fibonacci number in the sequence
	while x <= 4000000:
		if x % 2 == 0:
			ans += x
		x, y = y, x + y
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 45

```
def compute():
	i = 286
	j = 166
	k = 144
	while True:
		triangle = i * (i + 1) // 2
		pentagon = j * (j * 3 - 1) // 2
		hexagon  = k * (k * 2 - 1)
		minimum = min(triangle, pentagon, hexagon)
		if minimum == max(triangle, pentagon, hexagon):
			return str(triangle)
		if minimum == triangle: i += 1
		if minimum == pentagon: j += 1
		if minimum == hexagon : k += 1


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 17

- For the numbers 0 to 19, we write the single word:

{zero, one, two, three, four, five, six, seven, eight, nine,
ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen}.
- For the numbers 20 to 99, we write the word for the tens place:
{twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety}.
Subsequently if the last digit is not 0, then we write the word for the ones place (one to nine).
- For the numbers 100 to 999, we write the ones word for the hundreds place followed by "hundred":
{one hundred, two hundred, three hundred, ..., eight hundred, nine hundred}.
Subsequently if the last two digits are not 00, then we write the word "and"
followed by the phrase for the last two digits (from 01 to 99).
- For the numbers 1000 to 999999, we write the word for the three digits starting at the
thousands place and going leftward, followed by "thousand". Subsequently if the last three
digits are not 000, then we write the phrase for the last three digits (from 001 to 999).

```
def compute():
	ans = sum(len(to_english(i)) for i in range(1, 1001))
	return str(ans)


def to_english(n):
	if 0 <= n < 20:
		return ONES[n]
	elif 20 <= n < 100:
		return TENS[n // 10] + (ONES[n % 10] if (n % 10 != 0) else "")
	elif 100 <= n < 1000:
		return ONES[n // 100] + "hundred" + (("and" + to_english(n % 100)) if (n % 100 != 0) else "")
	elif 1000 <= n < 1000000:
		return to_english(n // 1000) + "thousand" + (to_english(n % 1000) if (n % 1000 != 0) else "")
	else:
		raise ValueError()


ONES = ["zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine",
	"ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"]
TENS = ["", "", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"]


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 29

We generate all the possible powers in the given range, put each value
into a set, and let the set count the number of unique values present.

```
def compute():
	seen = set(a**b for a in range(2, 101) for b in range(2, 101))
	return str(len(seen))


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 27

```
import eulerlib, itertools


def compute():
	ans = max(((a, b) for a in range(-999, 1000) for b in range(2, 1000)),
		key=count_consecutive_primes)
	return str(ans[0] * ans[1])


def count_consecutive_primes(ab):
	a, b = ab
	for i in itertools.count():
		n = i * i + i * a + b
		if not is_prime(n):
			return i


isprimecache = eulerlib.list_primality(1000)

def is_prime(n):
	if n < 0:
		return False
	elif n < len(isprimecache):
		return isprimecache[n]
	else:
		return eulerlib.is_prime(n)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 21

We first compute a table of sum-of-proper-divisors, then we use it to test which numbers are amicable.
This approach differs from the Java implementation because trying to directly compute
the proper-divisor-sum of each number by brute force is unacceptably slow in Python.

```
def compute():
	Compute sum of proper divisors for each number
	divisorsum = [0] * 10000
	for i in range(1, len(divisorsum)):
		for j in range(i * 2, len(divisorsum), i):
			divisorsum[j] += i

	Find all amicable pairs within range
	ans = 0
	for i in range(1, len(divisorsum)):
		j = divisorsum[i]
		if j != i and j < len(divisorsum) and divisorsum[j] == i:
			ans += i
	return str(ans)
```

```
if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 25

```
import itertools


Because the target number is relatively small, we simply compute each Fibonacci number starting
from the beginning until we encounter one with exactly 1000 digits. The Fibonacci sequence grows
exponentially with a base of about 1.618, so the numbers in base 10 will lengthen by one digit
after every log10(1.618) ~= 4.78 steps on average. This means the answer is at index around 4780.
def compute():
	DIGITS = 1000
	prev = 1
	cur = 0
	for i in itertools.count():
		At this point, prev = fibonacci(i - 1) and cur = fibonacci(i)
		if len(str(cur)) > DIGITS:
			raise RuntimeError("Not found")
		elif len(str(cur)) == DIGITS:
			return str(i)

		Advance the Fibonacci sequence by one step
		prev, cur = cur, prev + cur


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 34

```
import math, sys
if sys.version_info.major == 2:
	range = xrange


def compute():
	As stated in the problem, 1 = 1! and 2 = 2! are excluded.
	If a number has at least n >= 8 digits, then even if every digit is 9,
	n * 9! is still less than the number (which is at least 10^n).
	ans = sum(i for i in range(3, 10000000) if i == factorial_digit_sum(i))
	return str(ans)


def factorial_digit_sum(n):
	result = 0
	while n >= 10000:
		result += FACTORIAL_DIGITS_SUM_WITH_LEADING_ZEROS[n % 10000]
```

```
            n //= 10000
        return result + FACTORIAL_DIGITS_SUM_WITHOUT_LEADING_ZEROS[n]

FACTORIAL_DIGITS_SUM_WITHOUT_LEADING_ZEROS = [sum(math.factorial(int(c)) for c in str(i)) for i in range(10000)]
FACTORIAL_DIGITS_SUM_WITH_LEADING_ZEROS = [sum(math.factorial(int(c)) for c in str(i).zfill(4)) for i in range(10000)]


if __name__ == "__main__":
    print(compute())
```

Solution to Project Euler problem 24

```
import itertools, sys


We initialize a list as the lowest permutation of the given digits, which is the sequence
(0,1,2,3,4,5,6,7,8,9). Then we call a Python library function that generates a stream of
all permutations of the values, seek to the 999 999th element (0-based indexing), and stringify it.
def compute():
    arr = list(range(10))
    temp = itertools.islice(itertools.permutations(arr), 999999, None)
    return "".join(str(x) for x in next(temp))


if __name__ == "__main__":
    print(compute())
```

Solution to Project Euler problem 47

```
import eulerlib, itertools, sys
if sys.version_info.major == 2:
    filter = itertools.ifilter


def compute():
    cond = lambda i: all((count_distinct_prime_factors(i + j) == 4) for j in range(4))
    ans = next(filter(cond, itertools.count()))
    return str(ans)


@eulerlib.memoize
def count_distinct_prime_factors(n):
    count = 0
    while n > 1:
        count += 1
        for i in range(2, eulerlib.sqrt(n) + 1):
            if n % i == 0:
                while True:
```

```
                                        n //= i
                                        if n % i != 0:
                                                break
                                break
                        else:
                                break  n is prime
        return count


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 14

```
import eulerlib, sys
if sys.version_info.major == 2:
        range = xrange


We compute the Collatz chain length for every integer in the range according to the iteration rule.
Also, we cache the Collatz value for all integer arguments to speed up the computation.
def compute():
        sys.setrecursionlimit(3000)
        ans = max(range(1, 1000000), key=collatz_chain_length)
        return str(ans)


@eulerlib.memoize
def collatz_chain_length(x):
        if x == 1:
                return 1
        if x % 2 == 0:
                y = x // 2
        else:
                y = x * 3 + 1
        return collatz_chain_length(y) + 1


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 31

```
We use the standard dynamic programming algorithm to solve the subset sum problem over integers.
The order of the coin values does not matter, but the values need to be unique.
def compute():
        TOTAL = 200

                At the start of each loop iteration, ways[i] is the number of ways to use {any copies
```

```
        of the all the coin values seen before this iteration} to form an unordered sum of i
        ways = [1] + [0] * TOTAL
        for coin in [1, 2, 5, 10, 20, 50, 100, 200]:
                for i in range(len(ways) - coin):
                        ways[i + coin] += ways[i]
        return str(ways[-1])


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 46

```
import eulerlib, itertools, sys
if sys.version_info.major == 2:
        filterfalse = itertools.ifilterfalse
else:
        filterfalse = itertools.filterfalse


def compute():
        ans = next(filterfalse(test_goldbach, itertools.count(9, 2)))
        return str(ans)


def test_goldbach(n):
        if n % 2 == 0 or eulerlib.is_prime(n):
                return True
        for i in itertools.count(1):
                k = n - 2 * i * i
                if k <= 0:
                        return False
                elif eulerlib.is_prime(k):
                        return True


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 19

```
import datetime


We simply use Python's built-in date library to compute the answer by brute force.
def compute():
        ans = sum(1
                for y in range(1901, 2001)
                for m in range(1, 13)
                if datetime.date(y, m, 1).weekday() == 6)
```

```
        return str(ans)


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 6

Computers are fast, so we can implement this solution directly without any clever math.
However for the mathematically inclined, there are closed-form formulas:
  s = N(N + 1) / 2.
  s2 = N(N + 1)(2N + 1) / 6.
Hence s^2 - s2 = (N^4 / 4) + (N^3 / 6) - (N^2 / 4) - (N / 6).

```
def compute():
        N = 100
        s = sum(i for i in range(1, N + 1))
        s2 = sum(i**2 for i in range(1, N + 1))
        return str(s**2 - s2)


if __name__ == "__main__":
        print(compute())
```

Open the file in read mode

```
Namelist = []

Totaltext = []

text1 = "print"


import os

List all files in a directory using os.listdir
basepath = '/home/amritpal/Downloads/Coding/Test'
for entry in os.listdir(basepath):
    if os.path.isfile(os.path.join(basepath, entry)):
        Namelist.append(entry)
    #print(Namelist)

for n in range (0, len(Namelist)):
    '''print(n)
    print(Namelist[n])'''
    nameoffile = Namelist[n]
    text = open( nameoffile ,  "r")

    for line in text:
        Totaltext.append(line)
        #print(line)
        text1 += line

print(text1)
```

```
file1 = open("text.txt" , "w+")
\n is placed to indicate EOL (End of Line)
file1.write(text1)
file1.close()
```

Solution to Project Euler problem 28

From the diagram, let's observe the four corners of an n * n square (where n is odd).
It's not hard to convince yourself that the top right corner always has the value n^2.
Working counterclockwise (backwards), the top left corner has the value n^2 - (n - 1),
the bottom left corner has the value n^2 - 2(n - 1), and the bottom right is n^2 - 3(n - 1).
Putting it all together, this outermost ring contributes 4n^2 - 6(n - 1) to the final sum.

Incidentally, the closed form of this sum is (4m^3 + 3m^2 + 8m - 9) / 6, where m = size.

```
def compute():
	SIZE = 1001  Must be odd
	ans = 1  Special case for size 1
	ans += sum(4 * i * i - 6 * (i - 1) for i in range(3, SIZE + 1, 2))
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 39

```
def compute():
	ans = max(range(1, 1001), key=count_solutions)
	return str(ans)


def count_solutions(p):
	result = 0
	for a in range(1, p + 1):
		for b in range(a, (p - a) // 2 + 1):
			c = p - a - b  c >= b
			if a * a + b * b == c * c:
				result += 1
	return result


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 20

```
import math


We do a straightforward computation thanks to Python's built-in arbitrary precision integer type.
def compute():
	n = math.factorial(100)
	ans = sum(int(c) for c in str(n))
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 10

```
import eulerlib


Call the sieve of Eratosthenes and sum the primes found.
def compute():
	ans = sum(eulerlib.list_primes(1999999))
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 8

```
We implement a straightforward algorithm that examines every substring of length 13.
def compute():
	ans = max(digit_product(NUMBER[i : i + ADJACENT]) for i in range(len(NUMBER) - ADJACENT + 1))
	return str(ans)


def digit_product(s):
	result = 1
	for c in s:
		result *= int(c)
	return result


NUMBER =
"73167176531330624919225119674426574742355349194934969835203127745063262395783180169848018694788
```

```
85184385861560789112949495459501737958331952853208805511125406987471585238630507156932909632952
27443043557668966489504452445231617318564030987111217223831136222989342338030813533627661428280
64444866452387493035890729629049156044077239071381051585930796086670172427121883998797908792274
92190169972088809377665727333001053367881220235421809751254540594752243525849077116705560136048
39586446706324415722155397536978179778461740649551492908625693219784686224828397224137565705605
74902614079729686524145351004748216637048440319989000889524345065854122758866688116427171479924
44292823086346567481391912316282458617866458359124566529476545682848912883142607690042242190226
71055626321111109370544217506941658960408071984038509624554443629812309878799272442849091888458
01561660979191338754992005240636899125607176060588611646710940507754100225698315520005593572972
57163626956188267042825248360082325753042075296345 0"
ADJACENT = 13


if __name__ == "__main__":
    print(compute())
```

Solution to Project Euler problem 36

```
import sys
if sys.version_info.major == 2:
    range = xrange


def compute():
    ans = sum(i for i in range(1000000) if is_decimal_binary_palindrome(i))
    return str(ans)


def is_decimal_binary_palindrome(n):
    s = str(n)
    if s != s[ : : -1]:
        return False
    t = bin(n)[2 : ]
    return t == t[ : : -1]


if __name__ == "__main__":
    print(compute())
```

Solution to Project Euler problem 22

We apply straightforward algorithms to sort the names, sum the letter values, and multiply by the position.

```
def compute():
    ans = sum((i + 1) * (ord(c) - ord('A') + 1)
        for (i, name) in enumerate(sorted(NAMES))
        for c in name)
    return str(ans)
```

NAMES = [10 strings per line, except final line
	"MARY", "PATRICIA", "LINDA", "BARBARA", "ELIZABETH", "JENNIFER", "MARIA", "SUSAN", "MARGARET", "DOROTHY",
	"LISA", "NANCY", "KAREN", "BETTY", "HELEN", "SANDRA", "DONNA", "CAROL", "RUTH", "SHARON",
	"MICHELLE", "LAURA", "SARAH", "KIMBERLY", "DEBORAH", "JESSICA", "SHIRLEY", "CYNTHIA", "ANGELA", "MELISSA",
	"BRENDA", "AMY", "ANNA", "REBECCA", "VIRGINIA", "KATHLEEN", "PAMELA", "MARTHA", "DEBRA", "AMANDA",
	"STEPHANIE", "CAROLYN", "CHRISTINE", "MARIE", "JANET", "CATHERINE", "FRANCES", "ANN", "JOYCE", "DIANE",
	"ALICE", "JULIE", "HEATHER", "TERESA", "DORIS", "GLORIA", "EVELYN", "JEAN", "CHERYL", "MILDRED",
	"KATHERINE", "JOAN", "ASHLEY", "JUDITH", "ROSE", "JANICE", "KELLY", "NICOLE", "JUDY", "CHRISTINA",
	"KATHY", "THERESA", "BEVERLY", "DENISE", "TAMMY", "IRENE", "JANE", "LORI", "RACHEL", "MARILYN",
	"ANDREA", "KATHRYN", "LOUISE", "SARA", "ANNE", "JACQUELINE", "WANDA", "BONNIE", "JULIA", "RUBY",
	"LOIS", "TINA", "PHYLLIS", "NORMA", "PAULA", "DIANA", "ANNIE", "LILLIAN", "EMILY", "ROBIN",
	"PEGGY", "CRYSTAL", "GLADYS", "RITA", "DAWN", "CONNIE", "FLORENCE", "TRACY", "EDNA", "TIFFANY",
	"CARMEN", "ROSA", "CINDY", "GRACE", "WENDY", "VICTORIA", "EDITH", "KIM", "SHERRY", "SYLVIA",
	"JOSEPHINE", "THELMA", "SHANNON", "SHEILA", "ETHEL", "ELLEN", "ELAINE", "MARJORIE", "CARRIE", "CHARLOTTE",
	"MONICA", "ESTHER", "PAULINE", "EMMA", "JUANITA", "ANITA", "RHONDA", "HAZEL", "AMBER", "EVA",
	"DEBBIE", "APRIL", "LESLIE", "CLARA", "LUCILLE", "JAMIE", "JOANNE", "ELEANOR", "VALERIE", "DANIELLE",
	"MEGAN", "ALICIA", "SUZANNE", "MICHELE", "GAIL", "BERTHA", "DARLENE", "VERONICA", "JILL", "ERIN",
	"GERALDINE", "LAUREN", "CATHY", "JOANN", "LORRAINE", "LYNN", "SALLY", "REGINA", "ERICA", "BEATRICE",
	"DOLORES", "BERNICE", "AUDREY", "YVONNE", "ANNETTE", "JUNE", "SAMANTHA", "MARION", "DANA", "STACY",
	"ANA", "RENEE", "IDA", "VIVIAN", "ROBERTA", "HOLLY", "BRITTANY", "MELANIE", "LORETTA", "YOLANDA",
	"JEANETTE", "LAURIE", "KATIE", "KRISTEN", "VANESSA", "ALMA", "SUE", "ELSIE", "BETH", "JEANNE",
	"VICKI", "CARLA", "TARA", "ROSEMARY", "EILEEN", "TERRI", "GERTRUDE", "LUCY", "TONYA", "ELLA",
	"STACEY", "WILMA", "GINA", "KRISTIN", "JESSIE", "NATALIE", "AGNES", "VERA", "WILLIE", "CHARLENE",
	"BESSIE", "DELORES", "MELINDA", "PEARL", "ARLENE", "MAUREEN", "COLLEEN", "ALLISON", "TAMARA", "JOY",
	"GEORGIA", "CONSTANCE", "LILLIE", "CLAUDIA", "JACKIE", "MARCIA", "TANYA", "NELLIE", "MINNIE", "MARLENE",
	"HEIDI", "GLENDA", "LYDIA", "VIOLA", "COURTNEY", "MARIAN", "STELLA", "CAROLINE", "DORA", "JO",
	"VICKIE", "MATTIE", "TERRY", "MAXINE", "IRMA", "MABEL", "MARSHA", "MYRTLE", "LENA", "CHRISTY",
	"DEANNA", "PATSY", "HILDA", "GWENDOLYN", "JENNIE", "NORA", "MARGIE", "NINA", "CASSANDRA", "LEAH",
	"PENNY", "KAY", "PRISCILLA", "NAOMI", "CAROLE", "BRANDY", "OLGA", "BILLIE", "DIANNE", "TRACEY",
	"LEONA", "JENNY", "FELICIA", "SONIA", "MIRIAM", "VELMA", "BECKY", "BOBBIE", "VIOLET", "KRISTINA",

"TONI", "MISTY", "MAE", "SHELLY", "DAISY", "RAMONA", "SHERRI", "ERIKA", "KATRINA", "CLAIRE",
"LINDSEY", "LINDSAY", "GENEVA", "GUADALUPE", "BELINDA", "MARGARITA", "SHERYL", "CORA", "FAYE", "ADA",
"NATASHA", "SABRINA", "ISABEL", "MARGUERITE", "HATTIE", "HARRIET", "MOLLY", "CECILIA", "KRISTI", "BRANDI",
"BLANCHE", "SANDY", "ROSIE", "JOANNA", "IRIS", "EUNICE", "ANGIE", "INEZ", "LYNDA", "MADELINE",
"AMELIA", "ALBERTA", "GENEVIEVE", "MONIQUE", "JODI", "JANIE", "MAGGIE", "KAYLA", "SONYA", "JAN",
"LEE", "KRISTINE", "CANDACE", "FANNIE", "MARYANN", "OPAL", "ALISON", "YVETTE", "MELODY", "LUZ",
"SUSIE", "OLIVIA", "FLORA", "SHELLEY", "KRISTY", "MAMIE", "LULA", "LOLA", "VERNA", "BEULAH",
"ANTOINETTE", "CANDICE", "JUANA", "JEANNETTE", "PAM", "KELLI", "HANNAH", "WHITNEY", "BRIDGET", "KARLA",
"CELIA", "LATOYA", "PATTY", "SHELIA", "GAYLE", "DELLA", "VICKY", "LYNNE", "SHERI", "MARIANNE",
"KARA", "JACQUELYN", "ERMA", "BLANCA", "MYRA", "LETICIA", "PAT", "KRISTA", "ROXANNE", "ANGELICA",
"JOHNNIE", "ROBYN", "FRANCIS", "ADRIENNE", "ROSALIE", "ALEXANDRA", "BROOKE", "BETHANY", "SADIE", "BERNADETTE",
"TRACI", "JODY", "KENDRA", "JASMINE", "NICHOLE", "RACHAEL", "CHELSEA", "MABLE", "ERNESTINE", "MURIEL",
"MARCELLA", "ELENA", "KRYSTAL", "ANGELINA", "NADINE", "KARI", "ESTELLE", "DIANNA", "PAULETTE", "LORA",
"MONA", "DOREEN", "ROSEMARIE", "ANGEL", "DESIREE", "ANTONIA", "HOPE", "GINGER", "JANIS", "BETSY",
"CHRISTIE", "FREDA", "MERCEDES", "MEREDITH", "LYNETTE", "TERI", "CRISTINA", "EULA", "LEIGH", "MEGHAN",
"SOPHIA", "ELOISE", "ROCHELLE", "GRETCHEN", "CECELIA", "RAQUEL", "HENRIETTA", "ALYSSA", "JANA", "KELLEY",
"GWEN", "KERRY", "JENNA", "TRICIA", "LAVERNE", "OLIVE", "ALEXIS", "TASHA", "SILVIA", "ELVIRA",
"CASEY", "DELIA", "SOPHIE", "KATE", "PATTI", "LORENA", "KELLIE", "SONJA", "LILA", "LANA",
"DARLA", "MAY", "MINDY", "ESSIE", "MANDY", "LORENE", "ELSA", "JOSEFINA", "JEANNIE", "MIRANDA",
"DIXIE", "LUCIA", "MARTA", "FAITH", "LELA", "JOHANNA", "SHARI", "CAMILLE", "TAMI", "SHAWNA",
"ELISA", "EBONY", "MELBA", "ORA", "NETTIE", "TABITHA", "OLLIE", "JAIME", "WINIFRED", "KRISTIE",
"MARINA", "ALISHA", "AIMEE", "RENA", "MYRNA", "MARLA", "TAMMIE", "LATASHA", "BONITA", "PATRICE",
"RONDA", "SHERRIE", "ADDIE", "FRANCINE", "DELORIS", "STACIE", "ADRIANA", "CHERI", "SHELBY", "ABIGAIL",
"CELESTE", "JEWEL", "CARA", "ADELE", "REBEKAH", "LUCINDA", "DORTHY", "CHRIS", "EFFIE", "TRINA",
"REBA", "SHAWN", "SALLIE", "AURORA", "LENORA", "ETTA", "LOTTIE", "KERRI", "TRISHA", "NIKKI",
"ESTELLA", "FRANCISCA", "JOSIE", "TRACIE", "MARISSA", "KARIN", "BRITTNEY", "JANELLE", "LOURDES", "LAUREL",
"HELENE", "FERN", "ELVA", "CORINNE", "KELSEY", "INA", "BETTIE", "ELISABETH", "AIDA", "CAITLIN",
"INGRID", "IVA", "EUGENIA", "CHRISTA", "GOLDIE", "CASSIE", "MAUDE", "JENIFER", "THERESE", "FRANKIE",
"DENA", "LORNA", "JANETTE", "LATONYA", "CANDY", "MORGAN", "CONSUELO", "TAMIKA", "ROSETTA", "DEBORA",
"CHERIE", "POLLY", "DINA", "JEWELL", "FAY", "JILLIAN", "DOROTHEA", "NELL", "TRUDY", "ESPERANZA",

"PATRICA", "KIMBERLEY", "SHANNA", "HELENA", "CAROLINA", "CLEO", "STEFANIE", "ROSARIO", "OLA", "JANINE",
"MOLLIE", "LUPE", "ALISA", "LOU", "MARIBEL", "SUSANNE", "BETTE", "SUSANA", "ELISE", "CECILE",
"ISABELLE", "LESLEY", "JOCELYN", "PAIGE", "JONI", "RACHELLE", "LEOLA", "DAPHNE", "ALTA", "ESTER",
"PETRA", "GRACIELA", "IMOGENE", "JOLENE", "KEISHA", "LACEY", "GLENNA", "GABRIELA", "KERI", "URSULA",
"LIZZIE", "KIRSTEN", "SHANA", "ADELINE", "MAYRA", "JAYNE", "JACLYN", "GRACIE", "SONDRA", "CARMELA",
"MARISA", "ROSALIND", "CHARITY", "TONIA", "BEATRIZ", "MARISOL", "CLARICE", "JEANINE", "SHEENA", "ANGELINE",
"FRIEDA", "LILY", "ROBBIE", "SHAUNA", "MILLIE", "CLAUDETTE", "CATHLEEN", "ANGELIA", "GABRIELLE", "AUTUMN",
"KATHARINE", "SUMMER", "JODIE", "STACI", "LEA", "CHRISTI", "JIMMIE", "JUSTINE", "ELMA", "LUELLA",
"MARGRET", "DOMINIQUE", "SOCORRO", "RENE", "MARTINA", "MARGO", "MAVIS", "CALLIE", "BOBBI", "MARITZA",
"LUCILE", "LEANNE", "JEANNINE", "DEANA", "AILEEN", "LORIE", "LADONNA", "WILLA", "MANUELA", "GALE",
"SELMA", "DOLLY", "SYBIL", "ABBY", "LARA", "DALE", "IVY", "DEE", "WINNIE", "MARCY",
"LUISA", "JERI", "MAGDALENA", "OFELIA", "MEAGAN", "AUDRA", "MATILDA", "LEILA", "CORNELIA", "BIANCA",
"SIMONE", "BETTYE", "RANDI", "VIRGIE", "LATISHA", "BARBRA", "GEORGINA", "ELIZA", "LEANN", "BRIDGETTE",
"RHODA", "HALEY", "ADELA", "NOLA", "BERNADINE", "FLOSSIE", "ILA", "GRETA", "RUTHIE", "NELDA",
"MINERVA", "LILLY", "TERRIE", "LETHA", "HILARY", "ESTELA", "VALARIE", "BRIANNA", "ROSALYN", "EARLINE",
"CATALINA", "AVA", "MIA", "CLARISSA", "LIDIA", "CORRINE", "ALEXANDRIA", "CONCEPCION", "TIA", "SHARRON",
"RAE", "DONA", "ERICKA", "JAMI", "ELNORA", "CHANDRA", "LENORE", "NEVA", "MARYLOU", "MELISA",
"TABATHA", "SERENA", "AVIS", "ALLIE", "SOFIA", "JEANIE", "ODESSA", "NANNIE", "HARRIETT", "LORAINE",
"PENELOPE", "MILAGROS", "EMILIA", "BENITA", "ALLYSON", "ASHLEE", "TANIA", "TOMMIE", "ESMERALDA", "KARINA",
"EVE", "PEARLIE", "ZELMA", "MALINDA", "NOREEN", "TAMEKA", "SAUNDRA", "HILLARY", "AMIE", "ALTHEA",
"ROSALINDA", "JORDAN", "LILIA", "ALANA", "GAY", "CLARE", "ALEJANDRA", "ELINOR", "MICHAEL", "LORRIE",
"JERRI", "DARCY", "EARNESTINE", "CARMELLA", "TAYLOR", "NOEMI", "MARCIE", "LIZA", "ANNABELLE", "LOUISA",
"EARLENE", "MALLORY", "CARLENE", "NITA", "SELENA", "TANISHA", "KATY", "JULIANNE", "JOHN", "LAKISHA",
"EDWINA", "MARICELA", "MARGERY", "KENYA", "DOLLIE", "ROXIE", "ROSLYN", "KATHRINE", "NANETTE", "CHARMAINE",
"LAVONNE", "ILENE", "KRIS", "TAMMI", "SUZETTE", "CORINE", "KAYE", "JERRY", "MERLE", "CHRYSTAL",
"LINA", "DEANNE", "LILIAN", "JULIANA", "ALINE", "LUANN", "KASEY", "MARYANNE", "EVANGELINE", "COLETTE",
"MELVA", "LAWANDA", "YESENIA", "NADIA", "MADGE", "KATHIE", "EDDIE", "OPHELIA", "VALERIA", "NONA",
"MITZI", "MARI", "GEORGETTE", "CLAUDINE", "FRAN", "ALISSA", "ROSEANN", "LAKEISHA", "SUSANNA", "REVA",
"DEIDRE", "CHASITY", "SHEREE", "CARLY", "JAMES", "ELVIA", "ALYCE", "DEIRDRE", "GENA", "BRIANA",
"ARACELI", "KATELYN", "ROSANNE", "WENDI", "TESSA", "BERTA", "MARVA", "IMELDA", "MARIETTA", "MARCI",

"LEONOR", "ARLINE", "SASHA", "MADELYN", "JANNA", "JULIETTE", "DEENA", "AURELIA", "JOSEFA", "AUGUSTA",
"LILIANA", "YOUNG", "CHRISTIAN", "LESSIE", "AMALIA", "SAVANNAH", "ANASTASIA", "VILMA", "NATALIA", "ROSELLA",
"LYNNETTE", "CORINA", "ALFREDA", "LEANNA", "CAREY", "AMPARO", "COLEEN", "TAMRA", "AISHA", "WILDA",
"KARYN", "CHERRY", "QUEEN", "MAURA", "MAI", "EVANGELINA", "ROSANNA", "HALLIE", "ERNA", "ENID",
"MARIANA", "LACY", "JULIET", "JACKLYN", "FREIDA", "MADELEINE", "MARA", "HESTER", "CATHRYN", "LELIA",
"CASANDRA", "BRIDGETT", "ANGELITA", "JANNIE", "DIONNE", "ANNMARIE", "KATINA", "BERYL", "PHOEBE", "MILLICENT",
"KATHERYN", "DIANN", "CARISSA", "MARYELLEN", "LIZ", "LAURI", "HELGA", "GILDA", "ADRIAN", "RHEA",
"MARQUITA", "HOLLIE", "TISHA", "TAMERA", "ANGELIQUE", "FRANCESCA", "BRITNEY", "KAITLIN", "LOLITA", "FLORINE",
"ROWENA", "REYNA", "TWILA", "FANNY", "JANELL", "INES", "CONCETTA", "BERTIE", "ALBA", "BRIGITTE",
"ALYSON", "VONDA", "PANSY", "ELBA", "NOELLE", "LETITIA", "KITTY", "DEANN", "BRANDIE", "LOUELLA",
"LETA", "FELECIA", "SHARLENE", "LESA", "BEVERLEY", "ROBERT", "ISABELLA", "HERMINIA", "TERRA", "CELINA",
"TORI", "OCTAVIA", "JADE", "DENICE", "GERMAINE", "SIERRA", "MICHELL", "CORTNEY", "NELLY", "DORETHA",
"SYDNEY", "DEIDRA", "MONIKA", "LASHONDA", "JUDI", "CHELSEY", "ANTIONETTE", "MARGOT", "BOBBY", "ADELAIDE",
"NAN", "LEEANN", "ELISHA", "DESSIE", "LIBBY", "KATHI", "GAYLA", "LATANYA", "MINA", "MELLISA",
"KIMBERLEE", "JASMIN", "RENAE", "ZELDA", "ELDA", "MA", "JUSTINA", "GUSSIE", "EMILIE", "CAMILLA",
"ABBIE", "ROCIO", "KAITLYN", "JESSE", "EDYTHE", "ASHLEIGH", "SELINA", "LAKESHA", "GERI", "ALLENE",
"PAMALA", "MICHAELA", "DAYNA", "CARYN", "ROSALIA", "SUN", "JACQULINE", "REBECA", "MARYBETH", "KRYSTLE",
"IOLA", "DOTTIE", "BENNIE", "BELLE", "AUBREY", "GRISELDA", "ERNESTINA", "ELIDA", "ADRIANNE", "DEMETRIA",
"DELMA", "CHONG", "JAQUELINE", "DESTINY", "ARLEEN", "VIRGINA", "RETHA", "FATIMA", "TILLIE", "ELEANORE",
"CARI", "TREVA", "BIRDIE", "WILHELMINA", "ROSALEE", "MAURINE", "LATRICE", "YONG", "JENA", "TARYN",
"ELIA", "DEBBY", "MAUDIE", "JEANNA", "DELILAH", "CATRINA", "SHONDA", "HORTENCIA", "THEODORA", "TERESITA",
"ROBBIN", "DANETTE", "MARYJANE", "FREDDIE", "DELPHINE", "BRIANNE", "NILDA", "DANNA", "CINDI", "BESS",
"IONA", "HANNA", "ARIEL", "WINONA", "VIDA", "ROSITA", "MARIANNA", "WILLIAM", "RACHEAL", "GUILLERMINA",
"ELOISA", "CELESTINE", "CAREN", "MALISSA", "LONA", "CHANTEL", "SHELLIE", "MARISELA", "LEORA", "AGATHA",
"SOLEDAD", "MIGDALIA", "IVETTE", "CHRISTEN", "ATHENA", "JANEL", "CHLOE", "VEDA", "PATTIE", "TESSIE",
"TERA", "MARILYNN", "LUCRETIA", "KARRIE", "DINAH", "DANIELA", "ALECIA", "ADELINA", "VERNICE", "SHIELA",
"PORTIA", "MERRY", "LASHAWN", "DEVON", "DARA", "TAWANA", "OMA", "VERDA", "CHRISTIN", "ALENE",
"ZELLA", "SANDI", "RAFAELA", "MAYA", "KIRA", "CANDIDA", "ALVINA", "SUZAN", "SHAYLA", "LYN",
"LETTIE", "ALVA", "SAMATHA", "ORALIA", "MATILDE", "MADONNA", "LARISSA", "VESTA", "RENITA", "INDIA",
"DELOIS", "SHANDA", "PHILLIS", "LORRI", "ERLINDA", "CRUZ", "CATHRINE", "BARB", "ZOE", "ISABELL",

"IONE", "GISELA", "CHARLIE", "VALENCIA", "ROXANNA", "MAYME", "KISHA", "ELLIE", "MELLISSA", "DORRIS",
"DALIA", "BELLA", "ANNETTA", "ZOILA", "RETA", "REINA", "LAURETTA", "KYLIE", "CHRISTAL", "PILAR",
"CHARLA", "ELISSA", "TIFFANI", "TANA", "PAULINA", "LEOTA", "BREANNA", "JAYME", "CARMEL", "VERNELL",
"TOMASA", "MANDI", "DOMINGA", "SANTA", "MELODIE", "LURA", "ALEXA", "TAMELA", "RYAN", "MIRNA",
"KERRIE", "VENUS", "NOEL", "FELICITA", "CRISTY", "CARMELITA", "BERNIECE", "ANNEMARIE", "TIARA", "ROSEANNE",
"MISSY", "CORI", "ROXANA", "PRICILLA", "KRISTAL", "JUNG", "ELYSE", "HAYDEE", "ALETHA", "BETTINA",
"MARGE", "GILLIAN", "FILOMENA", "CHARLES", "ZENAIDA", "HARRIETTE", "CARIDAD", "VADA", "UNA", "ARETHA",
"PEARLINE", "MARJORY", "MARCELA", "FLOR", "EVETTE", "ELOUISE", "ALINA", "TRINIDAD", "DAVID", "DAMARIS",
"CATHARINE", "CARROLL", "BELVA", "NAKIA", "MARLENA", "LUANNE", "LORINE", "KARON", "DORENE", "DANITA",
"BRENNA", "TATIANA", "SAMMIE", "LOUANN", "LOREN", "JULIANNA", "ANDRIA", "PHILOMENA", "LUCILA", "LEONORA",
"DOVIE", "ROMONA", "MIMI", "JACQUELIN", "GAYE", "TONJA", "MISTI", "JOE", "GENE", "CHASTITY",
"STACIA", "ROXANN", "MICAELA", "NIKITA", "MEI", "VELDA", "MARLYS", "JOHNNA", "AURA", "LAVERN",
"IVONNE", "HAYLEY", "NICKI", "MAJORIE", "HERLINDA", "GEORGE", "ALPHA", "YADIRA", "PERLA", "GREGORIA",
"DANIEL", "ANTONETTE", "SHELLI", "MOZELLE", "MARIAH", "JOELLE", "CORDELIA", "JOSETTE", "CHIQUITA", "TRISTA",
"LOUIS", "LAQUITA", "GEORGIANA", "CANDI", "SHANON", "LONNIE", "HILDEGARD", "CECIL", "VALENTINA", "STEPHANY",
"MAGDA", "KAROL", "GERRY", "GABRIELLA", "TIANA", "ROMA", "RICHELLE", "RAY", "PRINCESS", "OLETA",
"JACQUE", "IDELLA", "ALAINA", "SUZANNA", "JOVITA", "BLAIR", "TOSHA", "RAVEN", "NEREIDA", "MARLYN",
"KYLA", "JOSEPH", "DELFINA", "TENA", "STEPHENIE", "SABINA", "NATHALIE", "MARCELLE", "GERTIE", "DARLEEN",
"THEA", "SHARONDA", "SHANTEL", "BELEN", "VENESSA", "ROSALINA", "ONA", "GENOVEVA", "COREY", "CLEMENTINE",
"ROSALBA", "RENATE", "RENATA", "MI", "IVORY", "GEORGIANNA", "FLOY", "DORCAS", "ARIANA", "TYRA",
"THEDA", "MARIAM", "JULI", "JESICA", "DONNIE", "VIKKI", "VERLA", "ROSELYN", "MELVINA", "JANNETTE",
"GINNY", "DEBRAH", "CORRIE", "ASIA", "VIOLETA", "MYRTIS", "LATRICIA", "COLLETTE", "CHARLEEN", "ANISSA",
"VIVIANA", "TWYLA", "PRECIOUS", "NEDRA", "LATONIA", "LAN", "HELLEN", "FABIOLA", "ANNAMARIE", "ADELL",
"SHARYN", "CHANTAL", "NIKI", "MAUD", "LIZETTE", "LINDY", "KIA", "KESHA", "JEANA", "DANELLE",
"CHARLINE", "CHANEL", "CARROL", "VALORIE", "LIA", "DORTHA", "CRISTAL", "SUNNY", "LEONE", "LEILANI",
"GERRI", "DEBI", "ANDRA", "KESHIA", "IMA", "EULALIA", "EASTER", "DULCE", "NATIVIDAD", "LINNIE",
"KAMI", "GEORGIE", "CATINA", "BROOK", "ALDA", "WINNIFRED", "SHARLA", "RUTHANN", "MEAGHAN", "MAGDALENE",
"LISSETTE", "ADELAIDA", "VENITA", "TRENA", "SHIRLENE", "SHAMEKA", "ELIZEBETH", "DIAN", "SHANTA", "MICKEY",
"LATOSHA", "CARLOTTA", "WINDY", "SOON", "ROSINA", "MARIANN", "LEISA", "JONNIE", "DAWNA", "CATHIE",
"BILLY", "ASTRID", "SIDNEY", "LAUREEN", "JANEEN", "HOLLI", "FAWN", "VICKEY", "TERESSA", "SHANTE",

"RUBYE", "MARCELINA", "CHANDA", "CARY", "TERESE", "SCARLETT", "MARTY", "MARNIE", "LULU", "LISETTE",
"JENIFFER", "ELENOR", "DORINDA", "DONITA", "CARMAN", "BERNITA", "ALTAGRACIA", "ALETA", "ADRIANNA", "ZORAIDA",
"RONNIE", "NICOLA", "LYNDSEY", "KENDALL", "JANINA", "CHRISSY", "AMI", "STARLA", "PHYLIS", "PHUONG",
"KYRA", "CHARISSE", "BLANCH", "SANJUANITA", "RONA", "NANCI", "MARILEE", "MARANDA", "CORY", "BRIGETTE",
"SANJUANA", "MARITA", "KASSANDRA", "JOYCELYN", "IRA", "FELIPA", "CHELSIE", "BONNY", "MIREYA", "LORENZA",
"KYONG", "ILEANA", "CANDELARIA", "TONY", "TOBY", "SHERIE", "OK", "MARK", "LUCIE", "LEATRICE",
"LAKESHIA", "GERDA", "EDIE", "BAMBI", "MARYLIN", "LAVON", "HORTENSE", "GARNET", "EVIE", "TRESSA",
"SHAYNA", "LAVINA", "KYUNG", "JEANETTA", "SHERRILL", "SHARA", "PHYLISS", "MITTIE", "ANABEL", "ALESIA",
"THUY", "TAWANDA", "RICHARD", "JOANIE", "TIFFANIE", "LASHANDA", "KARISSA", "ENRIQUETA", "DARIA", "DANIELLA",
"CORINNA", "ALANNA", "ABBEY", "ROXANE", "ROSEANNA", "MAGNOLIA", "LIDA", "KYLE", "JOELLEN", "ERA",
"CORAL", "CARLEEN", "TRESA", "PEGGIE", "NOVELLA", "NILA", "MAYBELLE", "JENELLE", "CARINA", "NOVA",
"MELINA", "MARQUERITE", "MARGARETTE", "JOSEPHINA", "EVONNE", "DEVIN", "CINTHIA", "ALBINA", "TOYA", "TAWNYA",
"SHERITA", "SANTOS", "MYRIAM", "LIZABETH", "LISE", "KEELY", "JENNI", "GISELLE", "CHERYLE", "ARDITH",
"ARDIS", "ALESHA", "ADRIANE", "SHAINA", "LINNEA", "KAROLYN", "HONG", "FLORIDA", "FELISHA", "DORI",
"DARCI", "ARTIE", "ARMIDA", "ZOLA", "XIOMARA", "VERGIE", "SHAMIKA", "NENA", "NANNETTE", "MAXIE",
"LOVIE", "JEANE", "JAIMIE", "INGE", "FARRAH", "ELAINA", "CAITLYN", "STARR", "FELICITAS", "CHERLY",
"CARYL", "YOLONDA", "YASMIN", "TEENA", "PRUDENCE", "PENNIE", "NYDIA", "MACKENZIE", "ORPHA", "MARVEL",
"LIZBETH", "LAURETTE", "JERRIE", "HERMELINDA", "CAROLEE", "TIERRA", "MIRIAN", "META", "MELONY", "KORI",
"JENNETTE", "JAMILA", "ENA", "ANH", "YOSHIKO", "SUSANNAH", "SALINA", "RHIANNON", "JOLEEN", "CRISTINE",
"ASHTON", "ARACELY", "TOMEKA", "SHALONDA", "MARTI", "LACIE", "KALA", "JADA", "ILSE", "HAILEY",
"BRITTANI", "ZONA", "SYBLE", "SHERRYL", "RANDY", "NIDIA", "MARLO", "KANDICE", "KANDI", "DEB",
"DEAN", "AMERICA", "ALYCIA", "TOMMY", "RONNA", "NORENE", "MERCY", "JOSE", "INGEBORG", "GIOVANNA",
"GEMMA", "CHRISTEL", "AUDRY", "ZORA", "VITA", "VAN", "TRISH", "STEPHAINE", "SHIRLEE", "SHANIKA",
"MELONIE", "MAZIE", "JAZMIN", "INGA", "HOA", "HETTIE", "GERALYN", "FONDA", "ESTRELLA", "ADELLA",
"SU", "SARITA", "RINA", "MILISSA", "MARIBETH", "GOLDA", "EVON", "ETHELYN", "ENEDINA", "CHERISE",
"CHANA", "VELVA", "TAWANNA", "SADE", "MIRTA", "LI", "KARIE", "JACINTA", "ELNA", "DAVINA",
"CIERRA", "ASHLIE", "ALBERTHA", "TANESHA", "STEPHANI", "NELLE", "MINDI", "LU", "LORINDA", "LARUE",
"FLORENE", "DEMETRA", "DEDRA", "CIARA", "CHANTELLE", "ASHLY", "SUZY", "ROSALVA", "NOELIA", "LYDA",
"LEATHA", "KRYSTYNA", "KRISTAN", "KARRI", "DARLINE", "DARCIE", "CINDA", "CHEYENNE", "CHERRIE", "AWILDA",
"ALMEDA", "ROLANDA", "LANETTE", "JERILYN", "GISELE", "EVALYN", "CYNDI", "CLETA", "CARIN", "ZINA",

"ZENA", "VELIA", "TANIKA", "PAUL", "CHARISSA", "THOMAS", "TALIA", "MARGARETE", "LAVONDA", "KAYLEE",
"KATHLENE", "JONNA", "IRENA", "ILONA", "IDALIA", "CANDIS", "CANDANCE", "BRANDEE", "ANITRA", "ALIDA",
"SIGRID", "NICOLETTE", "MARYJO", "LINETTE", "HEDWIG", "CHRISTIANA", "CASSIDY", "ALEXIA", "TRESSIE", "MODESTA",
"LUPITA", "LITA", "GLADIS", "EVELIA", "DAVIDA", "CHERRI", "CECILY", "ASHELY", "ANNABEL", "AGUSTINA",
"WANITA", "SHIRLY", "ROSAURA", "HULDA", "EUN", "BAILEY", "YETTA", "VERONA", "THOMASINA", "SIBYL",
"SHANNAN", "MECHELLE", "LUE", "LEANDRA", "LANI", "KYLEE", "KANDY", "JOLYNN", "FERNE", "EBONI",
"CORENE", "ALYSIA", "ZULA", "NADA", "MOIRA", "LYNDSAY", "LORRETTA", "JUAN", "JAMMIE", "HORTENSIA",
"GAYNELL", "CAMERON", "ADRIA", "VINA", "VICENTA", "TANGELA", "STEPHINE", "NORINE", "NELLA", "LIANA",
"LESLEE", "KIMBERELY", "ILIANA", "GLORY", "FELICA", "EMOGENE", "ELFRIEDE", "EDEN", "EARTHA", "CARMA",
"BEA", "OCIE", "MARRY", "LENNIE", "KIARA", "JACALYN", "CARLOTA", "ARIELLE", "YU", "STAR",
"OTILIA", "KIRSTIN", "KACEY", "JOHNETTA", "JOEY", "JOETTA", "JERALDINE", "JAUNITA", "ELANA", "DORTHEA",
"CAMI", "AMADA", "ADELIA", "VERNITA", "TAMAR", "SIOBHAN", "RENEA", "RASHIDA", "OUIDA", "ODELL",
"NILSA", "MERYL", "KRISTYN", "JULIETA", "DANICA", "BREANNE", "AUREA", "ANGLEA", "SHERRON", "ODETTE",
"MALIA", "LORELEI", "LIN", "LEESA", "KENNA", "KATHLYN", "FIONA", "CHARLETTE", "SUZIE", "SHANTELL",
"SABRA", "RACQUEL", "MYONG", "MIRA", "MARTINE", "LUCIENNE", "LAVADA", "JULIANN", "JOHNIE", "ELVERA",
"DELPHIA", "CLAIR", "CHRISTIANE", "CHAROLETTE", "CARRI", "AUGUSTINE", "ASHA", "ANGELLA", "PAOLA", "NINFA",
"LEDA", "LAI", "EDA", "SUNSHINE", "STEFANI", "SHANELL", "PALMA", "MACHELLE", "LISSA", "KECIA",
"KATHRYNE", "KARLENE", "JULISSA", "JETTIE", "JENNIFFER", "HUI", "CORRINA", "CHRISTOPHER", "CAROLANN", "ALENA",
"TESS", "ROSARIA", "MYRTICE", "MARYLEE", "LIANE", "KENYATTA", "JUDIE", "JANEY", "IN", "ELMIRA",
"ELDORA", "DENNA", "CRISTI", "CATHI", "ZAIDA", "VONNIE", "VIVA", "VERNIE", "ROSALINE", "MARIELA",
"LUCIANA", "LESLI", "KARAN", "FELICE", "DENEEN", "ADINA", "WYNONA", "TARSHA", "SHERON", "SHASTA",
"SHANITA", "SHANI", "SHANDRA", "RANDA", "PINKIE", "PARIS", "NELIDA", "MARILOU", "LYLA", "LAURENE",
"LACI", "JOI", "JANENE", "DOROTHA", "DANIELE", "DANI", "CAROLYNN", "CARLYN", "BERENICE", "AYESHA",
"ANNELIESE", "ALETHEA", "THERSA", "TAMIKO", "RUFINA", "OLIVA", "MOZELL", "MARYLYN", "MADISON", "KRISTIAN",
"KATHYRN", "KASANDRA", "KANDACE", "JANAE", "GABRIEL", "DOMENICA", "DEBBRA", "DANNIELLE", "CHUN", "BUFFY",
"BARBIE", "ARCELIA", "AJA", "ZENOBIA", "SHAREN", "SHAREE", "PATRICK", "PAGE", "MY", "LAVINIA",
"KUM", "KACIE", "JACKELINE", "HUONG", "FELISA", "EMELIA", "ELEANORA", "CYTHIA", "CRISTIN", "CLYDE",
"CLARIBEL", "CARON", "ANASTACIA", "ZULMA", "ZANDRA", "YOKO", "TENISHA", "SUSANN", "SHERILYN", "SHAY",
"SHAWANDA", "SABINE", "ROMANA", "MATHILDA", "LINSEY", "KEIKO", "JOANA", "ISELA", "GRETTA", "GEORGETTA",
"EUGENIE", "DUSTY", "DESIRAE", "DELORA", "CORAZON", "ANTONINA", "ANIKA", "WILLENE", "TRACEE", "TAMATHA",

"REGAN", "NICHELLE", "MICKIE", "MAEGAN", "LUANA", "LANITA", "KELSIE", "EDELMIRA", "BREE", "AFTON",
"TEODORA", "TAMIE", "SHENA", "MEG", "LINH", "KELI", "KACI", "DANYELLE", "BRITT", "ARLETTE",
"ALBERTINE", "ADELLE", "TIFFINY", "STORMY", "SIMONA", "NUMBERS", "NICOLASA", "NICHOL", "NIA", "NAKISHA",
"MEE", "MAIRA", "LOREEN", "KIZZY", "JOHNNY", "JAY", "FALLON", "CHRISTENE", "BOBBYE", "ANTHONY",
"YING", "VINCENZA", "TANJA", "RUBIE", "RONI", "QUEENIE", "MARGARETT", "KIMBERLI", "IRMGARD", "IDELL",
"HILMA", "EVELINA", "ESTA", "EMILEE", "DENNISE", "DANIA", "CARL", "CARIE", "ANTONIO", "WAI",
"SANG", "RISA", "RIKKI", "PARTICIA", "MUI", "MASAKO", "MARIO", "LUVENIA", "LOREE", "LONI",
"LIEN", "KEVIN", "GIGI", "FLORENCIA", "DORIAN", "DENITA", "DALLAS", "CHI", "BILLYE", "ALEXANDER",
"TOMIKA", "SHARITA", "RANA", "NIKOLE", "NEOMA", "MARGARITE", "MADALYN", "LUCINA", "LAILA", "KALI",
"JENETTE", "GABRIELE", "EVELYNE", "ELENORA", "CLEMENTINA", "ALEJANDRINA", "ZULEMA", "VIOLETTE", "VANNESSA", "THRESA",
"RETTA", "PIA", "PATIENCE", "NOELLA", "NICKIE", "JONELL", "DELTA", "CHUNG", "CHAYA", "CAMELIA",
"BETHEL", "ANYA", "ANDREW", "THANH", "SUZANN", "SPRING", "SHU", "MILA", "LILLA", "LAVERNA",
"KEESHA", "KATTIE", "GIA", "GEORGENE", "EVELINE", "ESTELL", "ELIZBETH", "VIVIENNE", "VALLIE", "TRUDIE",
"STEPHANE", "MICHEL", "MAGALY", "MADIE", "KENYETTA", "KARREN", "JANETTA", "HERMINE", "HARMONY", "DRUCILLA",
"DEBBI", "CELESTINA", "CANDIE", "BRITNI", "BECKIE", "AMINA", "ZITA", "YUN", "YOLANDE", "VIVIEN",
"VERNETTA", "TRUDI", "SOMMER", "PEARLE", "PATRINA", "OSSIE", "NICOLLE", "LOYCE", "LETTY", "LARISA",
"KATHARINA", "JOSELYN", "JONELLE", "JENELL", "IESHA", "HEIDE", "FLORINDA", "FLORENTINA", "FLO", "ELODIA",
"DORINE", "BRUNILDA", "BRIGID", "ASHLI", "ARDELLA", "TWANA", "THU", "TARAH", "SUNG", "SHEA",
"SHAVON", "SHANE", "SERINA", "RAYNA", "RAMONITA", "NGA", "MARGURITE", "LUCRECIA", "KOURTNEY", "KATI",
"JESUS", "JESENIA", "DIAMOND", "CRISTA", "AYANA", "ALICA", "ALIA", "VINNIE", "SUELLEN", "ROMELIA",
"RACHELL", "PIPER", "OLYMPIA", "MICHIKO", "KATHALEEN", "JOLIE", "JESSI", "JANESSA", "HANA", "HA",
"ELEASE", "CARLETTA", "BRITANY", "SHONA", "SALOME", "ROSAMOND", "REGENA", "RAINA", "NGOC", "NELIA",
"LOUVENIA", "LESIA", "LATRINA", "LATICIA", "LARHONDA", "JINA", "JACKI", "HOLLIS", "HOLLEY", "EMMY",
"DEEANN", "CORETTA", "ARNETTA", "VELVET", "THALIA", "SHANICE", "NETA", "MIKKI", "MICKI", "LONNA",
"LEANA", "LASHUNDA", "KILEY", "JOYE", "JACQULYN", "IGNACIA", "HYUN", "HIROKO", "HENRY", "HENRIETTE",
"ELAYNE", "DELINDA", "DARNELL", "DAHLIA", "COREEN", "CONSUELA", "CONCHITA", "CELINE", "BABETTE", "AYANNA",
"ANETTE", "ALBERTINA", "SKYE", "SHAWNEE", "SHANEKA", "QUIANA", "PAMELIA", "MIN", "MERRI", "MERLENE",
"MARGIT", "KIESHA", "KIERA", "KAYLENE", "JODEE", "JENISE", "ERLENE", "EMMIE", "ELSE", "DARYL",
"DALILA", "DAISEY", "CODY", "CASIE", "BELIA", "BABARA", "VERSIE", "VANESA", "SHELBA", "SHAWNDA",
"SAM", "NORMAN", "NIKIA", "NAOMA", "MARNA", "MARGERET", "MADALINE", "LAWANA", "KINDRA", "JUTTA",

"JAZMINE", "JANETT", "HANNELORE", "GLENDORA", "GERTRUD", "GARNETT", "FREEDA", "FREDERICA", "FLORANCE", "FLAVIA",
"DENNIS", "CARLINE", "BEVERLEE", "ANJANETTE", "VALDA", "TRINITY", "TAMALA", "STEVIE", "SHONNA", "SHA",
"SARINA", "ONEIDA", "MICAH", "MERILYN", "MARLEEN", "LURLINE", "LENNA", "KATHERIN", "JIN", "JENI",
"HAE", "GRACIA", "GLADY", "FARAH", "ERIC", "ENOLA", "EMA", "DOMINQUE", "DEVONA", "DELANA",
"CECILA", "CAPRICE", "ALYSHA", "ALI", "ALETHIA", "VENA", "THERESIA", "TAWNY", "SONG", "SHAKIRA",
"SAMARA", "SACHIKO", "RACHELE", "PAMELLA", "NICKY", "MARNI", "MARIEL", "MAREN", "MALISA", "LIGIA",
"LERA", "LATORIA", "LARAE", "KIMBER", "KATHERN", "KAREY", "JENNEFER", "JANETH", "HALINA", "FREDIA",
"DELISA", "DEBROAH", "CIERA", "CHIN", "ANGELIKA", "ANDREE", "ALTHA", "YEN", "VIVAN", "TERRESA",
"TANNA", "SUK", "SUDIE", "SOO", "SIGNE", "SALENA", "RONNI", "REBBECCA", "MYRTIE", "MCKENZIE",
"MALIKA", "MAIDA", "LOAN", "LEONARDA", "KAYLEIGH", "FRANCE", "ETHYL", "ELLYN", "DAYLE", "CAMMIE",
"BRITTNI", "BIRGIT", "AVELINA", "ASUNCION", "ARIANNA", "AKIKO", "VENICE", "TYESHA", "TONIE", "TIESHA",
"TAKISHA", "STEFFANIE", "SINDY", "SANTANA", "MEGHANN", "MANDA", "MACIE", "LADY", "KELLYE", "KELLEE",
"JOSLYN", "JASON", "INGER", "INDIRA", "GLINDA", "GLENNIS", "FERNANDA", "FAUSTINA", "ENEIDA", "ELICIA",
"DOT", "DIGNA", "DELL", "ARLETTA", "ANDRE", "WILLIA", "TAMMARA", "TABETHA", "SHERRELL", "SARI",
"REFUGIO", "REBBECA", "PAULETTA", "NIEVES", "NATOSHA", "NAKITA", "MAMMIE", "KENISHA", "KAZUKO", "KASSIE",
"GARY", "EARLEAN", "DAPHINE", "CORLISS", "CLOTILDE", "CAROLYNE", "BERNETTA", "AUGUSTINA", "AUDREA", "ANNIS",
"ANNABELL", "YAN", "TENNILLE", "TAMICA", "SELENE", "SEAN", "ROSANA", "REGENIA", "QIANA", "MARKITA",
"MACY", "LEEANNE", "LAURINE", "KYM", "JESSENIA", "JANITA", "GEORGINE", "GENIE", "EMIKO", "ELVIE",
"DEANDRA", "DAGMAR", "CORIE", "COLLEN", "CHERISH", "ROMAINE", "PORSHA", "PEARLENE", "MICHELINE", "MERNA",
"MARGORIE", "MARGARETTA", "LORE", "KENNETH", "JENINE", "HERMINA", "FREDERICKA", "ELKE", "DRUSILLA", "DORATHY",
"DIONE", "DESIRE", "CELENA", "BRIGIDA", "ANGELES", "ALLEGRA", "THEO", "TAMEKIA", "SYNTHIA", "STEPHEN",
"SOOK", "SLYVIA", "ROSANN", "REATHA", "RAYE", "MARQUETTA", "MARGART", "LING", "LAYLA", "KYMBERLY",
"KIANA", "KAYLEEN", "KATLYN", "KARMEN", "JOELLA", "IRINA", "EMELDA", "ELENI", "DETRA", "CLEMMIE",
"CHERYLL", "CHANTELL", "CATHEY", "ARNITA", "ARLA", "ANGLE", "ANGELIC", "ALYSE", "ZOFIA", "THOMASINE",
"TENNIE", "SON", "SHERLY", "SHERLEY", "SHARYL", "REMEDIOS", "PETRINA", "NICKOLE", "MYUNG", "MYRLE",
"MOZELLA", "LOUANNE", "LISHA", "LATIA", "LANE", "KRYSTA", "JULIENNE", "JOEL", "JEANENE", "JACQUALINE",
"ISAURA", "GWENDA", "EARLEEN", "DONALD", "CLEOPATRA", "CARLIE", "AUDIE", "ANTONIETTA", "ALISE", "ALEX",
"VERDELL", "VAL", "TYLER", "TOMOKO", "THAO", "TALISHA", "STEVEN", "SO", "SHEMIKA", "SHAUN",
"SCARLET", "SAVANNA", "SANTINA", "ROSIA", "RAEANN", "ODILIA", "NANA", "MINNA", "MAGAN", "LYNELLE",
"LE", "KARMA", "JOEANN", "IVANA", "INELL", "ILANA", "HYE", "HONEY", "HEE", "GUDRUN",

"FRANK", "DREAMA", "CRISSY", "CHANTE", "CARMELINA", "ARVILLA", "ARTHUR", "ANNAMAE", "ALVERA", "ALEIDA",
"AARON", "YEE", "YANIRA", "VANDA", "TIANNA", "TAM", "STEFANIA", "SHIRA", "PERRY", "NICOL",
"NANCIE", "MONSERRATE", "MINH", "MELYNDA", "MELANY", "MATTHEW", "LOVELLA", "LAURE", "KIRBY", "KACY",
"JACQUELYNN", "HYON", "GERTHA", "FRANCISCO", "ELIANA", "CHRISTENA", "CHRISTEEN", "CHARISE", "CATERINA", "CARLEY",
"CANDYCE", "ARLENA", "AMMIE", "YANG", "WILLETTE", "VANITA", "TUYET", "TINY", "SYREETA", "SILVA",
"SCOTT", "RONALD", "PENNEY", "NYLA", "MICHAL", "MAURICE", "MARYAM", "MARYA", "MAGEN", "LUDIE",
"LOMA", "LIVIA", "LANELL", "KIMBERLIE", "JULEE", "DONETTA", "DIEDRA", "DENISHA", "DEANE", "DAWNE",
"CLARINE", "CHERRYL", "BRONWYN", "BRANDON", "ALLA", "VALERY", "TONDA", "SUEANN", "SORAYA", "SHOSHANA",
"SHELA", "SHARLEEN", "SHANELLE", "NERISSA", "MICHEAL", "MERIDITH", "MELLIE", "MAYE", "MAPLE", "MAGARET",
"LUIS", "LILI", "LEONILA", "LEONIE", "LEEANNA", "LAVONIA", "LAVERA", "KRISTEL", "KATHEY", "KATHE",
"JUSTIN", "JULIAN", "JIMMY", "JANN", "ILDA", "HILDRED", "HILDEGARDE", "GENIA", "FUMIKO", "EVELIN",
"ERMELINDA", "ELLY", "DUNG", "DOLORIS", "DIONNA", "DANAE", "BERNEICE", "ANNICE", "ALIX", "VERENA",
"VERDIE", "TRISTAN", "SHAWNNA", "SHAWANA", "SHAUNNA", "ROZELLA", "RANDEE", "RANAE", "MILAGRO", "LYNELL",
"LUISE", "LOUIE", "LOIDA", "LISBETH", "KARLEEN", "JUNITA", "JONA", "ISIS", "HYACINTH", "HEDY",
"GWENN", "ETHELENE", "ERLINE", "EDWARD", "DONYA", "DOMONIQUE", "DELICIA", "DANNETTE", "CICELY", "BRANDA",
"BLYTHE", "BETHANN", "ASHLYN", "ANNALEE", "ALLINE", "YUKO", "VELLA", "TRANG", "TOWANDA", "TESHA",
"SHERLYN", "NARCISA", "MIGUELINA", "MERI", "MAYBELL", "MARLANA", "MARGUERITA", "MADLYN", "LUNA", "LORY",
"LORIANN", "LIBERTY", "LEONORE", "LEIGHANN", "LAURICE", "LATESHA", "LARONDA", "KATRICE", "KASIE", "KARL",
"KALEY", "JADWIGA", "GLENNIE", "GEARLDINE", "FRANCINA", "EPIFANIA", "DYAN", "DORIE", "DIEDRE", "DENESE",
"DEMETRICE", "DELENA", "DARBY", "CRISTIE", "CLEORA", "CATARINA", "CARISA", "BERNIE", "BARBERA", "ALMETA",
"TRULA", "TEREASA", "SOLANGE", "SHEILAH", "SHAVONNE", "SANORA", "ROCHELL", "MATHILDE", "MARGARETA", "MAIA",
"LYNSEY", "LAWANNA", "LAUNA", "KENA", "KEENA", "KATIA", "JAMEY", "GLYNDA", "GAYLENE", "ELVINA",
"ELANOR", "DANUTA", "DANIKA", "CRISTEN", "CORDIE", "COLETTA", "CLARITA", "CARMON", "BRYNN", "AZUCENA",
"AUNDREA", "ANGELE", "YI", "WALTER", "VERLIE", "VERLENE", "TAMESHA", "SILVANA", "SEBRINA", "SAMIRA",
"REDA", "RAYLENE", "PENNI", "PANDORA", "NORAH", "NOMA", "MIREILLE", "MELISSIA", "MARYALICE", "LARAINE",
"KIMBERY", "KARYL", "KARINE", "KAM", "JOLANDA", "JOHANA", "JESUSA", "JALEESA", "JAE", "JACQUELYNE",
"IRISH", "ILUMINADA", "HILARIA", "HANH", "GENNIE", "FRANCIE", "FLORETTA", "EXIE", "EDDA", "DREMA",
"DELPHA", "BEV", "BARBAR", "ASSUNTA", "ARDELL", "ANNALISA", "ALISIA", "YUKIKO", "YOLANDO", "WONDA",
"WEI", "WALTRAUD", "VETA", "TEQUILA", "TEMEKA", "TAMEIKA", "SHIRLEEN", "SHENITA", "PIEDAD", "OZELLA",
"MIRTHA", "MARILU", "KIMIKO", "JULIANE", "JENICE", "JEN", "JANAY", "JACQUILINE", "HILDE", "FE",

"FAE", "EVAN", "EUGENE", "ELOIS", "ECHO", "DEVORAH", "CHAU", "BRINDA", "BETSEY", "ARMINDA",
"ARACELIS", "APRYL", "ANNETT", "ALISHIA", "VEOLA", "USHA", "TOSHIKO", "THEOLA", "TASHIA", "TALITHA",
"SHERY", "RUDY", "RENETTA", "REIKO", "RASHEEDA", "OMEGA", "OBDULIA", "MIKA", "MELAINE", "MEGGAN",
"MARTIN", "MARLEN", "MARGET", "MARCELINE", "MANA", "MAGDALEN", "LIBRADA", "LEZLIE", "LEXIE", "LATASHIA",
"LASANDRA", "KELLE", "ISIDRA", "ISA", "INOCENCIA", "GWYN", "FRANCOISE", "ERMINIA", "ERINN", "DIMPLE",
"DEVORA", "CRISELDA", "ARMANDA", "ARIE", "ARIANE", "ANGELO", "ANGELENA", "ALLEN", "ALIZA", "ADRIENE",
"ADALINE", "XOCHITL", "TWANNA", "TRAN", "TOMIKO", "TAMISHA", "TAISHA", "SUSY", "SIU", "RUTHA",
"ROXY", "RHONA", "RAYMOND", "OTHA", "NORIKO", "NATASHIA", "MERRIE", "MELVIN", "MARINDA", "MARIKO",
"MARGERT", "LORIS", "LIZZETTE", "LEISHA", "KAILA", "KA", "JOANNIE", "JERRICA", "JENE", "JANNET",
"JANEE", "JACINDA", "HERTA", "ELENORE", "DORETTA", "DELAINE", "DANIELL", "CLAUDIE", "CHINA", "BRITTA",
"APOLONIA", "AMBERLY", "ALEASE", "YURI", "YUK", "WEN", "WANETA", "UTE", "TOMI", "SHARRI",
"SANDIE", "ROSELLE", "REYNALDA", "RAGUEL", "PHYLICIA", "PATRIA", "OLIMPIA", "ODELIA", "MITZIE", "MITCHELL",
"MISS", "MINDA", "MIGNON", "MICA", "MENDY", "MARIVEL", "MAILE", "LYNETTA", "LAVETTE", "LAURYN",
"LATRISHA", "LAKIESHA", "KIERSTEN", "KARY", "JOSPHINE", "JOLYN", "JETTA", "JANISE", "JACQUIE", "IVELISSE",
"GLYNIS", "GIANNA", "GAYNELLE", "EMERALD", "DEMETRIUS", "DANYELL", "DANILLE", "DACIA", "CORALEE", "CHER",
"CEOLA", "BRETT", "BELL", "ARIANNE", "ALESHIA", "YUNG", "WILLIEMAE", "TROY", "TRINH", "THORA",
"TAI", "SVETLANA", "SHERIKA", "SHEMEKA", "SHAUNDA", "ROSELINE", "RICKI", "MELDA", "MALLIE", "LAVONNA",
"LATINA", "LARRY", "LAQUANDA", "LALA", "LACHELLE", "KLARA", "KANDIS", "JOHNA", "JEANMARIE", "JAYE",
"HANG", "GRAYCE", "GERTUDE", "EMERITA", "EBONIE", "CLORINDA", "CHING", "CHERY", "CAROLA", "BREANN",
"BLOSSOM", "BERNARDINE", "BECKI", "ARLETHA", "ARGELIA", "ARA", "ALITA", "YULANDA", "YON", "YESSENIA",
"TOBI", "TASIA", "SYLVIE", "SHIRL", "SHIRELY", "SHERIDAN", "SHELLA", "SHANTELLE", "SACHA", "ROYCE",
"REBECKA", "REAGAN", "PROVIDENCIA", "PAULENE", "MISHA", "MIKI", "MARLINE", "MARICA", "LORITA", "LATOYIA",
"LASONYA", "KERSTIN", "KENDA", "KEITHA", "KATHRIN", "JAYMIE", "JACK", "GRICELDA", "GINETTE", "ERYN",
"ELINA", "ELFRIEDA", "DANYEL", "CHEREE", "CHANELLE", "BARRIE", "AVERY", "AURORE", "ANNAMARIA", "ALLEEN",
"AILENE", "AIDE", "YASMINE", "VASHTI", "VALENTINE", "TREASA", "TORY", "TIFFANEY", "SHERYLL", "SHARIE",
"SHANAE", "SAU", "RAISA", "PA", "NEDA", "MITSUKO", "MIRELLA", "MILDA", "MARYANNA", "MARAGRET",
"MABELLE", "LUETTA", "LORINA", "LETISHA", "LATARSHA", "LANELLE", "LAJUANA", "KRISSY", "KARLY", "KARENA",
"JON", "JESSIKA", "JERICA", "JEANELLE", "JANUARY", "JALISA", "JACELYN", "IZOLA", "IVEY", "GREGORY",
"EUNA", "ETHA", "DREW", "DOMITILA", "DOMINICA", "DAINA", "CREOLA", "CARLI", "CAMIE", "BUNNY",
"BRITTNY", "ASHANTI", "ANISHA", "ALEEN", "ADAH", "YASUKO", "WINTER", "VIKI", "VALRIE", "TONA",

"TINISHA", "THI", "TERISA", "TATUM", "TANEKA", "SIMONNE", "SHALANDA", "SERITA", "RESSIE", "REFUGIA",
"PAZ", "OLENE", "NA", "MERRILL", "MARGHERITA", "MANDIE", "MAN", "MAIRE", "LYNDIA", "LUCI",
"LORRIANE", "LORETA", "LEONIA", "LAVONA", "LASHAWNDA", "LAKIA", "KYOKO", "KRYSTINA", "KRYSTEN", "KENIA",
"KELSI", "JUDE", "JEANICE", "ISOBEL", "GEORGIANN", "GENNY", "FELICIDAD", "EILENE", "DEON", "DELOISE",
"DEEDEE", "DANNIE", "CONCEPTION", "CLORA", "CHERILYN", "CHANG", "CALANDRA", "BERRY", "ARMANDINA", "ANISA",
"ULA", "TIMOTHY", "TIERA", "THERESSA", "STEPHANIA", "SIMA", "SHYLA", "SHONTA", "SHERA", "SHAQUITA",
"SHALA", "SAMMY", "ROSSANA", "NOHEMI", "NERY", "MORIAH", "MELITA", "MELIDA", "MELANI", "MARYLYNN",
"MARISHA", "MARIETTE", "MALORIE", "MADELENE", "LUDIVINA", "LORIA", "LORETTE", "LORALEE", "LIANNE", "LEON",
"LAVENIA", "LAURINDA", "LASHON", "KIT", "KIMI", "KEILA", "KATELYNN", "KAI", "JONE", "JOANE",
"JI", "JAYNA", "JANELLA", "JA", "HUE", "HERTHA", "FRANCENE", "ELINORE", "DESPINA", "DELSIE",
"DEEDRA", "CLEMENCIA", "CARRY", "CAROLIN", "CARLOS", "BULAH", "BRITTANIE", "BOK", "BLONDELL", "BIBI",
"BEAULAH", "BEATA", "ANNITA", "AGRIPINA", "VIRGEN", "VALENE", "UN", "TWANDA", "TOMMYE", "TOI",
"TARRA", "TARI", "TAMMERA", "SHAKIA", "SADYE", "RUTHANNE", "ROCHEL", "RIVKA", "PURA", "NENITA",
"NATISHA", "MING", "MERRILEE", "MELODEE", "MARVIS", "LUCILLA", "LEENA", "LAVETA", "LARITA", "LANIE",
"KEREN", "ILEEN", "GEORGEANN", "GENNA", "GENESIS", "FRIDA", "EWA", "EUFEMIA", "EMELY", "ELA",
"EDYTH", "DEONNA", "DEADRA", "DARLENA", "CHANELL", "CHAN", "CATHERN", "CASSONDRA", "CASSAUNDRA", "BERNARDA",
"BERNA", "ARLINDA", "ANAMARIA", "ALBERT", "WESLEY", "VERTIE", "VALERI", "TORRI", "TATYANA", "STASIA",
"SHERISE", "SHERILL", "SEASON", "SCOTTIE", "SANDA", "RUTHE", "ROSY", "ROBERTO", "ROBBI", "RANEE",
"QUYEN", "PEARLY", "PALMIRA", "ONITA", "NISHA", "NIESHA", "NIDA", "NEVADA", "NAM", "MERLYN",
"MAYOLA", "MARYLOUISE", "MARYLAND", "MARX", "MARTH", "MARGENE", "MADELAINE", "LONDA", "LEONTINE", "LEOMA",
"LEIA", "LAWRENCE", "LAURALEE", "LANORA", "LAKITA", "KIYOKO", "KETURAH", "KATELIN", "KAREEN", "JONIE",
"JOHNETTE", "JENEE", "JEANETT", "IZETTA", "HIEDI", "HEIKE", "HASSIE", "HAROLD", "GIUSEPPINA", "GEORGANN",
"FIDELA", "FERNANDE", "ELWANDA", "ELLAMAE", "ELIZ", "DUSTI", "DOTTY", "CYNDY", "CORALIE", "CELESTA",
"ARGENTINA", "ALVERTA", "XENIA", "WAVA", "VANETTA", "TORRIE", "TASHINA", "TANDY", "TAMBRA", "TAMA",
"STEPANIE", "SHILA", "SHAUNTA", "SHARAN", "SHANIQUA", "SHAE", "SETSUKO", "SERAFINA", "SANDEE", "ROSAMARIA",
"PRISCILA", "OLINDA", "NADENE", "MUOI", "MICHELINA", "MERCEDEZ", "MARYROSE", "MARIN", "MARCENE", "MAO",
"MAGALI", "MAFALDA", "LOGAN", "LINN", "LANNIE", "KAYCE", "KAROLINE", "KAMILAH", "KAMALA", "JUSTA",
"JOLINE", "JENNINE", "JACQUETTA", "IRAIDA", "GERALD", "GEORGEANNA", "FRANCHESCA", "FAIRY", "EMELINE", "ELANE",
"EHTEL", "EARLIE", "DULCIE", "DALENE", "CRIS", "CLASSIE", "CHERE", "CHARIS", "CAROYLN", "CARMINA",
"CARITA", "BRIAN", "BETHANIE", "AYAKO", "ARICA", "AN", "ALYSA", "ALESSANDRA", "AKILAH", "ADRIEN",

"ZETTA", "YOULANDA", "YELENA", "YAHAIRA", "XUAN", "WENDOLYN", "VICTOR", "TIJUANA", "TERRELL", "TERINA",
"TERESIA", "SUZI", "SUNDAY", "SHERELL", "SHAVONDA", "SHAUNTE", "SHARDA", "SHAKITA", "SENA", "RYANN",
"RUBI", "RIVA", "REGINIA", "REA", "RACHAL", "PARTHENIA", "PAMULA", "MONNIE", "MONET", "MICHAELE",
"MELIA", "MARINE", "MALKA", "MAISHA", "LISANDRA", "LEO", "LEKISHA", "LEAN", "LAURENCE", "LAKENDRA",
"KRYSTIN", "KORTNEY", "KIZZIE", "KITTIE", "KERA", "KENDAL", "KEMBERLY", "KANISHA", "JULENE", "JULE",
"JOSHUA", "JOHANNE", "JEFFREY", "JAMEE", "HAN", "HALLEY", "GIDGET", "GALINA", "FREDRICKA", "FLETA",
"FATIMAH", "EUSEBIA", "ELZA", "ELEONORE", "DORTHEY", "DORIA", "DONELLA", "DINORAH", "DELORSE", "CLARETHA",
"CHRISTINIA", "CHARLYN", "BONG", "BELKIS", "AZZIE", "ANDERA", "AIKO", "ADENA", "YER", "YAJAIRA",
"WAN", "VANIA", "ULRIKE", "TOSHIA", "TIFANY", "STEFANY", "SHIZUE", "SHENIKA", "SHAWANNA", "SHAROLYN",
"SHARILYN", "SHAQUANA", "SHANTAY", "SEE", "ROZANNE", "ROSELEE", "RICKIE", "REMONA", "REANNA", "RAELENE",
"QUINN", "PHUNG", "PETRONILA", "NATACHA", "NANCEY", "MYRL", "MIYOKO", "MIESHA", "MERIDETH", "MARVELLA",
"MARQUITTA", "MARHTA", "MARCHELLE", "LIZETH", "LIBBIE", "LAHOMA", "LADAWN", "KINA", "KATHELEEN", "KATHARYN",
"KARISA", "KALEIGH", "JUNIE", "JULIEANN", "JOHNSIE", "JANEAN", "JAIMEE", "JACKQUELINE", "HISAKO", "HERMA",
"HELAINE", "GWYNETH", "GLENN", "GITA", "EUSTOLIA", "EMELINA", "ELIN", "EDRIS", "DONNETTE", "DONNETTA",
"DIERDRE", "DENAE", "DARCEL", "CLAUDE", "CLARISA", "CINDERELLA", "CHIA", "CHARLESETTA", "CHARITA", "CELSA",
"CASSY", "CASSI", "CARLEE", "BRUNA", "BRITTANEY", "BRANDE", "BILLI", "BAO", "ANTONETTA", "ANGLA",
"ANGELYN", "ANALISA", "ALANE", "WENONA", "WENDIE", "VERONIQUE", "VANNESA", "TOBIE", "TEMPIE", "SUMIKO",
"SULEMA", "SPARKLE", "SOMER", "SHEBA", "SHAYNE", "SHARICE", "SHANEL", "SHALON", "SAGE", "ROY",
"ROSIO", "ROSELIA", "RENAY", "REMA", "REENA", "PORSCHE", "PING", "PEG", "OZIE", "ORETHA",
"ORALEE", "ODA", "NU", "NGAN", "NAKESHA", "MILLY", "MARYBELLE", "MARLIN", "MARIS", "MARGRETT",
"MARAGARET", "MANIE", "LURLENE", "LILLIA", "LIESELOTTE", "LAVELLE", "LASHAUNDA", "LAKEESHA", "KEITH", "KAYCEE",
"KALYN", "JOYA", "JOETTE", "JENAE", "JANIECE", "ILLA", "GRISEL", "GLAYDS", "GENEVIE", "GALA",
"FREDDA", "FRED", "ELMER", "ELEONOR", "DEBERA", "DEANDREA", "DAN", "CORRINNE", "CORDIA", "CONTESSA",
"COLENE", "CLEOTILDE", "CHARLOTT", "CHANTAY", "CECILLE", "BEATRIS", "AZALEE", "ARLEAN", "ARDATH", "ANJELICA",
"ANJA", "ALFREDIA", "ALEISHA", "ADAM", "ZADA", "YUONNE", "XIAO", "WILLODEAN", "WHITLEY", "VENNIE",
"VANNA", "TYISHA", "TOVA", "TORIE", "TONISHA", "TILDA", "TIEN", "TEMPLE", "SIRENA", "SHERRIL",
"SHANTI", "SHAN", "SENAIDA", "SAMELLA", "ROBBYN", "RENDA", "REITA", "PHEBE", "PAULITA", "NOBUKO",
"NGUYET", "NEOMI", "MOON", "MIKAELA", "MELANIA", "MAXIMINA", "MARG", "MAISIE", "LYNNA", "LILLI",
"LAYNE", "LASHAUN", "LAKENYA", "LAEL", "KIRSTIE", "KATHLINE", "KASHA", "KARLYN", "KARIMA", "JOVAN",
"JOSEFINE", "JENNELL", "JACQUI", "JACKELYN", "HYO", "HIEN", "GRAZYNA", "FLORRIE", "FLORIA", "ELEONORA",

"DWANA", "DORLA", "DONG", "DELMY", "DEJA", "DEDE", "DANN", "CRYSTA", "CLELIA", "CLARIS",
"CLARENCE", "CHIEKO", "CHERLYN", "CHERELLE", "CHARMAIN", "CHARA", "CAMMY", "BEE", "ARNETTE", "ARDELLE",
"ANNIKA", "AMIEE", "AMEE", "ALLENA", "YVONE", "YUKI", "YOSHIE", "YEVETTE", "YAEL", "WILLETTA",
"VONCILE", "VENETTA", "TULA", "TONETTE", "TIMIKA", "TEMIKA", "TELMA", "TEISHA", "TAREN", "TA",
"STACEE", "SHIN", "SHAWNTA", "SATURNINA", "RICARDA", "POK", "PASTY", "ONIE", "NUBIA", "MORA",
"MIKE", "MARIELLE", "MARIELLA", "MARIANELA", "MARDELL", "MANY", "LUANNA", "LOISE", "LISABETH", "LINDSY",
"LILLIANA", "LILLIAM", "LELAH", "LEIGHA", "LEANORA", "LANG", "KRISTEEN", "KHALILAH", "KEELEY", "KANDRA",
"JUNKO", "JOAQUINA", "JERLENE", "JANI", "JAMIKA", "JAME", "HSIU", "HERMILA", "GOLDEN", "GENEVIVE",
"EVIA", "EUGENA", "EMMALINE", "ELFREDA", "ELENE", "DONETTE", "DELCIE", "DEEANNA", "DARCEY", "CUC",
"CLARINDA", "CIRA", "CHAE", "CELINDA", "CATHERYN", "CATHERIN", "CASIMIRA", "CARMELIA", "CAMELLIA", "BREANA",
"BOBETTE", "BERNARDINA", "BEBE", "BASILIA", "ARLYNE", "AMAL", "ALAYNA", "ZONIA", "ZENIA", "YURIKO",
"YAEKO", "WYNELL", "WILLOW", "WILLENA", "VERNIA", "TU", "TRAVIS", "TORA", "TERRILYN", "TERICA",
"TENESHA", "TAWNA", "TAJUANA", "TAINA", "STEPHNIE", "SONA", "SOL", "SINA", "SHONDRA", "SHIZUKO",
"SHERLENE", "SHERICE", "SHARIKA", "ROSSIE", "ROSENA", "RORY", "RIMA", "RIA", "RHEBA", "RENNA",
"PETER", "NATALYA", "NANCEE", "MELODI", "MEDA", "MAXIMA", "MATHA", "MARKETTA", "MARICRUZ", "MARCELENE",
"MALVINA", "LUBA", "LOUETTA", "LEIDA", "LECIA", "LAURAN", "LASHAWNA", "LAINE", "KHADIJAH", "KATERINE",
"KASI", "KALLIE", "JULIETTA", "JESUSITA", "JESTINE", "JESSIA", "JEREMY", "JEFFIE", "JANYCE", "ISADORA",
"GEORGIANNE", "FIDELIA", "EVITA", "EURA", "EULAH", "ESTEFANA", "ELSY", "ELIZABET", "ELADIA", "DODIE",
"DION", "DIA", "DENISSE", "DELORAS", "DELILA", "DAYSI", "DAKOTA", "CURTIS", "CRYSTLE", "CONCHA",
"COLBY", "CLARETTA", "CHU", "CHRISTIA", "CHARLSIE", "CHARLENA", "CARYLON", "BETTYANN", "ASLEY", "ASHLEA",
"AMIRA", "AI", "AGUEDA", "AGNUS", "YUETTE", "VINITA", "VICTORINA", "TYNISHA", "TREENA", "TOCCARA",
"TISH", "THOMASENA", "TEGAN", "SOILA", "SHILOH", "SHENNA", "SHARMAINE", "SHANTAE", "SHANDI", "SEPTEMBER",
"SARAN", "SARAI", "SANA", "SAMUEL", "SALLEY", "ROSETTE", "ROLANDE", "REGINE", "OTELIA", "OSCAR",
"OLEVIA", "NICHOLLE", "NECOLE", "NAIDA", "MYRTA", "MYESHA", "MITSUE", "MINTA", "MERTIE", "MARGY",
"MAHALIA", "MADALENE", "LOVE", "LOURA", "LOREAN", "LEWIS", "LESHA", "LEONIDA", "LENITA", "LAVONE",
"LASHELL", "LASHANDRA", "LAMONICA", "KIMBRA", "KATHERINA", "KARRY", "KANESHA", "JULIO", "JONG", "JENEVA",
"JAQUELYN", "HWA", "GILMA", "GHISLAINE", "GERTRUDIS", "FRANSISCA", "FERMINA", "ETTIE", "ETSUKO", "ELLIS",
"ELLAN", "ELIDIA", "EDRA", "DORETHEA", "DOREATHA", "DENYSE", "DENNY", "DEETTA", "DAINE", "CYRSTAL",
"CORRIN", "CAYLA", "CARLITA", "CAMILA", "BURMA", "BULA", "BUENA", "BLAKE", "BARABARA", "AVRIL",
"AUSTIN", "ALAINE", "ZANA", "WILHEMINA", "WANETTA", "VIRGIL", "VI", "VERONIKA", "VERNON", "VERLINE",

"VASILIKI", "TONITA", "TISA", "TEOFILA", "TAYNA", "TAUNYA", "TANDRA", "TAKAKO", "SUNNI", "SUANNE",
"SIXTA", "SHARELL", "SEEMA", "RUSSELL", "ROSENDA", "ROBENA", "RAYMONDE", "PEI", "PAMILA", "OZELL",
"NEIDA", "NEELY", "MISTIE", "MICHA", "MERISSA", "MAURITA", "MARYLN", "MARYETTA", "MARSHALL", "MARCELL",
"MALENA", "MAKEDA", "MADDIE", "LOVETTA", "LOURIE", "LORRINE", "LORILEE", "LESTER", "LAURENA", "LASHAY",
"LARRAINE", "LAREE", "LACRESHA", "KRISTLE", "KRISHNA", "KEVA", "KEIRA", "KAROLE", "JOIE", "JINNY",
"JEANNETTA", "JAMA", "HEIDY", "GILBERTE", "GEMA", "FAVIOLA", "EVELYNN", "ENDA", "ELLI", "ELLENA",
"DIVINA", "DAGNY", "COLLENE", "CODI", "CINDIE", "CHASSIDY", "CHASIDY", "CATRICE", "CATHERINA", "CASSEY",
"CAROLL", "CARLENA", "CANDRA", "CALISTA", "BRYANNA", "BRITTENY", "BEULA", "BARI", "AUDRIE", "AUDRIA",
"ARDELIA", "ANNELLE", "ANGILA", "ALONA", "ALLYN", "DOUGLAS", "ROGER", "JONATHAN", "RALPH", "NICHOLAS",
"BENJAMIN", "BRUCE", "HARRY", "WAYNE", "STEVE", "HOWARD", "ERNEST", "PHILLIP", "TODD", "CRAIG",
"ALAN", "PHILIP", "EARL", "DANNY", "BRYAN", "STANLEY", "LEONARD", "NATHAN", "MANUEL", "RODNEY",
"MARVIN", "VINCENT", "JEFFERY", "JEFF", "CHAD", "JACOB", "ALFRED", "BRADLEY", "HERBERT", "FREDERICK",
"EDWIN", "DON", "RICKY", "RANDALL", "BARRY", "BERNARD", "LEROY", "MARCUS", "THEODORE", "CLIFFORD",
"MIGUEL", "JIM", "TOM", "CALVIN", "BILL", "LLOYD", "DEREK", "WARREN", "DARRELL", "JEROME",
"FLOYD", "ALVIN", "TIM", "GORDON", "GREG", "JORGE", "DUSTIN", "PEDRO", "DERRICK", "ZACHARY",
"HERMAN", "GLEN", "HECTOR", "RICARDO", "RICK", "BRENT", "RAMON", "GILBERT", "MARC", "REGINALD",
"RUBEN", "NATHANIEL", "RAFAEL", "EDGAR", "MILTON", "RAUL", "BEN", "CHESTER", "DUANE", "FRANKLIN",
"BRAD", "RON", "ROLAND", "ARNOLD", "HARVEY", "JARED", "ERIK", "DARRYL", "NEIL", "JAVIER",
"FERNANDO", "CLINTON", "TED", "MATHEW", "TYRONE", "DARREN", "LANCE", "KURT", "ALLAN", "NELSON",
"GUY", "CLAYTON", "HUGH", "MAX", "DWAYNE", "DWIGHT", "ARMANDO", "FELIX", "EVERETT", "IAN",
"WALLACE", "KEN", "BOB", "ALFREDO", "ALBERTO", "DAVE", "IVAN", "BYRON", "ISAAC", "MORRIS",
"CLIFTON", "WILLARD", "ROSS", "ANDY", "SALVADOR", "KIRK", "SERGIO", "SETH", "KENT", "TERRANCE",
"EDUARDO", "TERRENCE", "ENRIQUE", "WADE", "STUART", "FREDRICK", "ARTURO", "ALEJANDRO", "NICK", "LUTHER",
"WENDELL", "JEREMIAH", "JULIUS", "OTIS", "TREVOR", "OLIVER", "LUKE", "HOMER", "GERARD", "DOUG",
"KENNY", "HUBERT", "LYLE", "MATT", "ALFONSO", "ORLANDO", "REX", "CARLTON", "ERNESTO", "NEAL",
"PABLO", "LORENZO", "OMAR", "WILBUR", "GRANT", "HORACE", "RODERICK", "ABRAHAM", "WILLIS", "RICKEY",
"ANDRES", "CESAR", "JOHNATHAN", "MALCOLM", "RUDOLPH", "DAMON", "KELVIN", "PRESTON", "ALTON", "ARCHIE",
"MARCO", "WM", "PETE", "RANDOLPH", "GARRY", "GEOFFREY", "JONATHON", "FELIPE", "GERARDO", "ED",
"DOMINIC", "DELBERT", "COLIN", "GUILLERMO", "EARNEST", "LUCAS", "BENNY", "SPENCER", "RODOLFO", "MYRON",
"EDMUND", "GARRETT", "SALVATORE", "CEDRIC", "LOWELL", "GREGG", "SHERMAN", "WILSON", "SYLVESTER", "ROOSEVELT",

"ISRAEL", "JERMAINE", "FORREST", "WILBERT", "LELAND", "SIMON", "CLARK", "IRVING", "BRYANT", "OWEN",
"RUFUS", "WOODROW", "KRISTOPHER", "MACK", "LEVI", "MARCOS", "GUSTAVO", "JAKE", "LIONEL", "GILBERTO",
"CLINT", "NICOLAS", "ISMAEL", "ORVILLE", "ERVIN", "DEWEY", "AL", "WILFRED", "JOSH", "HUGO",
"IGNACIO", "CALEB", "TOMAS", "SHELDON", "ERICK", "STEWART", "DOYLE", "DARREL", "ROGELIO", "TERENCE",
"SANTIAGO", "ALONZO", "ELIAS", "BERT", "ELBERT", "RAMIRO", "CONRAD", "NOAH", "GRADY", "PHIL",
"CORNELIUS", "LAMAR", "ROLANDO", "CLAY", "PERCY", "DEXTER", "BRADFORD", "DARIN", "AMOS", "MOSES",
"IRVIN", "SAUL", "ROMAN", "RANDAL", "TIMMY", "DARRIN", "WINSTON", "BRENDAN", "ABEL", "DOMINICK",
"BOYD", "EMILIO", "ELIJAH", "DOMINGO", "EMMETT", "MARLON", "EMANUEL", "JERALD", "EDMOND", "EMIL",
"DEWAYNE", "WILL", "OTTO", "TEDDY", "REYNALDO", "BRET", "JESS", "TRENT", "HUMBERTO", "EMMANUEL",
"STEPHAN", "VICENTE", "LAMONT", "GARLAND", "MILES", "EFRAIN", "HEATH", "RODGER", "HARLEY", "ETHAN",
"ELDON", "ROCKY", "PIERRE", "JUNIOR", "FREDDY", "ELI", "BRYCE", "ANTOINE", "STERLING", "CHASE",
"GROVER", "ELTON", "CLEVELAND", "DYLAN", "CHUCK", "DAMIAN", "REUBEN", "STAN", "AUGUST", "LEONARDO",
"JASPER", "RUSSEL", "ERWIN", "BENITO", "HANS", "MONTE", "BLAINE", "ERNIE", "CURT", "QUENTIN",
"AGUSTIN", "MURRAY", "JAMAL", "ADOLFO", "HARRISON", "TYSON", "BURTON", "BRADY", "ELLIOTT", "WILFREDO",
"BART", "JARROD", "VANCE", "DENIS", "DAMIEN", "JOAQUIN", "HARLAN", "DESMOND", "ELLIOT", "DARWIN",
"GREGORIO", "BUDDY", "XAVIER", "KERMIT", "ROSCOE", "ESTEBAN", "ANTON", "SOLOMON", "SCOTTY", "NORBERT",
"ELVIN", "WILLIAMS", "NOLAN", "ROD", "QUINTON", "HAL", "BRAIN", "ROB", "ELWOOD", "KENDRICK",
"DARIUS", "MOISES", "FIDEL", "THADDEUS", "CLIFF", "MARCEL", "JACKSON", "RAPHAEL", "BRYON", "ARMAND",
"ALVARO", "JEFFRY", "DANE", "JOESPH", "THURMAN", "NED", "RUSTY", "MONTY", "FABIAN", "REGGIE",
"MASON", "GRAHAM", "ISAIAH", "VAUGHN", "GUS", "LOYD", "DIEGO", "ADOLPH", "NORRIS", "MILLARD",
"ROCCO", "GONZALO", "DERICK", "RODRIGO", "WILEY", "RIGOBERTO", "ALPHONSO", "TY", "NOE", "VERN",
"REED", "JEFFERSON", "ELVIS", "BERNARDO", "MAURICIO", "HIRAM", "DONOVAN", "BASIL", "RILEY", "NICKOLAS",
"MAYNARD", "SCOT", "VINCE", "QUINCY", "EDDY", "SEBASTIAN", "FEDERICO", "ULYSSES", "HERIBERTO", "DONNELL",
"COLE", "DAVIS", "GAVIN", "EMERY", "WARD", "ROMEO", "JAYSON", "DANTE", "CLEMENT", "COY",
"MAXWELL", "JARVIS", "BRUNO", "ISSAC", "DUDLEY", "BROCK", "SANFORD", "CARMELO", "BARNEY", "NESTOR",
"STEFAN", "DONNY", "ART", "LINWOOD", "BEAU", "WELDON", "GALEN", "ISIDRO", "TRUMAN", "DELMAR",
"JOHNATHON", "SILAS", "FREDERIC", "DICK", "IRWIN", "MERLIN", "CHARLEY", "MARCELINO", "HARRIS", "CARLO",
"TRENTON", "KURTIS", "HUNTER", "AURELIO", "WINFRED", "VITO", "COLLIN", "DENVER", "CARTER", "LEONEL",
"EMORY", "PASQUALE", "MOHAMMAD", "MARIANO", "DANIAL", "LANDON", "DIRK", "BRANDEN", "ADAN", "BUFORD",
"GERMAN", "WILMER", "EMERSON", "ZACHERY", "FLETCHER", "JACQUES", "ERROL", "DALTON", "MONROE", "JOSUE",

"EDWARDO", "BOOKER", "WILFORD", "SONNY", "SHELTON", "CARSON", "THERON", "RAYMUNDO", "DAREN", "HOUSTON",
"ROBBY", "LINCOLN", "GENARO", "BENNETT", "OCTAVIO", "CORNELL", "HUNG", "ARRON", "ANTONY", "HERSCHEL",
"GIOVANNI", "GARTH", "CYRUS", "CYRIL", "RONNY", "LON", "FREEMAN", "DUNCAN", "KENNITH", "CARMINE",
"ERICH", "CHADWICK", "WILBURN", "RUSS", "REID", "MYLES", "ANDERSON", "MORTON", "JONAS", "FOREST",
"MITCHEL", "MERVIN", "ZANE", "RICH", "JAMEL", "LAZARO", "ALPHONSE", "RANDELL", "MAJOR", "JARRETT",
"BROOKS", "ABDUL", "LUCIANO", "SEYMOUR", "EUGENIO", "MOHAMMED", "VALENTIN", "CHANCE", "ARNULFO", "LUCIEN",
"FERDINAND", "THAD", "EZRA", "ALDO", "RUBIN", "ROYAL", "MITCH", "EARLE", "ABE", "WYATT",
"MARQUIS", "LANNY", "KAREEM", "JAMAR", "BORIS", "ISIAH", "EMILE", "ELMO", "ARON", "LEOPOLDO",
"EVERETTE", "JOSEF", "ELOY", "RODRICK", "REINALDO", "LUCIO", "JERROD", "WESTON", "HERSHEL", "BARTON",
"PARKER", "LEMUEL", "BURT", "JULES", "GIL", "ELISEO", "AHMAD", "NIGEL", "EFREN", "ANTWAN",
"ALDEN", "MARGARITO", "COLEMAN", "DINO", "OSVALDO", "LES", "DEANDRE", "NORMAND", "KIETH", "TREY",
"NORBERTO", "NAPOLEON", "JEROLD", "FRITZ", "ROSENDO", "MILFORD", "CHRISTOPER", "ALFONZO", "LYMAN", "JOSIAH",
"BRANT", "WILTON", "RICO", "JAMAAL", "DEWITT", "BRENTON", "OLIN", "FOSTER", "FAUSTINO", "CLAUDIO",
"JUDSON", "GINO", "EDGARDO", "ALEC", "TANNER", "JARRED", "DONN", "TAD", "PRINCE", "PORFIRIO",
"ODIS", "LENARD", "CHAUNCEY", "TOD", "MEL", "MARCELO", "KORY", "AUGUSTUS", "KEVEN", "HILARIO",
"BUD", "SAL", "ORVAL", "MAURO", "ZACHARIAH", "OLEN", "ANIBAL", "MILO", "JED", "DILLON",
"AMADO", "NEWTON", "LENNY", "RICHIE", "HORACIO", "BRICE", "MOHAMED", "DELMER", "DARIO", "REYES",
"MAC", "JONAH", "JERROLD", "ROBT", "HANK", "RUPERT", "ROLLAND", "KENTON", "DAMION", "ANTONE",
"WALDO", "FREDRIC", "BRADLY", "KIP", "BURL", "WALKER", "TYREE", "JEFFEREY", "AHMED", "WILLY",
"STANFORD", "OREN", "NOBLE", "MOSHE", "MIKEL", "ENOCH", "BRENDON", "QUINTIN", "JAMISON", "FLORENCIO",
"DARRICK", "TOBIAS", "HASSAN", "GIUSEPPE", "DEMARCUS", "CLETUS", "TYRELL", "LYNDON", "KEENAN", "WERNER",
"GERALDO", "COLUMBUS", "CHET", "BERTRAM", "MARKUS", "HUEY", "HILTON", "DWAIN", "DONTE", "TYRON",
"OMER", "ISAIAS", "HIPOLITO", "FERMIN", "ADALBERTO", "BO", "BARRETT", "TEODORO", "MCKINLEY", "MAXIMO",
"GARFIELD", "RALEIGH", "LAWERENCE", "ABRAM", "RASHAD", "KING", "EMMITT", "DARON", "SAMUAL", "MIQUEL",
"EUSEBIO", "DOMENIC", "DARRON", "BUSTER", "WILBER", "RENATO", "JC", "HOYT", "HAYWOOD", "EZEKIEL",
"CHAS", "FLORENTINO", "ELROY", "CLEMENTE", "ARDEN", "NEVILLE", "EDISON", "DESHAWN", "NATHANIAL", "JORDON",
"DANILO", "CLAUD", "SHERWOOD", "RAYMON", "RAYFORD", "CRISTOBAL", "AMBROSE", "TITUS", "HYMAN", "FELTON",
"EZEQUIEL", "ERASMO", "STANTON", "LONNY", "LEN", "IKE", "MILAN", "LINO", "JAROD", "HERB",
"ANDREAS", "WALTON", "RHETT", "PALMER", "DOUGLASS", "CORDELL", "OSWALDO", "ELLSWORTH", "VIRGILIO", "TONEY",
"NATHANAEL", "DEL", "BENEDICT", "MOSE", "JOHNSON", "ISREAL", "GARRET", "FAUSTO", "ASA", "ARLEN",

```
	"ZACK", "WARNER", "MODESTO", "FRANCESCO", "MANUAL", "GAYLORD", "GASTON",
"FILIBERTO", "DEANGELO", "MICHALE",
	"GRANVILLE", "WES", "MALIK", "ZACKARY", "TUAN", "ELDRIDGE", "CRISTOPHER", "CORTEZ",
"ANTIONE", "MALCOM",
	"LONG", "KOREY", "JOSPEH", "COLTON", "WAYLON", "VON", "HOSEA", "SHAD", "SANTO",
"RUDOLF",
	"ROLF", "REY", "RENALDO", "MARCELLUS", "LUCIUS", "KRISTOFER", "BOYCE", "BENTON",
"HAYDEN", "HARLAND",
	"ARNOLDO", "RUEBEN", "LEANDRO", "KRAIG", "JERRELL", "JEROMY", "HOBERT", "CEDRICK",
"ARLIE", "WINFORD",
	"WALLY", "LUIGI", "KENETH", "JACINTO", "GRAIG", "FRANKLYN", "EDMUNDO", "SID",
"PORTER", "LEIF",
	"JERAMY", "BUCK", "WILLIAN", "VINCENZO", "SHON", "LYNWOOD", "JERE", "HAI", "ELDEN",
"DORSEY",
	"DARELL", "BRODERICK", "ALONSO",
]


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 41

```
import eulerlib


def compute():
	Note: The only 1-digit pandigital number is 1, which is not prime. Thus we require n >= 2.
	for n in reversed(range(2, 10)):
		arr = list(reversed(range(1, n + 1)))
		while True:
			if arr[-1] not in NONPRIME_LAST_DIGITS:
				n = int("".join(str(x) for x in arr))
				if eulerlib.is_prime(n):
					return str(n)
			if not prev_permutation(arr):
				break
	raise AssertionError()

NONPRIME_LAST_DIGITS = {0, 2, 4, 5, 6, 8}


def prev_permutation(arr):
	i = len(arr) - 1
	while i > 0 and arr[i - 1] <= arr[i]:
		i -= 1
	if i <= 0:
		return False
	j = len(arr) - 1
	while arr[j] >= arr[i - 1]:
		j -= 1
	arr[i - 1], arr[j] = arr[j], arr[i - 1]
	arr[i : ] = arr[len(arr) - 1 : i - 1 : -1]
	return True
```

```
if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 50

```
import eulerlib, sys
if sys.version_info.major == 2:
	range = xrange


def compute():
	ans = 0
	isprime = eulerlib.list_primality(999999)
	primes = eulerlib.list_primes(999999)
	consecutive = 0
	for i in range(len(primes)):
		sum = primes[i]
		consec = 1
		for j in range(i + 1, len(primes)):
			sum += primes[j]
			consec += 1
			if sum >= len(isprime):
				break
			if isprime[sum] and consec > consecutive:
				ans = sum
				consecutive = consec
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 38

```
def compute():
	ans = ""
	for n in range(2, 10):
		for i in range(1, 10**(9 // n)):
			s = "".join(str(i * j) for j in range(1, n + 1))
			if "".join(sorted(s)) == "123456789":
				ans = max(s, ans)
	return ans


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 42

```
import itertools


def compute():
	ans = sum(1
		for s in WORDS
		if is_triangular_number(sum((ord(c) - ord('A') + 1) for c in s)))
	return str(ans)


def is_triangular_number(n):
	temp = 0
	for i in itertools.count():
		temp += i
		if n == temp:
			return True
		elif n < temp:
			return False


WORDS = [  10 strings per line, except final line
	"A", "ABILITY", "ABLE", "ABOUT", "ABOVE", "ABSENCE", "ABSOLUTELY", "ACADEMIC",
"ACCEPT", "ACCESS",
	"ACCIDENT", "ACCOMPANY", "ACCORDING", "ACCOUNT", "ACHIEVE", "ACHIEVEMENT",
"ACID", "ACQUIRE", "ACROSS", "ACT",
	"ACTION", "ACTIVE", "ACTIVITY", "ACTUAL", "ACTUALLY", "ADD", "ADDITION",
"ADDITIONAL", "ADDRESS", "ADMINISTRATION",
	"ADMIT", "ADOPT", "ADULT", "ADVANCE", "ADVANTAGE", "ADVICE", "ADVISE", "AFFAIR",
"AFFECT", "AFFORD",
	"AFRAID", "AFTER", "AFTERNOON", "AFTERWARDS", "AGAIN", "AGAINST", "AGE", "AGENCY",
"AGENT", "AGO",
	"AGREE", "AGREEMENT", "AHEAD", "AID", "AIM", "AIR", "AIRCRAFT", "ALL", "ALLOW",
"ALMOST",
	"ALONE", "ALONG", "ALREADY", "ALRIGHT", "ALSO", "ALTERNATIVE", "ALTHOUGH",
"ALWAYS", "AMONG", "AMONGST",
	"AMOUNT", "AN", "ANALYSIS", "ANCIENT", "AND", "ANIMAL", "ANNOUNCE", "ANNUAL",
"ANOTHER", "ANSWER",
	"ANY", "ANYBODY", "ANYONE", "ANYTHING", "ANYWAY", "APART", "APPARENT",
"APPARENTLY", "APPEAL", "APPEAR",
	"APPEARANCE", "APPLICATION", "APPLY", "APPOINT", "APPOINTMENT", "APPROACH",
"APPROPRIATE", "APPROVE", "AREA", "ARGUE",
	"ARGUMENT", "ARISE", "ARM", "ARMY", "AROUND", "ARRANGE", "ARRANGEMENT",
"ARRIVE", "ART", "ARTICLE",
	"ARTIST", "AS", "ASK", "ASPECT", "ASSEMBLY", "ASSESS", "ASSESSMENT", "ASSET",
"ASSOCIATE", "ASSOCIATION",
	"ASSUME", "ASSUMPTION", "AT", "ATMOSPHERE", "ATTACH", "ATTACK", "ATTEMPT",
"ATTEND", "ATTENTION", "ATTITUDE",
	"ATTRACT", "ATTRACTIVE", "AUDIENCE", "AUTHOR", "AUTHORITY", "AVAILABLE",
"AVERAGE", "AVOID", "AWARD", "AWARE",
	"AWAY", "AYE", "BABY", "BACK", "BACKGROUND", "BAD", "BAG", "BALANCE", "BALL",
"BAND",
	"BANK", "BAR", "BASE", "BASIC", "BASIS", "BATTLE", "BE", "BEAR", "BEAT", "BEAUTIFUL",
	"BECAUSE", "BECOME", "BED", "BEDROOM", "BEFORE", "BEGIN", "BEGINNING", "BEHAVIOUR",
"BEHIND", "BELIEF",
```

"BELIEVE", "BELONG", "BELOW", "BENEATH", "BENEFIT", "BESIDE", "BEST", "BETTER", "BETWEEN", "BEYOND",
"BIG", "BILL", "BIND", "BIRD", "BIRTH", "BIT", "BLACK", "BLOCK", "BLOOD", "BLOODY",
"BLOW", "BLUE", "BOARD", "BOAT", "BODY", "BONE", "BOOK", "BORDER", "BOTH", "BOTTLE",
"BOTTOM", "BOX", "BOY", "BRAIN", "BRANCH", "BREAK", "BREATH", "BRIDGE", "BRIEF", "BRIGHT",
"BRING", "BROAD", "BROTHER", "BUDGET", "BUILD", "BUILDING", "BURN", "BUS", "BUSINESS", "BUSY",
"BUT", "BUY", "BY", "CABINET", "CALL", "CAMPAIGN", "CAN", "CANDIDATE", "CAPABLE", "CAPACITY",
"CAPITAL", "CAR", "CARD", "CARE", "CAREER", "CAREFUL", "CAREFULLY", "CARRY", "CASE", "CASH",
"CAT", "CATCH", "CATEGORY", "CAUSE", "CELL", "CENTRAL", "CENTRE", "CENTURY", "CERTAIN", "CERTAINLY",
"CHAIN", "CHAIR", "CHAIRMAN", "CHALLENGE", "CHANCE", "CHANGE", "CHANNEL", "CHAPTER", "CHARACTER", "CHARACTERISTIC",
"CHARGE", "CHEAP", "CHECK", "CHEMICAL", "CHIEF", "CHILD", "CHOICE", "CHOOSE", "CHURCH", "CIRCLE",
"CIRCUMSTANCE", "CITIZEN", "CITY", "CIVIL", "CLAIM", "CLASS", "CLEAN", "CLEAR", "CLEARLY", "CLIENT",
"CLIMB", "CLOSE", "CLOSELY", "CLOTHES", "CLUB", "COAL", "CODE", "COFFEE", "COLD", "COLLEAGUE",
"COLLECT", "COLLECTION", "COLLEGE", "COLOUR", "COMBINATION", "COMBINE", "COME", "COMMENT", "COMMERCIAL", "COMMISSION",
"COMMIT", "COMMITMENT", "COMMITTEE", "COMMON", "COMMUNICATION", "COMMUNITY", "COMPANY", "COMPARE", "COMPARISON", "COMPETITION",
"COMPLETE", "COMPLETELY", "COMPLEX", "COMPONENT", "COMPUTER", "CONCENTRATE", "CONCENTRATION", "CONCEPT", "CONCERN", "CONCERNED",
"CONCLUDE", "CONCLUSION", "CONDITION", "CONDUCT", "CONFERENCE", "CONFIDENCE", "CONFIRM", "CONFLICT", "CONGRESS", "CONNECT",
"CONNECTION", "CONSEQUENCE", "CONSERVATIVE", "CONSIDER", "CONSIDERABLE", "CONSIDERATION", "CONSIST", "CONSTANT", "CONSTRUCTION", "CONSUMER",
"CONTACT", "CONTAIN", "CONTENT", "CONTEXT", "CONTINUE", "CONTRACT", "CONTRAST", "CONTRIBUTE", "CONTRIBUTION", "CONTROL",
"CONVENTION", "CONVERSATION", "COPY", "CORNER", "CORPORATE", "CORRECT", "COS", "COST", "COULD", "COUNCIL",
"COUNT", "COUNTRY", "COUNTY", "COUPLE", "COURSE", "COURT", "COVER", "CREATE", "CREATION", "CREDIT",
"CRIME", "CRIMINAL", "CRISIS", "CRITERION", "CRITICAL", "CRITICISM", "CROSS", "CROWD", "CRY", "CULTURAL",
"CULTURE", "CUP", "CURRENT", "CURRENTLY", "CURRICULUM", "CUSTOMER", "CUT", "DAMAGE", "DANGER", "DANGEROUS",
"DARK", "DATA", "DATE", "DAUGHTER", "DAY", "DEAD", "DEAL", "DEATH", "DEBATE", "DEBT",
"DECADE", "DECIDE", "DECISION", "DECLARE", "DEEP", "DEFENCE", "DEFENDANT", "DEFINE", "DEFINITION", "DEGREE",
"DELIVER", "DEMAND", "DEMOCRATIC", "DEMONSTRATE", "DENY", "DEPARTMENT", "DEPEND", "DEPUTY", "DERIVE", "DESCRIBE",
"DESCRIPTION", "DESIGN", "DESIRE", "DESK", "DESPITE", "DESTROY", "DETAIL", "DETAILED", "DETERMINE", "DEVELOP",
"DEVELOPMENT", "DEVICE", "DIE", "DIFFERENCE", "DIFFERENT", "DIFFICULT", "DIFFICULTY", "DINNER", "DIRECT", "DIRECTION",
"DIRECTLY", "DIRECTOR", "DISAPPEAR", "DISCIPLINE", "DISCOVER", "DISCUSS", "DISCUSSION", "DISEASE", "DISPLAY", "DISTANCE",
"DISTINCTION", "DISTRIBUTION", "DISTRICT", "DIVIDE", "DIVISION", "DO", "DOCTOR", "DOCUMENT", "DOG", "DOMESTIC",
"DOOR", "DOUBLE", "DOUBT", "DOWN", "DRAW", "DRAWING", "DREAM", "DRESS", "DRINK", "DRIVE",
"DRIVER", "DROP", "DRUG", "DRY", "DUE", "DURING", "DUTY", "EACH", "EAR", "EARLY",
"EARN", "EARTH", "EASILY", "EAST", "EASY", "EAT", "ECONOMIC", "ECONOMY", "EDGE", "EDITOR",

"EDUCATION", "EDUCATIONAL", "EFFECT", "EFFECTIVE", "EFFECTIVELY", "EFFORT", "EGG", "EITHER", "ELDERLY", "ELECTION",
"ELEMENT", "ELSE", "ELSEWHERE", "EMERGE", "EMPHASIS", "EMPLOY", "EMPLOYEE", "EMPLOYER", "EMPLOYMENT", "EMPTY",
"ENABLE", "ENCOURAGE", "END", "ENEMY", "ENERGY", "ENGINE", "ENGINEERING", "ENJOY", "ENOUGH", "ENSURE",
"ENTER", "ENTERPRISE", "ENTIRE", "ENTIRELY", "ENTITLE", "ENTRY", "ENVIRONMENT", "ENVIRONMENTAL", "EQUAL", "EQUALLY",
"EQUIPMENT", "ERROR", "ESCAPE", "ESPECIALLY", "ESSENTIAL", "ESTABLISH", "ESTABLISHMENT", "ESTATE", "ESTIMATE", "EVEN",
"EVENING", "EVENT", "EVENTUALLY", "EVER", "EVERY", "EVERYBODY", "EVERYONE", "EVERYTHING", "EVIDENCE", "EXACTLY",
"EXAMINATION", "EXAMINE", "EXAMPLE", "EXCELLENT", "EXCEPT", "EXCHANGE", "EXECUTIVE", "EXERCISE", "EXHIBITION", "EXIST",
"EXISTENCE", "EXISTING", "EXPECT", "EXPECTATION", "EXPENDITURE", "EXPENSE", "EXPENSIVE", "EXPERIENCE", "EXPERIMENT", "EXPERT",
"EXPLAIN", "EXPLANATION", "EXPLORE", "EXPRESS", "EXPRESSION", "EXTEND", "EXTENT", "EXTERNAL", "EXTRA", "EXTREMELY",
"EYE", "FACE", "FACILITY", "FACT", "FACTOR", "FACTORY", "FAIL", "FAILURE", "FAIR", "FAIRLY",
"FAITH", "FALL", "FAMILIAR", "FAMILY", "FAMOUS", "FAR", "FARM", "FARMER", "FASHION", "FAST",
"FATHER", "FAVOUR", "FEAR", "FEATURE", "FEE", "FEEL", "FEELING", "FEMALE", "FEW", "FIELD",
"FIGHT", "FIGURE", "FILE", "FILL", "FILM", "FINAL", "FINALLY", "FINANCE", "FINANCIAL", "FIND",
"FINDING", "FINE", "FINGER", "FINISH", "FIRE", "FIRM", "FIRST", "FISH", "FIT", "FIX",
"FLAT", "FLIGHT", "FLOOR", "FLOW", "FLOWER", "FLY", "FOCUS", "FOLLOW", "FOLLOWING", "FOOD",
"FOOT", "FOOTBALL", "FOR", "FORCE", "FOREIGN", "FOREST", "FORGET", "FORM", "FORMAL", "FORMER",
"FORWARD", "FOUNDATION", "FREE", "FREEDOM", "FREQUENTLY", "FRESH", "FRIEND", "FROM", "FRONT", "FRUIT",
"FUEL", "FULL", "FULLY", "FUNCTION", "FUND", "FUNNY", "FURTHER", "FUTURE", "GAIN", "GAME",
"GARDEN", "GAS", "GATE", "GATHER", "GENERAL", "GENERALLY", "GENERATE", "GENERATION", "GENTLEMAN", "GET",
"GIRL", "GIVE", "GLASS", "GO", "GOAL", "GOD", "GOLD", "GOOD", "GOVERNMENT", "GRANT",
"GREAT", "GREEN", "GREY", "GROUND", "GROUP", "GROW", "GROWING", "GROWTH", "GUEST", "GUIDE",
"GUN", "HAIR", "HALF", "HALL", "HAND", "HANDLE", "HANG", "HAPPEN", "HAPPY", "HARD",
"HARDLY", "HATE", "HAVE", "HE", "HEAD", "HEALTH", "HEAR", "HEART", "HEAT", "HEAVY",
"HELL", "HELP", "HENCE", "HER", "HERE", "HERSELF", "HIDE", "HIGH", "HIGHLY", "HILL",
"HIM", "HIMSELF", "HIS", "HISTORICAL", "HISTORY", "HIT", "HOLD", "HOLE", "HOLIDAY", "HOME",
"HOPE", "HORSE", "HOSPITAL", "HOT", "HOTEL", "HOUR", "HOUSE", "HOUSEHOLD", "HOUSING", "HOW",
"HOWEVER", "HUGE", "HUMAN", "HURT", "HUSBAND", "I", "IDEA", "IDENTIFY", "IF", "IGNORE",
"ILLUSTRATE", "IMAGE", "IMAGINE", "IMMEDIATE", "IMMEDIATELY", "IMPACT", "IMPLICATION", "IMPLY", "IMPORTANCE", "IMPORTANT",
"IMPOSE", "IMPOSSIBLE", "IMPRESSION", "IMPROVE", "IMPROVEMENT", "IN", "INCIDENT", "INCLUDE", "INCLUDING", "INCOME",
"INCREASE", "INCREASED", "INCREASINGLY", "INDEED", "INDEPENDENT", "INDEX", "INDICATE", "INDIVIDUAL", "INDUSTRIAL", "INDUSTRY",
"INFLUENCE", "INFORM", "INFORMATION", "INITIAL", "INITIATIVE", "INJURY", "INSIDE", "INSIST", "INSTANCE", "INSTEAD",
"INSTITUTE", "INSTITUTION", "INSTRUCTION", "INSTRUMENT", "INSURANCE", "INTEND", "INTENTION", "INTEREST", "INTERESTED", "INTERESTING",
"INTERNAL", "INTERNATIONAL", "INTERPRETATION", "INTERVIEW", "INTO", "INTRODUCE", "INTRODUCTION", "INVESTIGATE", "INVESTIGATION", "INVESTMENT",

"INVITE", "INVOLVE", "IRON", "IS", "ISLAND", "ISSUE", "IT", "ITEM", "ITS", "ITSELF",
"JOB", "JOIN", "JOINT", "JOURNEY", "JUDGE", "JUMP", "JUST", "JUSTICE", "KEEP", "KEY",
"KID", "KILL", "KIND", "KING", "KITCHEN", "KNEE", "KNOW", "KNOWLEDGE", "LABOUR", "LACK",
"LADY", "LAND", "LANGUAGE", "LARGE", "LARGELY", "LAST", "LATE", "LATER", "LATTER", "LAUGH",
"LAUNCH", "LAW", "LAWYER", "LAY", "LEAD", "LEADER", "LEADERSHIP", "LEADING", "LEAF", "LEAGUE",
"LEAN", "LEARN", "LEAST", "LEAVE", "LEFT", "LEG", "LEGAL", "LEGISLATION", "LENGTH", "LESS",
"LET", "LETTER", "LEVEL", "LIABILITY", "LIBERAL", "LIBRARY", "LIE", "LIFE", "LIFT", "LIGHT",
"LIKE", "LIKELY", "LIMIT", "LIMITED", "LINE", "LINK", "LIP", "LIST", "LISTEN", "LITERATURE",
"LITTLE", "LIVE", "LIVING", "LOAN", "LOCAL", "LOCATION", "LONG", "LOOK", "LORD", "LOSE",
"LOSS", "LOT", "LOVE", "LOVELY", "LOW", "LUNCH", "MACHINE", "MAGAZINE", "MAIN", "MAINLY",
"MAINTAIN", "MAJOR", "MAJORITY", "MAKE", "MALE", "MAN", "MANAGE", "MANAGEMENT", "MANAGER", "MANNER",
"MANY", "MAP", "MARK", "MARKET", "MARRIAGE", "MARRIED", "MARRY", "MASS", "MASTER", "MATCH",
"MATERIAL", "MATTER", "MAY", "MAYBE", "ME", "MEAL", "MEAN", "MEANING", "MEANS", "MEANWHILE",
"MEASURE", "MECHANISM", "MEDIA", "MEDICAL", "MEET", "MEETING", "MEMBER", "MEMBERSHIP", "MEMORY", "MENTAL",
"MENTION", "MERELY", "MESSAGE", "METAL", "METHOD", "MIDDLE", "MIGHT", "MILE", "MILITARY", "MILK",
"MIND", "MINE", "MINISTER", "MINISTRY", "MINUTE", "MISS", "MISTAKE", "MODEL", "MODERN", "MODULE",
"MOMENT", "MONEY", "MONTH", "MORE", "MORNING", "MOST", "MOTHER", "MOTION", "MOTOR", "MOUNTAIN",
"MOUTH", "MOVE", "MOVEMENT", "MUCH", "MURDER", "MUSEUM", "MUSIC", "MUST", "MY", "MYSELF",
"NAME", "NARROW", "NATION", "NATIONAL", "NATURAL", "NATURE", "NEAR", "NEARLY", "NECESSARILY", "NECESSARY",
"NECK", "NEED", "NEGOTIATION", "NEIGHBOUR", "NEITHER", "NETWORK", "NEVER", "NEVERTHELESS", "NEW", "NEWS",
"NEWSPAPER", "NEXT", "NICE", "NIGHT", "NO", "NOBODY", "NOD", "NOISE", "NONE", "NOR",
"NORMAL", "NORMALLY", "NORTH", "NORTHERN", "NOSE", "NOT", "NOTE", "NOTHING", "NOTICE", "NOTION",
"NOW", "NUCLEAR", "NUMBER", "NURSE", "OBJECT", "OBJECTIVE", "OBSERVATION", "OBSERVE", "OBTAIN", "OBVIOUS",
"OBVIOUSLY", "OCCASION", "OCCUR", "ODD", "OF", "OFF", "OFFENCE", "OFFER", "OFFICE", "OFFICER",
"OFFICIAL", "OFTEN", "OIL", "OKAY", "OLD", "ON", "ONCE", "ONE", "ONLY", "ONTO",
"OPEN", "OPERATE", "OPERATION", "OPINION", "OPPORTUNITY", "OPPOSITION", "OPTION", "OR", "ORDER", "ORDINARY",
"ORGANISATION", "ORGANISE", "ORGANIZATION", "ORIGIN", "ORIGINAL", "OTHER", "OTHERWISE", "OUGHT", "OUR", "OURSELVES",
"OUT", "OUTCOME", "OUTPUT", "OUTSIDE", "OVER", "OVERALL", "OWN", "OWNER", "PACKAGE", "PAGE",
"PAIN", "PAINT", "PAINTING", "PAIR", "PANEL", "PAPER", "PARENT", "PARK", "PARLIAMENT", "PART",
"PARTICULAR", "PARTICULARLY", "PARTLY", "PARTNER", "PARTY", "PASS", "PASSAGE", "PAST", "PATH", "PATIENT",
"PATTERN", "PAY", "PAYMENT", "PEACE", "PENSION", "PEOPLE", "PER", "PERCENT", "PERFECT", "PERFORM",
"PERFORMANCE", "PERHAPS", "PERIOD", "PERMANENT", "PERSON", "PERSONAL", "PERSUADE", "PHASE", "PHONE", "PHOTOGRAPH",
"PHYSICAL", "PICK", "PICTURE", "PIECE", "PLACE", "PLAN", "PLANNING", "PLANT", "PLASTIC", "PLATE",

"PLAY", "PLAYER", "PLEASE", "PLEASURE", "PLENTY", "PLUS", "POCKET", "POINT", "POLICE", "POLICY",
"POLITICAL", "POLITICS", "POOL", "POOR", "POPULAR", "POPULATION", "POSITION", "POSITIVE", "POSSIBILITY", "POSSIBLE",
"POSSIBLY", "POST", "POTENTIAL", "POUND", "POWER", "POWERFUL", "PRACTICAL", "PRACTICE", "PREFER", "PREPARE",
"PRESENCE", "PRESENT", "PRESIDENT", "PRESS", "PRESSURE", "PRETTY", "PREVENT", "PREVIOUS", "PREVIOUSLY", "PRICE",
"PRIMARY", "PRIME", "PRINCIPLE", "PRIORITY", "PRISON", "PRISONER", "PRIVATE", "PROBABLY", "PROBLEM", "PROCEDURE",
"PROCESS", "PRODUCE", "PRODUCT", "PRODUCTION", "PROFESSIONAL", "PROFIT", "PROGRAM", "PROGRAMME", "PROGRESS", "PROJECT",
"PROMISE", "PROMOTE", "PROPER", "PROPERLY", "PROPERTY", "PROPORTION", "PROPOSE", "PROPOSAL", "PROSPECT", "PROTECT",
"PROTECTION", "PROVE", "PROVIDE", "PROVIDED", "PROVISION", "PUB", "PUBLIC", "PUBLICATION", "PUBLISH", "PULL",
"PUPIL", "PURPOSE", "PUSH", "PUT", "QUALITY", "QUARTER", "QUESTION", "QUICK", "QUICKLY", "QUIET",
"QUITE", "RACE", "RADIO", "RAILWAY", "RAIN", "RAISE", "RANGE", "RAPIDLY", "RARE", "RATE",
"RATHER", "REACH", "REACTION", "READ", "READER", "READING", "READY", "REAL", "REALISE", "REALITY",
"REALIZE", "REALLY", "REASON", "REASONABLE", "RECALL", "RECEIVE", "RECENT", "RECENTLY", "RECOGNISE", "RECOGNITION",
"RECOGNIZE", "RECOMMEND", "RECORD", "RECOVER", "RED", "REDUCE", "REDUCTION", "REFER", "REFERENCE", "REFLECT",
"REFORM", "REFUSE", "REGARD", "REGION", "REGIONAL", "REGULAR", "REGULATION", "REJECT", "RELATE", "RELATION",
"RELATIONSHIP", "RELATIVE", "RELATIVELY", "RELEASE", "RELEVANT", "RELIEF", "RELIGION", "RELIGIOUS", "RELY", "REMAIN",
"REMEMBER", "REMIND", "REMOVE", "REPEAT", "REPLACE", "REPLY", "REPORT", "REPRESENT", "REPRESENTATION", "REPRESENTATIVE",
"REQUEST", "REQUIRE", "REQUIREMENT", "RESEARCH", "RESOURCE", "RESPECT", "RESPOND", "RESPONSE", "RESPONSIBILITY", "RESPONSIBLE",
"REST", "RESTAURANT", "RESULT", "RETAIN", "RETURN", "REVEAL", "REVENUE", "REVIEW", "REVOLUTION", "RICH",
"RIDE", "RIGHT", "RING", "RISE", "RISK", "RIVER", "ROAD", "ROCK", "ROLE", "ROLL",
"ROOF", "ROOM", "ROUND", "ROUTE", "ROW", "ROYAL", "RULE", "RUN", "RURAL", "SAFE",
"SAFETY", "SALE", "SAME", "SAMPLE", "SATISFY", "SAVE", "SAY", "SCALE", "SCENE", "SCHEME",
"SCHOOL", "SCIENCE", "SCIENTIFIC", "SCIENTIST", "SCORE", "SCREEN", "SEA", "SEARCH", "SEASON", "SEAT",
"SECOND", "SECONDARY", "SECRETARY", "SECTION", "SECTOR", "SECURE", "SECURITY", "SEE", "SEEK", "SEEM",
"SELECT", "SELECTION", "SELL", "SEND", "SENIOR", "SENSE", "SENTENCE", "SEPARATE", "SEQUENCE", "SERIES",
"SERIOUS", "SERIOUSLY", "SERVANT", "SERVE", "SERVICE", "SESSION", "SET", "SETTLE", "SETTLEMENT", "SEVERAL",
"SEVERE", "SEX", "SEXUAL", "SHAKE", "SHALL", "SHAPE", "SHARE", "SHE", "SHEET", "SHIP",
"SHOE", "SHOOT", "SHOP", "SHORT", "SHOT", "SHOULD", "SHOULDER", "SHOUT", "SHOW", "SHUT",
"SIDE", "SIGHT", "SIGN", "SIGNAL", "SIGNIFICANCE", "SIGNIFICANT", "SILENCE", "SIMILAR", "SIMPLE", "SIMPLY",
"SINCE", "SING", "SINGLE", "SIR", "SISTER", "SIT", "SITE", "SITUATION", "SIZE", "SKILL",
"SKIN", "SKY", "SLEEP", "SLIGHTLY", "SLIP", "SLOW", "SLOWLY", "SMALL", "SMILE", "SO",
"SOCIAL", "SOCIETY", "SOFT", "SOFTWARE", "SOIL", "SOLDIER", "SOLICITOR", "SOLUTION", "SOME", "SOMEBODY",
"SOMEONE", "SOMETHING", "SOMETIMES", "SOMEWHAT", "SOMEWHERE", "SON", "SONG", "SOON", "SORRY", "SORT",

"SOUND", "SOURCE", "SOUTH", "SOUTHERN", "SPACE", "SPEAK", "SPEAKER", "SPECIAL", "SPECIES", "SPECIFIC",
"SPEECH", "SPEED", "SPEND", "SPIRIT", "SPORT", "SPOT", "SPREAD", "SPRING", "STAFF", "STAGE",
"STAND", "STANDARD", "STAR", "START", "STATE", "STATEMENT", "STATION", "STATUS", "STAY", "STEAL",
"STEP", "STICK", "STILL", "STOCK", "STONE", "STOP", "STORE", "STORY", "STRAIGHT", "STRANGE",
"STRATEGY", "STREET", "STRENGTH", "STRIKE", "STRONG", "STRONGLY", "STRUCTURE", "STUDENT", "STUDIO", "STUDY",
"STUFF", "STYLE", "SUBJECT", "SUBSTANTIAL", "SUCCEED", "SUCCESS", "SUCCESSFUL", "SUCH", "SUDDENLY", "SUFFER",
"SUFFICIENT", "SUGGEST", "SUGGESTION", "SUITABLE", "SUM", "SUMMER", "SUN", "SUPPLY", "SUPPORT", "SUPPOSE",
"SURE", "SURELY", "SURFACE", "SURPRISE", "SURROUND", "SURVEY", "SURVIVE", "SWITCH", "SYSTEM", "TABLE",
"TAKE", "TALK", "TALL", "TAPE", "TARGET", "TASK", "TAX", "TEA", "TEACH", "TEACHER",
"TEACHING", "TEAM", "TEAR", "TECHNICAL", "TECHNIQUE", "TECHNOLOGY", "TELEPHONE", "TELEVISION", "TELL", "TEMPERATURE",
"TEND", "TERM", "TERMS", "TERRIBLE", "TEST", "TEXT", "THAN", "THANK", "THANKS", "THAT",
"THE", "THEATRE", "THEIR", "THEM", "THEME", "THEMSELVES", "THEN", "THEORY", "THERE", "THEREFORE",
"THESE", "THEY", "THIN", "THING", "THINK", "THIS", "THOSE", "THOUGH", "THOUGHT", "THREAT",
"THREATEN", "THROUGH", "THROUGHOUT", "THROW", "THUS", "TICKET", "TIME", "TINY", "TITLE", "TO",
"TODAY", "TOGETHER", "TOMORROW", "TONE", "TONIGHT", "TOO", "TOOL", "TOOTH", "TOP", "TOTAL",
"TOTALLY", "TOUCH", "TOUR", "TOWARDS", "TOWN", "TRACK", "TRADE", "TRADITION", "TRADITIONAL", "TRAFFIC",
"TRAIN", "TRAINING", "TRANSFER", "TRANSPORT", "TRAVEL", "TREAT", "TREATMENT", "TREATY", "TREE", "TREND",
"TRIAL", "TRIP", "TROOP", "TROUBLE", "TRUE", "TRUST", "TRUTH", "TRY", "TURN", "TWICE",
"TYPE", "TYPICAL", "UNABLE", "UNDER", "UNDERSTAND", "UNDERSTANDING", "UNDERTAKE", "UNEMPLOYMENT", "UNFORTUNATELY", "UNION",
"UNIT", "UNITED", "UNIVERSITY", "UNLESS", "UNLIKELY", "UNTIL", "UP", "UPON", "UPPER", "URBAN",
"US", "USE", "USED", "USEFUL", "USER", "USUAL", "USUALLY", "VALUE", "VARIATION", "VARIETY",
"VARIOUS", "VARY", "VAST", "VEHICLE", "VERSION", "VERY", "VIA", "VICTIM", "VICTORY", "VIDEO",
"VIEW", "VILLAGE", "VIOLENCE", "VISION", "VISIT", "VISITOR", "VITAL", "VOICE", "VOLUME", "VOTE",
"WAGE", "WAIT", "WALK", "WALL", "WANT", "WAR", "WARM", "WARN", "WASH", "WATCH",
"WATER", "WAVE", "WAY", "WE", "WEAK", "WEAPON", "WEAR", "WEATHER", "WEEK", "WEEKEND",
"WEIGHT", "WELCOME", "WELFARE", "WELL", "WEST", "WESTERN", "WHAT", "WHATEVER", "WHEN", "WHERE",
"WHEREAS", "WHETHER", "WHICH", "WHILE", "WHILST", "WHITE", "WHO", "WHOLE", "WHOM", "WHOSE",
"WHY", "WIDE", "WIDELY", "WIFE", "WILD", "WILL", "WIN", "WIND", "WINDOW", "WINE",
"WING", "WINNER", "WINTER", "WISH", "WITH", "WITHDRAW", "WITHIN", "WITHOUT", "WOMAN", "WONDER",
"WONDERFUL", "WOOD", "WORD", "WORK", "WORKER", "WORKING", "WORKS", "WORLD", "WORRY", "WORTH",
"WOULD", "WRITE", "WRITER", "WRITING", "WRONG", "YARD", "YEAH", "YEAR", "YES", "YESTERDAY",
"YET", "YOU", "YOUNG", "YOUR", "YOURSELF", "YOUTH",
]

```
if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 35

```
import eulerlib, sys
if sys.version_info.major == 2:
	range = xrange


def compute():
	isprime = eulerlib.list_primality(999999)
	def is_circular_prime(n):
		s = str(n)
		return all(isprime[int(s[i : ] + s[ : i])] for i in range(len(s)))

	ans = sum(1
		for i in range(len(isprime))
		if is_circular_prime(i))
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 18

We create a new blank triangle with the same dimensions as the original big triangle.
For each cell of the big triangle, we consider the sub-triangle whose top is at this cell,
calculate the maximum path sum when starting from this cell, and store the result
in the corresponding cell of the blank triangle.

If we start at a particular cell, what is the maximum path total? If the cell is at the
bottom of the big triangle, then it is simply the cell's value. Otherwise the answer is
the cell's value plus either {the maximum path total of the cell down and to the left}
or {the maximum path total of the cell down and to the right}, whichever is greater.
By computing the blank triangle's values from bottom up, the dependent values are always
computed before they are utilized. This technique is known as dynamic programming.

```
def compute():
	for i in reversed(range(len(triangle) - 1)):
		for j in range(len(triangle[i])):
			triangle[i][j] += max(triangle[i + 1][j], triangle[i + 1][j + 1])
	return str(triangle[0][0])


triangle = [  Mutable
	[75],
	[95,64],
```

```
        [17,47,82],
        [18,35,87,10],
        [20, 4,82,47,65],
        [19, 1,23,75, 3,34],
        [88, 2,77,73, 7,63,67],
        [99,65, 4,28, 6,16,70,92],
        [41,41,26,56,83,40,80,70,33],
        [41,48,72,33,47,32,37,16,94,29],
        [53,71,44,65,25,43,91,52,97,51,14],
        [70,11,33,28,77,73,17,78,39,68,17,57],
        [91,71,52,38,17,14,91,43,58,50,27,29,48],
        [63,66, 4,68,89,53,67,30,73,16,69,87,40,31],
        [ 4,62,98,27,23, 9,70,98,73,93,38,53,60, 4,23],
]


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 15

```
import eulerlib


This is a classic combinatorics problem. To get from the top left corner to the bottom right corner of an N*N grid,
it involves making exactly N moves right and N moves down in some order. Because each individual down or right
move
is indistinguishable, there are exactly 2N choose N (binomial coefficient) ways of arranging these moves.
def compute():
        return str(eulerlib.binomial(40, 20))


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 7

```
import eulerlib, itertools, sys
if sys.version_info.major == 2:
        filter = itertools.ifilter


Computers are fast, so we can implement this solution by testing each number
individually for primeness, instead of using the more efficient sieve of Eratosthenes.

The algorithm starts with an infinite stream of incrementing integers starting at 2,
filters them to keep only the prime numbers, drops the first 10000 items,
and finally returns the first item thereafter.
def compute():
        ans = next(itertools.islice(filter(eulerlib.is_prime, itertools.count(2)), 10000, None))
```

```
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 37

```
import eulerlib, itertools, sys
if sys.version_info.major == 2:
	filter = itertools.ifilter


def compute():
	ans = sum(itertools.islice(filter(is_truncatable_prime, itertools.count(10)), 11))
	return str(ans)


def is_truncatable_prime(n):
	Test if left-truncatable
	i = 10
	while i <= n:
		if not eulerlib.is_prime(n % i):
			return False
		i *= 10

	Test if right-truncatable
	while n > 0:
		if not eulerlib.is_prime(n):
			return False
		n //= 10
	return True


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 11

We visit each grid cell and compute the product in the 4 directions starting from that cell.

```
def compute():
	ans = -1
	width = len(GRID[0])
	height = len(GRID)
	for y in range(height):
		for x in range(width):
			if x + CONSECUTIVE <= width:
				ans = max(grid_product(x, y,  1, 0, CONSECUTIVE), ans)
			if y + CONSECUTIVE <= height:
```

```
                        ans = max(grid_product(x, y,  0, 1, CONSECUTIVE), ans)
                    if x + CONSECUTIVE <= width and y + CONSECUTIVE <= height:
                        ans = max(grid_product(x, y,  1, 1, CONSECUTIVE), ans)
                    if x - CONSECUTIVE >= -1    and y + CONSECUTIVE <= height:
                        ans = max(grid_product(x, y, -1, 1, CONSECUTIVE), ans)
        return str(ans)


def grid_product(ox, oy, dx, dy, n):
        result = 1
        for i in range(n):
                result *= GRID[oy + i * dy][ox + i * dx]
        return result


GRID = [
        [ 8, 2,22,97,38,15, 0,40, 0,75, 4, 5, 7,78,52,12,50,77,91, 8],
        [49,49,99,40,17,81,18,57,60,87,17,40,98,43,69,48, 4,56,62, 0],
        [81,49,31,73,55,79,14,29,93,71,40,67,53,88,30, 3,49,13,36,65],
        [52,70,95,23, 4,60,11,42,69,24,68,56, 1,32,56,71,37, 2,36,91],
        [22,31,16,71,51,67,63,89,41,92,36,54,22,40,40,28,66,33,13,80],
        [24,47,32,60,99, 3,45, 2,44,75,33,53,78,36,84,20,35,17,12,50],
        [32,98,81,28,64,23,67,10,26,38,40,67,59,54,70,66,18,38,64,70],
        [67,26,20,68, 2,62,12,20,95,63,94,39,63, 8,40,91,66,49,94,21],
        [24,55,58, 5,66,73,99,26,97,17,78,78,96,83,14,88,34,89,63,72],
        [21,36,23, 9,75, 0,76,44,20,45,35,14, 0,61,33,97,34,31,33,95],
        [78,17,53,28,22,75,31,67,15,94, 3,80, 4,62,16,14, 9,53,56,92],
        [16,39, 5,42,96,35,31,47,55,58,88,24, 0,17,54,24,36,29,85,57],
        [86,56, 0,48,35,71,89, 7, 5,44,44,37,44,60,21,58,51,54,17,58],
        [19,80,81,68, 5,94,47,69,28,73,92,13,86,52,17,77, 4,89,55,40],
        [ 4,52, 8,83,97,35,99,16, 7,97,57,32,16,26,26,79,33,27,98,66],
        [88,36,68,87,57,62,20,72, 3,46,33,67,46,55,12,32,63,93,53,69],
        [ 4,42,16,73,38,25,39,11,24,94,72,18, 8,46,29,32,40,62,76,36],
        [20,69,36,41,72,30,23,88,34,62,99,69,82,67,59,85,74, 4,36,16],
        [20,73,35,29,78,31,90, 1,74,31,49,71,48,86,81,16,23,57, 5,54],
        [ 1,70,54,71,83,51,54,69,16,92,33,48,61,43,52, 1,89,19,67,48],
]
CONSECUTIVE = 4


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 40

```
import sys
if sys.version_info.major == 2:
        range = xrange


def compute():
        s = "".join(str(i) for i in range(1, 1000000))
        ans = 1
        for i in range(7):
                ans *= int(s[10**i - 1])
```

```
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 43

```
import itertools


def compute():
	ans = sum(int("".join(map(str, num)))
		for num in itertools.permutations(list(range(10)))
		if is_substring_divisible(num))
	return str(ans)


DIVISIBILITY_TESTS = [2, 3, 5, 7, 11, 13, 17]

def is_substring_divisible(num):
	return all((num[i + 1] * 100 + num[i + 2] * 10 + num[i + 3]) % p == 0
		for (i, p) in enumerate(DIVISIBILITY_TESTS))


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 12

```
import eulerlib, itertools


def compute():
	triangle = 0
	for i in itertools.count(1):
		triangle += i  This is the ith triangle number, i.e. num = 1 + 2 + ... + i = i * (i + 1) / 2
		if num_divisors(triangle) > 500:
			return str(triangle)


Returns the number of integers in the range [1, n] that divide n.
def num_divisors(n):
	end = eulerlib.sqrt(n)
	result = sum(2
		for i in range(1, end + 1)
		if n % i == 0)
	if end**2 == n:
		result -= 1
	return result
```

```
if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 1

Computers are fast, so we can implement this solution directly without any clever math.

```
def compute():
	ans = sum(x for x in range(1000) if (x % 3 == 0 or x % 5 == 0))
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 30

```
import sys
if sys.version_info.major == 2:
	range = xrange


def compute():
	As stated in the problem, 1 = 1^5 is excluded.
	If a number has at least n >= 7 digits, then even if every digit is 9,
	n * 9^5 is still less than the number (which is at least 10^n).
	ans = sum(i for i in range(2, 1000000) if i == fifth_power_digit_sum(i))
	return str(ans)


def fifth_power_digit_sum(n):
	return sum(int(c)**5 for c in str(n))


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 26

```
import itertools


def compute():
```

```
	ans = max(range(1, 1000), key=reciprocal_cycle_len)
	return str(ans)


def reciprocal_cycle_len(n):
	seen = {}
	x = 1
	for i in itertools.count():
		if x in seen:
			return i - seen[x]
		else:
			seen[x] = i
			x = x * 10 % n


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 13

We do a straightforward sum thanks to Python's built-in arbitrary precision integer type.

```
def compute():
	return str(sum(NUMBERS))[ : 10]


NUMBERS = [
	37107287533902102798797998220837590246510135740250,
	46376937677490009712648124896970078050417018260538,
	74324986199524741059474233309513058123726617309629,
	91942213363574161572522430563301811072406154908250,
	23067588207539346171171980310421047513778063246676,
	89261670696623633820136378418383684178734361726757,
	28112879812849979408065481931592621691275889832738,
	44274228917432520321923589422876796487670272189318,
	47451445736001306439091167216856844588711603153276,
	70386486105843025439939619828917593665686757934951,
	62176457141856560629502157223196586755079324193331,
	64906352462741904929101432445813822663347944758178,
	92575867718337217661963751590579239728245598838407,
	58203565325359399008402633568948830189458628227828,
	80181199384826282014278194139940567587151170094390,
	35398664372827112653829987240784473053190104293586,
	86515506006295864861532075273371959191420517255829,
	71693888707715466499115593487603532921714970056938,
	54370070576826684624621495650076471787294438377604,
	53282654108756828443191190634694037855217779295145,
	36123272525000296071075082563815656710885258350721,
	45876576172410976447339110607218265236877223636045,
	17423706905851860660448207621209813287860733969412,
	81142660418086830619328460811191061556940512689692,
	51934325451728388641918047049293215058642563049483,
	62467221648435076201727918039944693004732956340691,
	15732444386908125794514089057706229429197107928209,
	55037687525678773091862540744969844508330393682126,
```

183363848253301546861961243487676812975343759465 15,
803862875928784902015216855548287172012192577669 54,
781828337579931036147403568564490955270978647975 81,
167263201004368978425535399209318374414978068609 84,
484030981290777917990882187953273644756755908480 30,
870869875513927118545170785441618524243206931503 32,
599594068957565367821070749269665376763262354472 10,
697939506796526947425977097391666937630426339870 85,
410526847082990852113994273657341161827603150012 71,
653786073615010808570091499395125570281987460043 75,
358290353174347173269321235781549826297425527373 07,
949537597651053059469660676831565743771674018752 75,
889028025717332296191766687138199318110487701902 71,
252676802760780030136786809925254634010616328665 26,
362702185404977055856299465806362379931407462559 62,
240744869082311749777923654662572469233228109171 41,
914302881971032885978066697608929386382850253334 03,
344130655780161278159218150055618688364684200904 70,
230530811728164304876237919698424872550366387845 83,
114876969321549028104240201383351244621814417734 70,
637832994906362596664985876182212252255124867645 33,
677201869716985443124195724099139590089523100588 22,
955482553002635207815322967962494816419538682187 74,
760853271322857231104248034561248676970645079952 36,
377742425354112916842768655389262050249103265729 67,
237019132757256752856532482582654630922070585965 22,
297988602722583319131263751473419948895347657455 01,
184957014548792889848568277260777137214037988797 15,
382982037830314735277215803481445134913731226651 381,
348295438291999181802789165224310273922511228695 39,
409579530664052326325380441000596549391598795936 35,
297461521855023713076422551211836938038058038858 4903,
416981162220729771861582366784246891579935329619 22,
624679571944012690438771072750481023908955235974 57,
231897067725479150615055049539229795309011299675 19,
861880882258753145295840992512038290094077707756 72,
113067397083047244838165338735023408456470580773 08,
829591747671403631980081871290118754913105471265 81,
976233310448183862695154563349263665728975634005 00,
428462801835170705278318394258821455212272512503 27,
551216035469812005817621652128276527516912968977 89,
322381957343293399464375019078369457658833523998 86,
755061649651847751807381688378610915273579297013 37,
621778427521926234019423996391680449839931733127 31,
329241857071473495669166746876346609150359146775 04,
995186714302352196288948901024233251169136196266 22,
732674608005915474718307983928685352069469445407 24,
768418225246744171615140364279822733480555562148 18,
971426179103425986472045168939894221798260880768 52,
877836461827993463137677543078093633330189826420 90,
108488025216746708832151201858835432238128769527 86,
713296124747824645386369930090493103636197638780 39,
621840735723997942234062353938083396513274080111 16,
666278919814880877979418768761442300309844908514 11,
606618262936828367647447792391803351109890697907 14,
857869440895529906536404474255760836599766457950 96,
660243964099053896071201982199760475994901972302 97,
649139826800329731560371200413779037855660850892 52,
167309393198727502754689069037075394130426523150 11,
948093772450487951509541009216458637547105984367 91,

```
        786391670211874924319957006419179697775990283006 99,
        153687137119366149528113058763802784107544497330 78,
        407899231155355625611423224232550336854424889173 53,
        448899115014406480203690680639606723221932041495 35,
        415031288803395360532993403680069777106505666319 54,
        812348806732101467390585685579345814036278227032 80,
        826165707739483275922328459417065250945123252306 08,
        229188020587773197198394501808880724296619808111 97,
        771585425020165450904132458097868827789487218596 17,
        721078384350691861554356628840622574736922845095 16,
        208496039801340017239306716668235552452528046097 22,
        535035342264725242508740540755917897812643303316 90,
]


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 4

Computers are fast, so we can implement this solution directly without any clever math.

```
def compute():
        ans = max(i * j
                for i in range(100, 1000)
                for j in range(100, 1000)
                if str(i * j) == str(i * j)[ : : -1])
        return str(ans)


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 3

```
import eulerlib
```

By the fundamental theorem of arithmetic, every integer n > 1 has a unique factorization as a product of prime numbers.
In other words, the theorem says that n = p_0 * p_1 * ... * p_{m-1}, where each p_i > 1 is prime but not necessarily unique.
Now if we take the number n and repeatedly divide out its smallest factor (which must also be prime), then the last factor that we divide out must be the largest prime factor of n. For reference, 600851475143 = 71 * 839 * 1471 * 6857.

```
def compute():
        n = 600851475143
        while True:
                p = smallest_prime_factor(n)
                if p < n:
```

```
                n //= p
        else:
                return str(n)


Returns the smallest factor of n, which is in the range [2, n]. The result is always prime.
def smallest_prime_factor(n):
        assert n >= 2
        for i in range(2, eulerlib.sqrt(n) + 1):
                if n % i == 0:
                        return i
        return n  n itself is prime


if __name__ == "__main__":
        print(compute())
```

Solution to Project Euler problem 33

```
import fractions


def compute():
        Consider an arbitrary fraction n/d:
          Let n = 10 * n1 + n0 be the numerator.
          Let d = 10 * d1 + d0 be the denominator.
        As stated in the problem, we need 10 <= n < d < 100.
        We must disregard trivial simplifications where n0 = d0 = 0.

        Now, a simplification with n0 = d0 is impossible because:
          n1 / d1 = n / d = (10*n1 + n0) / (10*d1 + n0).
          n1 * (10*d1 + n0) = d1 * (10*n1 + n0).
          10*n1*d1 + n1*n0 = 10*d1*n1 + d1*n0.
          n1*n0 = d1*n0.
          n1 = d1.
          This implies n = d, which contradicts the fact that n < d.
        Similarly, we cannot have a simplification with n1 = d1 for the same reason.

        Therefore we only need to consider the cases where n0 = d1 or n1 = d0.
        In the first case, check that n1/d0 = n/d;
        in the second case, check that n0/d1 = n/d.
        numer = 1
        denom = 1
        for d in range(10, 100):
                for n in range(10, d):
                        n0 = n % 10
                        n1 = n // 10
                        d0 = d % 10
                        d1 = d // 10
                        if (n1 == d0 and n0 * d == n * d1) or (n0 == d1 and n1 * d == n * d0):
                                numer *= n
                                denom *= d
        return str(denom // fractions.gcd(numer, denom))
```

```
if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 9

Computers are fast, so we can implement a brute-force search to directly solve the problem.

```
def compute():
	PERIMETER = 1000
	for a in range(1, PERIMETER + 1):
		for b in range(a + 1, PERIMETER + 1):
			c = PERIMETER - a - b
			if a * a + b * b == c * c:
				It is now implied that b < c, because we have a > 0
				return str(a * b * c)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 32

```
import eulerlib


def compute():
	For contradiction suppose a candidate (x, y, z) has z >= 10000.
	Then x*y consumes at least 5 digits. With the 4 (or fewer) remaining digits, even the
	upper bound of x=99 and y=99 produces a product of x*y < 10000, which is unequal to z.
	Therefore we need the product z < 10000 to be able to find possible x and y values.
	ans = sum(i for i in range(1, 10000) if has_pandigital_product(i))
	return str(ans)


def has_pandigital_product(n):
	Find and examine all factors of n
	for i in range(1, eulerlib.sqrt(n) + 1):
		if n % i == 0:
			temp = str(n) + str(i) + str(n // i)
			if "".join(sorted(temp)) == "123456789":
				return True
	return False


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 44

```
import itertools, sys
if sys.version_info.major == 2:
	range = xrange


def compute():
	pentanum = PentagonalNumberHelper()
	min_d = None  None means not found yet, positive number means found a candidate
	For each upper pentagonal number index, going upward
	for i in itertools.count(2):
		pent_i = pentanum.term(i)
		If the next number down is at least as big as a found difference, then conclude searching
		if min_d is not None and pent_i - pentanum.term(i - 1) >= min_d:
			break

		For each lower pentagonal number index, going downward
		for j in range(i - 1, 0, -1):
			pent_j = pentanum.term(j)
			diff = pent_i - pent_j
			If the difference is at least as big as a found difference, then stop testing lower pentagonal
numbers
			if min_d is not None and diff >= min_d:
				break
			elif pentanum.is_term(pent_i + pent_j) and pentanum.is_term(diff):
				min_d = diff  Found a smaller difference
	return str(min_d)


Provides memoization for generating and testing pentagonal numbers.
class PentagonalNumberHelper(object):
	def __init__(self):
		self.term_list = [0]
		self.term_set = set()

	def term(self, x):
		assert x > 0
		while len(self.term_list) <= x:
			n = len(self.term_list)
			term = (n * (n * 3 - 1)) >> 1
			self.term_list.append(term)
			self.term_set.add(term)
		return self.term_list[x]

	def is_term(self, y):
		assert y > 0
		while self.term_list[-1] < y:
			n = len(self.term_list)
			term = (n * (n * 3 - 1)) >> 1
			self.term_list.append(term)
			self.term_set.add(term)
		return y in self.term_set


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 48

```
def compute():
	MOD = 10**10
	ans = sum(pow(i, i, MOD) for i in range(1, 1001)) % MOD
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 16

We implement this solution in a straightforward way thanks to Python's built-in arbitrary precision integer type.

```
def compute():
	n = 2**1000
	ans = sum(int(c) for c in str(n))
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 23

```
def compute():
	LIMIT = 28124
	divisorsum = [0] * LIMIT
	for i in range(1, LIMIT):
		for j in range(i * 2, LIMIT, i):
			divisorsum[j] += i
	abundantnums = [i for (i, x) in enumerate(divisorsum) if x > i]

	expressible = [False] * LIMIT
	for i in abundantnums:
		for j in abundantnums:
			if i + j < LIMIT:
				expressible[i + j] = True
			else:
				break

	ans = sum(i for (i, x) in enumerate(expressible) if not x)
```

```
	return str(ans)


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 49

```
import eulerlib


def compute():
	LIMIT = 10000
	isprime = eulerlib.list_primality(LIMIT - 1)
	for base in range(1000, LIMIT):
		if isprime[base]:
			for step in range(1, LIMIT):
				a = base + step
				b = a + step
				if     a < LIMIT and isprime[a] and has_same_digits(a, base) \
				   and b < LIMIT and isprime[b] and has_same_digits(b, base) \
				   and (base != 1487 or a != 4817):
					return str(base) + str(a) + str(b)
	raise RuntimeError("Not found")


def has_same_digits(x, y):
	return sorted(str(x)) == sorted(str(y))


if __name__ == "__main__":
	print(compute())
```

Solution to Project Euler problem 5

```
import fractions


The smallest number n that is evenly divisible by every number in a set {k1, k2, ..., k_m}
is also known as the lowest common multiple (LCM) of the set of numbers.
The LCM of two natural numbers x and y is given by LCM(x, y) = x * y / GCD(x, y).
When LCM is applied to a collection of numbers, it is commutative, associative, and idempotent.
Hence LCM(k1, k2, ..., k_m) = LCM(...(LCM(LCM(k1, k2), k3)...), k_m).
def compute():
	ans = 1
	for i in range(1, 21):
		ans *= i // fractions.gcd(i, ans)
	return str(ans)
```

```
if __name__ == "__main__":
        print(compute())
```