

CONTENTS

1 C.py (Digital Tool Station)
2 README.md
3 setup.py
4 requirements.txt
5 Coordinates_Python.txt (Works with the C.py Programming Engine)
6 maincpp-output.py
7 main.cpp (Data Structures Manager)
8 MechanicalComputer.py (Model)
9 Computer.py (Model)
10 Enumerator.py (Model)
11 LICENSE

Functional Files available at:

<https://github.com/DOMIAXEGDE/cpy.git>

C.py |

```
from tkinter import *
import tkinter as tk
import random
import os
import math
from PIL import Image
import subprocess
import io # experiment
import pygame
import importlib
import requests
import webbrowser
import csv
from bs4 import BeautifulSoup
import datetime
import sys
import json
import re
from PIL import ImageGrab
import pickle
from tkinter import simpledialog
from tkinter import Menu, Checkbutton, BooleanVar #For Windows OS
# from tkinter import Menu, Checkbutton #For macOS
# from tkinter import BooleanVar #For macOS
from tkinter.colorchooser import askcolor
import logging
import time
import pyautogui
import keyboard
from tkinter import filedialog
```

```

from PIL import ImageTk
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from qiskit import QuantumCircuit
from qiskit.visualization import circuit_drawer
import schemdraw
import schemdraw.elements as elm
from schemdraw import logic
from itertools import product
from matplotlib import pyplot as plt
import numpy as np
import glob
from collections import defaultdict
from matplotlib.animation import FuncAnimation
import psutil
import schedule
from pathlib import Path
import shutil
import importlib.util
# import pyautogui
# from PIL import Image, experiment

from PIL import Image, ImageDraw
import plotly.graph_objects as go
from PyQt5 import QtWidgets, QtCore
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QLineEdit, QLabel, QPushButton, QScrollArea, QTextEdit
os.environ['QT_API'] = 'pyqt5'
templates = {
    "basic": ["id", "name", "value"],
    "detailed": ["id", "name", "description", "timestamp"]
}
}

print("""Here is a list of one hundred distinct fields within the realm of mathematical research and development:

```

1. Algebraic Geometry
2. Algebraic Topology
3. Algebraic Number Theory
4. Algebraic Combinatorics
5. Analytic Number Theory
6. Applied Mathematics
7. Approximation Theory
8. Arithmetic Geometry
9. Asymptotic Analysis
10. Biomathematics
11. Braid Theory
12. Calculus of Variations

- 13. Category Theory
- 14. Chaos Theory
- 15. Coding Theory
- 16. Combinatorial Topology
- 17. Combinatorics
- 18. Complex Analysis
- 19. Computational Algebra
- 20. Computational Geometry
- 21. Computational Number Theory
- 22. Control Theory
- 23. Cryptography
- 24. Differential Algebra
- 25. Differential Equations
- 26. Differential Geometry
- 27. Differential Topology
- 28. Discrete Geometry
- 29. Discrete Mathematics
- 30. Dynamical Systems
- 31. Elliptic Curves
- 32. Enumerative Combinatorics
- 33. Ergodic Theory
- 34. Experimental Mathematics
- 35. Finite Geometry
- 36. Fluid Dynamics
- 37. Fourier Analysis
- 38. Fractal Geometry
- 39. Functional Analysis
- 40. Fuzzy Mathematics
- 41. Game Theory
- 42. Geometric Analysis
- 43. Geometric Topology
- 44. Graph Theory
- 45. Group Theory
- 46. Harmonic Analysis
- 47. Homological Algebra
- 48. Homotopy Theory
- 49. Hyperbolic Geometry
- 50. Incidence Geometry
- 51. Information Theory
- 52. Integral Equations
- 53. Integrable Systems
- 54. K-Theory
- 55. Knot Theory
- 56. Lie Algebras
- 57. Lie Groups
- 58. Linear Algebra
- 59. Linear Programming
- 60. Logic
- 61. Manifold Theory
- 62. Mathematical Biology

- 63. Mathematical Finance
- 64. Mathematical Logic
- 65. Mathematical Physics
- 66. Matrix Theory
- 67. Measure Theory
- 68. Model Theory
- 69. Noncommutative Geometry
- 70. Nonlinear Dynamics
- 71. Number Theory
- 72. Numerical Analysis
- 73. Operator Algebras
- 74. Operator Theory
- 75. Optimization
- 76. Ordinary Differential Equations
- 77. Partial Differential Equations
- 78. Percolation Theory
- 79. Perturbation Theory
- 80. Probability Theory
- 81. Quantum Algebra
- 82. Quantum Groups
- 83. Quantum Topology
- 84. Queueing Theory
- 85. Real Analysis
- 86. Representation Theory
- 87. Riemannian Geometry
- 88. Ring Theory
- 89. Set Theory
- 90. Singularity Theory
- 91. Spectral Theory
- 92. Statistical Mechanics
- 93. Stochastic Processes
- 94. Symplectic Geometry
- 95. Systems Theory
- 96. Tensor Analysis
- 97. Topological Groups
- 98. Topology
- 99. Tropical Geometry
- 100. Wavelets and Multiresolution Analysis

These fields represent a vast landscape of mathematical inquiry and have extensive applications across various scientific and engineering domains.

Here is a description of twelve broad fields of study that encompass many of the specialized areas of mathematical research and development just listed:

1. Algebraic Structures: This field includes studies in algebraic geometry, algebraic topology, algebraic number theory, and algebraic combinatorics. It involves the exploration of mathematical structures that are fundamentally algebraic in nature, involving operations within sets that follow specific axioms, like groups, rings, and fields. These studies are crucial for understanding geometrical properties through algebraic expressions.

2. Number Theory: Comprising analytic number theory, algebraic number theory, and arithmetic geometry, this field focuses on the properties and relationships of numbers, particularly the integers. It utilizes techniques from a broad range of mathematical disciplines to solve problems related to divisibility, congruences, and the distribution of primes.
3. Combinatorics and Graph Theory: This includes enumerative combinatorics, algebraic combinatorics, and graph theory. It studies combinatorial structures and their algebraic properties to understand configurations and relations like those found in graph structures, designs, and codes.
4. Topological Studies: This field covers areas such as algebraic topology, differential topology, and geometric topology. It deals with properties that are preserved through deformations, twistings, and stretchings of objects, investigating concepts like continuity, compactness, and connectedness.
5. Differential Geometry and Geometric Analysis: This encompasses differential geometry, differential topology, and geometric analysis. It involves the use of calculus and algebra to study problems in geometry, focusing on curves, surfaces, and higher-dimensional analogues.
6. Analytical Studies: Fields like complex analysis, functional analysis, and harmonic analysis fall under this category. They involve the detailed investigation of functions, their spaces, and other related mathematical entities, providing a deep understanding of their behavior and properties.
7. Applied Mathematical Sciences: This includes applied mathematics, mathematical physics, mathematical finance, and biomathematics, focusing on the application of mathematical methods by different fields such as science and engineering. This area applies theories and techniques from the pure parts of mathematics to practical problems.
8. Computational Mathematics: This field covers computational algebra, computational geometry, and numerical analysis. It deals with mathematical research in areas that require large-scale computation and algorithmic precision, often for simulations, optimizations, and complex calculations.
9. Discrete Mathematics: Including discrete geometry, combinatorial topology, and discrete mathematics itself, this field investigates mathematical structures that are fundamentally discrete rather than continuous. It has applications in computer science, cryptography, and information theory.
10. Dynamical Systems and Ergodic Theory: Studying areas like dynamical systems, chaos theory, and ergodic theory, this field explores systems that evolve over time according to specific rules. It examines how these systems behave, evolve, and respond to various inputs over the long term.
11. Mathematical Logic and Foundations: Covering logic, set theory, and model theory, this field studies the formal basis of mathematics. It investigates the principles of mathematical reasoning, the nature of mathematical objects, and the theoretical underpinnings of mathematical theories.
12. Optimization and Control Theory: Encompassing linear programming, optimization, control theory, and systems theory, this field is concerned with finding the best possible solution to a problem, given constraints and objectives, and controlling the behavior of dynamic systems in an optimal manner.

Each of these fields is vast and interconnects with multiple areas of mathematics, illustrating the profound depth and breadth of mathematical research and development.

Enumerative Mathematical Framework of Thought:

1 [Definitions And Operators]

1 Proof = The absence of doubt.

2 Solution = To solve a problem without the cause of another.

3 Problem = The presence of doubt.

4 Context = Only one context is ever present. That is, once true, always true.

5 Number = A physical structure, present due to a state transition.

6 Unit transition = The least effective change of state.

7 Compound transition = A sequence of unit transitions.

8 Set = An array of numbers.

9 Map = One Number results in another Number, by a defined Unit or Compound transition.

10 Claim = A map.

11 Completion = Proof of Claim.

12 And

13 Xor (Exclusively Or)

14 Not

2 [Statement]

1 $1.6 + 1.6 = 1.7$

2 by definition

3 2 of 1.6 and 1.7 1 has achieved 1.11

3 [Statement]

1 $(a - b)$ is prime

2 $i = a$

3 $i = b$

4 Summation of

5 (i + 1)
6 4 5 from 2 to 3
7 The square root of
8 6/(a/2)
9 decimal part of 8
10 Exclusively OR
11 1- (decimal part of 8)
12 9 10 11
13 8 plus or minus 12, is prime
14 zero is less than 10 equal to 12 which is less than 10 equal to one
15 (-1) is the step value of 6
16 (a > b)
17 zero is less than (a- b) which is less than 10 equal to 12
18 1 6 8 12 17 13 15 16
19 Execute 18 as the 1.6 10 1.7
20 19 outputs (The number 2, 3 Xor 5)""")

```
#test
# Define functions for various operations
def list_files(directory='.'):
    return os.listdir(directory)

def get_system_info():
    return {
        'cpu_percent': psutil.cpu_percent(interval=1),
        'virtual_memory': psutil.virtual_memory().__dict__,
        'disk_usage': psutil.disk_usage('/').__dict__
    }

def download_file(url, save_path):
    response = requests.get(url)
    with open(save_path, 'wb') as f:
        f.write(response.content)
    return f'Downloaded {url} to {save_path}'
```

```
def move_mouse(x, y, duration=1):
    try:
        x = int(x)
        y = int(y)
        duration = float(duration)
        pyautogui.moveTo(x, y, duration=duration)
        return f"Moved mouse to ({x}, {y}) in {duration} seconds"
    except ValueError:
        return "Error: x and y must be integers, and duration must be a float or an integer."
    except Exception as e:
        return f"Error executing move_mouse: {e}"

def take_screenshot(save_path):
    screenshot = pyautogui.screenshot()
    screenshot.save(save_path)
    return f"Screenshot saved to {save_path}"

# Define a mapping of command names to functions
COMMANDS = {
    'list_files': list_files,
    'get_system_info': get_system_info,
    'download_file': download_file,
    'move_mouse': move_mouse,
    'take_screenshot': take_screenshot,
    # Add more commands as needed
}

# Create a global dictionary to hold the user-defined variables and functions
global_context = {}

# Function to execute predefined commands
def execute_user_command(command_name, *args):
    if command_name in COMMANDS:
        try:
            result = COMMANDS[command_name](*args)
            return result
        except Exception as e:
            return f"Error executing command: {e}"
    else:
        return f"Unknown command: {command_name}"

# Function to execute dynamic code
def execute_dynamic_code(user_code):
    try:
        # Execute the code within the global context
        exec(user_code, global_context)
    except Exception as e:
        print(f"Error executing code: {e}")

def eval_dynamic_code(user_code):
```

```
try:
    # Evaluate the code within the global context
    result = eval(user_code, global_context)
    print(f"Result: {result}")
except Exception as e:
    print(f"Error evaluating expression: {e}")

# Function to handle multiline input
def multiline_input():
    print("Enter your Python code (type 'end' on a new line to finish):")
    lines = []
    while True:
        line = input()
        if line.strip().lower() == 'end':
            break
        lines.append(line)
    return '\n'.join(lines)

# Interactive CLI
def interactive_cli():
    print("Welcome to the Python CLI for Windows Control")
    print("Type 'help' to see available commands, 'exit' to quit, or prefix with '!' to execute Python code.")
    print("Type 'multi' to enter multiline Python code.")

    while True:
        user_input = input("> ")

        if user_input in ['exit', 'quit']:
            break
        elif user_input == 'help':
            print("Available commands:")
            for cmd in COMMANDS:
                print(f"  {cmd}")
            print("You can also enter Python code prefixed with '!' to execute.")
            print("Type 'multi' to enter multiline Python code.")
        elif user_input == 'multi':
            code = multiline_input()
            execute_dynamic_code(code)
        elif user_input.startswith('!'):
            if "=" in user_input or 'import' in user_input:
                # Handle assignment and execution statements
                execute_dynamic_code(user_input[1:])
            else:
                # Handle evaluation statements
                eval_dynamic_code(user_input[1:])
        else:
            # Execute predefined commands
            command_parts = user_input.split()
            command_name = command_parts[0]
            command_args = command_parts[1:]
```

```

output = execute_user_command(command_name, *command_args)
print(output)
#test

class TuringCompleteAutomaton23:
    def __init__(self):
        self.tape = {}
        self.head_position = 0

    def write_tape23(self, value):
        self.tape[self.head_position] = value
        # Generate the point before moving the head
        self.generate_point23()
        print(f"Written {value} at position {self.head_position}")
        self.move_head23('right', 1) # Move head right after writing and generating point

    def move_head23(self, direction, steps=1):
        old_position = self.head_position
        if direction == 'left':
            self.head_position -= steps
        elif direction == 'right':
            self.head_position += steps
        print(f"Moved from {old_position} to {self.head_position}")

    def generate_point23(self):
        x = self.head_position
        y = self.tape.get(self.head_position, 0)
        print(f"Generated point ({x}, {y})")

    def read_file23(filename):
        with open(filename, 'r', encoding='utf-8') as file:
            return file.read()

    def process_file_content23(content, tca):
        character_map = {
            'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4, 'f': 5, 'g': 6, 'h': 7, 'i': 8, 'j': 9, 'k': 10, 'l': 11, 'm': 12, 'n': 13,
            'o': 14, 'p': 15, 'q': 16, 'r': 17, 's': 18, 't': 19, 'u': 20, 'v': 21, 'w': 22, 'x': 23, 'y': 24, 'z': 25,
            '\n': 26, '\n': 27, '\t': 28, '\\': 29, '\\"': 30, '\\'': 31, '/': 32, '<': 33, '>': 34, '?': 35, '!': 36, '|': 37,
            '@': 38, '#': 39, '^': 40, ']': 41, '[': 42, '{': 43, '}': 44, '\"': 45, '-': 46, '|': 47, '|': 48, '|': 49,
            '£': 50, '$': 51, '%': 52, '^': 53, '&': 54, '*': 55, '(': 56, ')': 57, '_': 58, '_': 59, '+': 60, '=': 61,
            '|': 62, 'A': 63, 'B': 64, 'C': 65, 'D': 66, 'E': 67, 'F': 68, 'G': 69, 'H': 70, 'I': 71, 'J': 72, 'K': 73,
            'L': 74, 'M': 75, 'N': 76, 'O': 77, 'P': 78, 'Q': 79, 'R': 80, 'S': 81, 'T': 82, 'U': 83, 'V': 84, 'W': 85,
            'X': 86, 'Y': 87, 'Z': 88, '0': 89, '1': 90, '2': 91, '3': 92, '4': 93, '5': 94, '6': 95, '7': 96, '8': 97, '9': 98, '|': 99
        }
        for char in content:
            value = character_map.get(char,-1)
            tca.write_tape23(value)

    def main23():
        filename = input("Enter the filename of the .txt file: ")

```

```

content = read_file23(filename)
tca = TuringCompleteAutomaton23()
process_file_content23(content, tca)
plot_coordinates23(tca.tape)

def plot_coordinates23(tape):
    x_vals, y_vals = list(tape.keys()), list(tape.values())
    fig = go.Figure(data=go.Scatter(x=x_vals, y=y_vals, mode='markers+lines', name='Coordinates'))
    fig.update_layout(title='Interactive Plot of Generated Coordinates',
                      xaxis_title='X Axis',
                      yaxis_title='Y Axis')
    fig.show()
# Optionally, you can save the plot as an HTML file
fig.write_html('plot.html')
print("Plot saved as 'plot.html'.")

class TuringCompleteAutomaton:
    def __init__(self):
        self.tape = {}
        self.head_position = 0

    def write_tape(self, value):
        self.tape[self.head_position] = value
        print(f"Written {value} at position {self.head_position}")

    def move_head(self, direction, steps):
        old_position = self.head_position
        if direction == 'left':
            self.head_position -= steps
        elif direction == 'right':
            self.head_position += steps
        print(f"Moved from {old_position} to {self.head_position}")

    def generate_point(self):
        x = self.head_position
        y = self.tape.get(self.head_position, 0)
        print(f"Generated point ({x}, {y})")
        return (x, y)

def main22():
    tca = TuringCompleteAutomaton()
    points = []

    print("Welcome to the Turing Complete Coordinate Point Simulator")
    print("Instructions:")
    print(" write [value]- Write a value at the current tape position.")
    print(" move left [steps]- Move the head left by a specified number of steps.")
    print(" move right [steps]- Move the head right by a specified number of steps.")
    print(" generate- Generate a point based on the current tape and head position.")
    print(" exit- Exit the program.")

```

```

print()

while True:
    command = input("Enter command: ").strip().lower()
    parts = command.split()
    action = parts[0]

    if action == "write" and len(parts) > 1:
        value = float(parts[1])
        tca.write_tape(value)
    elif action in ["move", "move"] and len(parts) == 3:
        direction = parts[1]
        steps = int(parts[2])
        tca.move_head(direction, steps)
    elif action == "generate":
        points.append(tca.generate_point())
    elif action == "exit":
        print("Exiting...")
        break
    else:
        print("Unknown command. Please try again.")

print("Generated Coordinates:", points)
plot_coordinates22(points)

def plot_coordinates22(points):
    x_vals, y_vals = zip(*points) if points else ([], [])
    fig = go.Figure(data=go.Scatter(x=x_vals, y=y_vals, mode='markers+lines', name='Coordinates'))
    fig.update_layout(title='Interactive Plot of Generated Coordinates',
                      xaxis_title='X Axis',
                      yaxis_title='Y Axis')
    fig.show()
    # Optionally, you can save the plot as an HTML file
    fig.write_html('plot.html')
    print("Plot saved as 'plot.html'.")

def is_prime21(num):
    """Check if a number is prime."""
    if num < 2:
        return False
    for i in range(2, int(math.sqrt(num)) + 1):
        if num % i == 0:
            return False
    return True

class ProximalPrimeFSM:
    def __init__(self, initial_a, initial_b, user_iterations):
        self.initial_a = initial_a
        self.initial_b = initial_b
        self.user_iterations = user_iterations

```

```

self.primes = []

def run(self):
    base_iterations = 144 # Default for all combinations
    total_iterations = base_iterations + self.user_iterations
    for _ in range(total_iterations):
        a, b = self.initial_a, self.initial_b
        for j in range(12, 0, -1): # Descending loop for a
            for k in range(12, 0, -1): # Descending loop for b
                self.a = a + j
                self.b = b + k
                if self.validate():
                    p = self.compute_p()
                    z = self.compute_z(p)
                    epsilon1 = z - int(z)
                    return [epsilon1, 1 - epsilon1, 1]
                else:
                    self.prepare_z_values(z, epsilon1)

    # Reset to initial values plus maximum shift for next iteration
    self.initial_a += 12
    self.initial_b += 12

def validate(self):
    return self.a > self.b and 0 < (self.a - self.b) <= 12

def compute_p(self):
    return sum(i + 1 for i in range(self.b, self.a + 1))

def compute_z(self, p):
    return math.sqrt(p / (self.a / 2))

def compute_epsilon1(self, z):
    epsilon1 = z - int(z)
    return [epsilon1, 1 - epsilon1, 1]

def prepare_z_values(self, z, epsilon1):
    z_values = [z + epsilon1 for epsilon1 in epsilon1] + [z - epsilon1 for epsilon1 in epsilon1]
    for z_val in z_values:
        if int(z_val) == z_val and is_prime21(int(z_val)):
            self.primes.append(z_val)

# Prepare 3D coordinates
def prepare_coordinates21(primes):
    coords = []
    for i in range(0, len(primes), 3):
        chunk = primes[i:i+3]
        if len(chunk) < 3:
            chunk += [1] * (3 - len(chunk))
        coords.append(chunk)
    return coords

# Animation setup

```

```

def animate21(i):
    ax.clear()
    ax.scatter(*zip(*coords[:i+1]), c='r', marker='o')
    ax.set_xlim(0, max(coords, key=lambda x: x[0])[0] + 10)
    ax.set_ylim(0, max(coords, key=lambda x: x[1])[1] + 10)
    ax.set_zlim(0, max(coords, key=lambda x: x[2])[2] + 10)

#####
#####

def index_to_rgb19(color_index):
    """Convert a color index to an RGB tuple."""
    red = (color_index >> 16) & 255 # Shift right by 16 bits and mask with 255 to get the red component
    green = (color_index >> 8) & 255 # Shift right by 8 bits and mask with 255 to get the green component
    blue = color_index & 255 # Mask with 255 to get the blue component
    return (red, green, blue)

def calculate_configuration_index19(square_color_index, symbol_index, font_color_index):
    NUM_SYMBOLS = 97
    NUM_FONT_COLORS = 256**3
    index = (square_color_index * NUM_SYMBOLS * NUM_FONT_COLORS) + (symbol_index * NUM_FONT_COLORS) + font_color_index
    return index

def save_configuration_to_file19(file_name, grid_size, square_size, font_size, configurations):
    with open(file_name, 'w') as file:
        file.write(f"Grid Size: {grid_size}x{grid_size}\n")
        file.write(f"Sub-square Pixel Length: {square_size}\n")
        file.write(f"Font Size: {font_size}\n")
        file.write("Configurations:\n")

        for config in configurations:
            square_color_index, symbol_index, font_color_index = config
            index = calculate_configuration_index19(square_color_index, symbol_index, font_color_index)
            # Convert indices to RGB
            square_color_rgb = index_to_rgb19(square_color_index)
            font_color_rgb = index_to_rgb19(font_color_index)
            file.write(f"Index: {index}, Square Color Index: {square_color_index} (RGB: {square_color_rgb}),\n"
                      f"Symbol: {symbol_index}, Font Color Index: {font_color_index} (RGB: {font_color_rgb})\n")

def is_prime18(num):
    """Check if a number is prime. Assumes num is a positive integer."""
    if num < 2:
        return False
    for i in range(2, int(math.sqrt(num)) + 1):
        if num % i == 0:
            return False
    return True

```

```

def main18():
    # Input a and b
    a = int(input("Enter a (a > b): "))
    b = int(input("Enter b (b < a): "))

    # Validate conditions for a and b
    if not (a > b and 0 < (a - b) <= 12):
        print("The conditions 0 < (a- b) <= 12 and a > b are not met.")
        return

    # Calculate p, the sum from i = a to i = b of (i + 1)
    p = sum(i + 1 for i in range(b, a + 1))

    # Calculate z
    z = math.sqrt(p / (a / 2))

    # Calculate epsilon (decimal part of z)
    epsilon1 = z - int(z)
    epsilon2 = 1 - epsilon1
    epsilon3 = 1

    # Prepare the possible z values to check for primality
    z_values = {
        "z + epsilon2": z + epsilon2,
        "z - epsilon1": z - epsilon1,
        "z - epsilon2": z - epsilon2,
        "z + epsilon1": z + epsilon1,
        "z + epsilon3": z + epsilon3,
        "z - epsilon3": z - epsilon3
    }

    # Get file name from user and write the results
    file_name = input("Enter the name of the file to save results: ")
    with open(file_name, "w") as file:
        file.write(f"Input values: a = {a}, b = {b}\n")
        file.write(f"Calculated p: {p}\n")
        file.write(f"Calculated z: {z}\n")
        file.write(f"Calculated epsilon1: {epsilon1}\n")
        file.write(f"Calculated epsilon2: {epsilon2}\n")
        file.write(f"epsilon3: {epsilon3}\n")
        for label, value in z_values.items():
            if int(value) == value:
                prime_status = is_prime18(int(value))
            else:
                prime_status = "N/A (not an integer)"
            file.write(f"Is {label} = {value} prime? {prime_status}\n")

def append_string_to_file2(directory, filename, user_string):
    """Appends the user string to a file if it does not already exist in the file."""
    # Create full file path

```

```

file_path = os.path.join(directory, filename)

# Check if file exists and read its contents if it does
if(os.path.exists(file_path) and len(user_string) > 4):
    with open(file_path, 'r', encoding='utf-8') as file:
        if user_string in file.read():
            print("The string is already in the file.")
            return
    else:
        print(f"\n\n{filename} does exist.")
        print("\n\nNote: Your string must be greater than 4 characters in length. Generate 1.txt, 2.txt, 3.txt, 4.txt in llanguageMod/mappings using mode 12 ...\\n\\n")

if len(user_string) > 4:
    # First, determine the current number of lines in the file
    if os.path.exists(file_path):
        with open(file_path, 'r', encoding='utf-8') as file:
            current_line_number = sum(1 for _ in file) + 1
    else:
        current_line_number = 1 # File doesn't exist yet, so start from line 1

    # Append the string to the file, adding the line number before the string
    with open(file_path, 'a', encoding='utf-8') as file:
        file.write(f"{current_line_number} {user_string}\\n")
    print(f"String added to {filename}")

def main2():
    # Read user input
    user_string = input("Please enter a string: ")

    # Calculate the length of the string
    n = len(user_string)

    # Define directory and filename
    directory = "llanguageMod/mappings"
    filename = f"{n}.txt"

    # Ensure the directory exists
    if not os.path.exists(directory):
        os.makedirs(directory)
    print(f"Created directory {directory}")

    # Append string to the appropriate file
    append_string_to_file2(directory, filename, user_string)

def base_x_to_binary(num, base):
    """Converts a base x integer to a binary string."""
    digits = "0123456789ABCDEF" # Up to base 16
    if base < 2 or base > 16:

```

```

raise ValueError("Base must be between 2 and 16")

binary = ""
while num > 0:
    binary = digits[num % 2] + binary
    num //= 2
return binary

def convert_file(input_file, output_file, base):
    """Converts a file of base x integers to a file of binary strings.

Args:
    input_file: Path to the input file.
    output_file: Path to the output file.
    base: The base of the integers in the input file.
"""
    with open(input_file, 'r') as f_in, open(output_file, 'w') as f_out:
        for line in f_in:
            numbers = [int(x) for x in line.strip().split()]
            binary_strings = [base_x_to_binary(num, base) for num in numbers]
            f_out.write(" ".join(binary_strings) + "\n")

def generate_and_save_n_bit_combinations(file_path, n):
    with open(file_path, "w") as file:
        for i in range(2**n): # 2^n combinations
            combination = bin(i)[2:].zfill(n)
            file.write(combination + "\n")

def decompile_data():
    """Decompiles data using a dynamically loaded decompile function."""
    name = input("Enter the name of the data structure to decompile: ")
    filename = f"{name}_logic.py"
    if not os.path.exists(filename):
        print(f"Logic script for {name} not found. Please ensure the data structure is defined and compiled first.")
        return
    compiled_input = input("Enter compiled data string: ")
    try:
        decompiled_data = execute_dynamic_script(name, f"decompile_{name}", compiled_input)
        print("Decomplied Data:", json.dumps(decompiled_data, indent=4))
    except Exception as e:
        print("An error occurred:", e)

def execute_dynamic_script(name, function_name, data):
    """Dynamically import and execute a function from a generated script."""
    filename = f"{name}_logic.py"
    spec = importlib.util.spec_from_file_location(name, filename)
    module = importlib.util.module_from_spec(spec)

```

```

spec.loader.exec_module(module)
func = getattr(module, function_name)
return func(data)

def populate_data(name):
    """Prompt the user to enter data for the defined structure and save to a text file."""
    filename = f"{name}.json"
    if not os.path.exists(filename):
        print(f"Data structure definition for {name} not found. Please define the data structure first.")
        return

    try:
        with open(filename, 'r') as file:
            data_structure = json.load(file)

        data = {}
        for field in data_structure['fields']:
            data[field] = input(f"Enter value for {field}: ")

        data_filename = f"{name}_data.txt"
        with open(data_filename, 'a') as file:
            file.write(' | '.join(str(data[field]) for field in data_structure['fields'])) + '\n'
        print(f"Data saved to {data_filename}.")

    except FileNotFoundError:
        print("Error loading data structure file. Please check the file system.")

```

```

def countables():
while True:
    print("\nMain Menu:")
    print("1. Define Data Structure")
    print("2. Populate Data")
    print("3. Compile Data")
    print("4. Decompile Data")
    print("5. Exit")
    choice = input("Enter choice: ")

    if choice == "1":
        define_data_structure()
    elif choice == "2":
        name = input("Enter the name of the data structure to populate: ")
        populate_data(name)
    elif choice == "3":
        name = input("Enter the name of the data structure to compile: ")
        compile_data(name)
    elif choice == "4":
        decompile_data()
    elif choice == "5":
        break

```

```

else:
    print("Invalid choice. Please select again.")

# Assuming the execute_dynamic_script function is already defined and imported as discussed.

def define_data_structure():
    template_choice = input("Select template (basic, detailed): ")
    if template_choice not in templates:
        print("Invalid template. Returning to main menu.")
        return

    name = input("Enter name for the data structure: ")
    data_structure = {'name': name, 'fields': templates[template_choice]}
    filename = f"{name}.json"
    with open(filename, 'w') as file:
        json.dump(data_structure, file)
    script_content = generate_scripts(name, templates[template_choice])
    save_script(name, script_content)
    print(f"Data structure {name} defined with template {template_choice} and saved as {filename}.")

def generate_scripts(name, fields):
    compile_func = f"def compile_{name}(data):\n"
    compile_func += "    return '|'.join(str(data[field]) for field in data)\n"

    decompile_func = f"def decompile_{name}(data):\n"
    decompile_func += "    fields = " + str(fields) + "\n"
    decompile_func += "    return dict(zip(fields, data.split('|')))\n"

    return compile_func + "\n" + decompile_func

def save_script(name, content):
    filename = f"{name}_logic.py"
    with open(filename, 'w') as file:
        file.write(content)
    print(f"Compile/Decompile logic saved to {filename}.")

def compile_data(name):
    """Compiles data using a dynamically loaded compile function."""
    filename = f"{name}_logic.py"
    if not os.path.exists(filename):
        print(f"Logic script for {name} not found. Please ensure the data structure is defined and compiled first.")
        return

    print("Enter data to compile (format as JSON, e.g., {'id':'123', 'name':'John'}):")
    print("Note: Ensure your JSON is correctly formatted for your command line environment.")
    data_input = input()
    try:
        # Strip out unwanted characters if necessary (specific to your environment)
        data_input = data_input.strip()

```

```

if data_input.startswith("") and data_input.endswith(""):
    data_input = data_input[1:-1] # Remove surrounding single quotes for Unix/Linux shells
data = json.loads(data_input) # Safely parse the JSON input
compiled_data = execute_dynamic_script(name, f"compile_{name}", data)
print("Compiled Data:", compiled_data)
except json.JSONDecodeError:
    print("Invalid JSON input. Please check the format and try again.")
except Exception as e:
    print("An error occurred:", e)

# Directly invoking the main menu function when the script runs
#countables()
#####
#####

def main0():
    print("\n\nWelcome. Choose your operation: ")
    # User option to generate mappings or not
    generate_mappings = input("Do you want to generate mappings? 1 [for yes], 0 [for no]: ")
    if generate_mappings == "1":
        Llanguage_model()

    # Define input and output directories
    in1 = "llanguageMod/inputs"
    out1 = "llanguageMod/outfile"
    out2 = "llanguageMod/outb"

    proces = input("Do you want to process files? 1 [for yes], 0 [for no]: ")
    if(proces == '1'):
        # Read mappings only if they are necessary
        mappings = read_mappings()
        # Get user input for the range of files to process
        start = int(input("Enter the start of the range of files to read (e.g., 5 for 5.txt): "))
        end = int(input("Enter the end of the range of files to read (e.g., 15 for 15.txt):"))

        # Process files within the specified range
        process_files(in1, out1, start, end, mappings, out2)

    # Optionally, generate an image from text input
    questione = input("Generate text input to image file? 1 [for yes], 0 [for no]: ")
    if questione == "1":
        data = open_file0()
        if data:
            process_image_generation0(data)

def process_image_generation0(data):
    """Generates an image from provided data."""
    if not data:
        print("No data to process for image generation.")

```

```

    return
print("Data received for image generation:", data)
color_data = data_to_color0(data)
if not color_data:
    print("No color data generated.")
    return
image = create_image0(color_data)
image.show() # Optionally display the image
save_option = input("Do you want to save this image as a PDF? (yes/no): ")
if save_option.lower() == 'yes':
    # Assuming 'image' is a PIL Image object you've created or manipulated
    file_path = input("Enter the path to save the PDF file (including filename): ")
    save_image_as_pdf0(image, file_path)

def create_image0(color_data):
    """ Create an image from the color data based on user input dimensions and pixel size. """
    # Dimensions based on the length of color_data to create a square image
    num_pixels_width = int(len(color_data) ** 0.5) # Assuming a roughly square image for simplicity
    num_pixels_height = num_pixels_width if num_pixels_width ** 2 == len(color_data) else
    num_pixels_width + 1

    pixel_side_length = 10 # Each color block will be 10x10 pixels

    img_width = pixel_side_length * num_pixels_width
    img_height = pixel_side_length * num_pixels_height

    # Create a new image with white background
    image = Image.new('RGB', (img_width, img_height), 'white')
    draw = ImageDraw.Draw(image)

    # Draw each block
    index = 0
    for i in range(num_pixels_height):
        for j in range(num_pixels_width):
            if index < len(color_data):
                top_left = (j * pixel_side_length, i * pixel_side_length)
                bottom_right = ((j + 1) * pixel_side_length, (i + 1) * pixel_side_length)
                draw.rectangle([top_left, bottom_right], fill=color_data[index])
            index += 1

    return image

def data_to_color0(data):
    """ Convert each data item to a unique color based on its hash value. """
    if not data: # Validate input
        print("No data provided to convert to color.")
        return []
    return ['#' + '{:06X}'.format(hash(datum)) & 0xFFFF for datum in data if isinstance(datum, str)]

```

```

def open_file():
    """ Prompt user to enter a file path, read the file, and return the content as a list of words. """
    file_path = input("Enter the path of the text file: ")
    try:
        with open(file_path, 'r') as file:
            data = file.read().split()
        print("File successfully opened.")
        return data
    except Exception as e:
        print(f"Error opening file: {e}")
        return []

# Uncomment the following line to run the main function in a script environment
# main()

def Llanguage_model():
    characters = 'abcdefghijklmnopqrstuvwxyz\n\t\\`/,<>?;:@#~[{}\\`-|;!*£$%^&*()_-+=.ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
    # Asking user for input
    try:
        pinned = int(input("Enter the number of characters to use (up to 100), or 100 for all: "))
        characters = characters[:pinned]
    except ValueError:
        print("Invalid number, using all characters.")
        pinned = 100

    start_length = int(input("Enter the start length for combinations (e.g., 1): "))
    end_length = int(input("Enter the end length for combinations (e.g., 4): "))
    create_custom_array = input("Do you want to create a custom array? (yes/no): ").lower() == 'yes'

    if create_custom_array:
        custom_array = []
        print("Enter the indices for your custom character array:")
        for _ in range(pinned):
            index = int(input("Enter index {0 + 1}/{pinned} (0-indexed): "))
            custom_array.append(characters[index])
        characters = custom_array

    # Ensure the output directory exists
    output_directory = "llanguageMod/mappings"
    os.makedirs(output_directory, exist_ok=True)

    for length in range(start_length, end_length + 1):
        with open(f"{output_directory}/{length}.txt", "w") as file:
            print(f"Generating {length}-character combinations...")
            num_combinations = math.pow(len(characters), length)
            for i in range(int(num_combinations)):
                combination = ""

```

```

temp = i
for j in range(length):
    combination = characters[temp % len(characters)] + combination
    temp //= len(characters)
file.write(f"\n{i} {combination}\n")

print("Combinations generated successfully.")

# Example of how to call the function (uncomment the following line to use it in an actual script)
# Llanguage_model()

def read_mappings():
    """Reads mappings from files named 1.txt to 4.txt and stores them in a list of dictionaries."""
    mappings = [{}) for _ in range(4)]
    for i in range(1, 5):
        try:
            with open(f"llanguageMod/mappings/{i}.txt", "r") as file:
                for line in file:
                    parts = line.strip().split()
                    if len(parts) == 2:
                        id_str, value_str = parts
                        mappings[i-1][value_str] = id_str
        except FileNotFoundError:
            print(f"Error opening file mappings/{i}.txt")
    return mappings

def process_files(input_directory, output_directory, start, end, mappings, outdir2):
    """Processes files within a given range using predefined mappings and generates output files."""
    max_file_number = max([int(os.path.splitext(os.path.basename(f))[0]) for f in
    glob.glob(f"{output_directory}/*.{txt}"), default=0)
    for i in range(start, end + 1):
        input_file_path = f"{input_directory}/{i}.txt"
        try:
            process_large_file(input_file_path, output_directory, mappings, max_file_number)
            process_large_file2(input_file_path, outdir2, mappings, max_file_number)
            max_file_number += 1
        except FileNotFoundError:
            print(f"Could not open the file {input_file_path}")

def process_large_file(input_file_path, output_directory, mappings, file_number):
    """Efficiently processes a large file using mappings and saves to an output file."""
    output_file_path = f"{output_directory}/{file_number + 1}.txt"
    with open(input_file_path, "r") as input_file, open(output_file_path, "w") as output_file:
        for line in input_file:
            output_content = convert_content(line, mappings)
            output_file.write(output_content)

def process_large_file2(input_file_path, output_directory, mappings, file_number):
    """Efficiently processes a large file using mappings and saves to an output file."""

```

```

output_file_path = f"{output_directory}/{file_number + 1}.txt"
with open(input_file_path, "r") as input_file, open(output_file_path, "w") as output_file:
    for line in input_file:
        output_b = convert_content2(line, mappings)
        output_file.write(output_b)

def convert_content(content, mappings):
    """Converts content using mappings."""
    output_content = ""
    index = 0
    while index < len(content):
        if content[index] == '\n':
            output_content += "1.27 "
            index += 1
            continue

        found = False
        for length in range(4, 0, -1):
            if index + length <= len(content):
                substr = content[index:index + length]
                if substr in mappings[length - 1]:
                    output_content += f"\{length}.{mappings[length - 1][substr]} "
                    index += length
                    found = True
                    break
        if not found:
            output_content += "0 "
            index += 1
    return output_content

def convert_content2(content, mappings):
    """Converts content using mappings."""
    output_b = ""
    index = 0
    while index < len(content):
        if content[index] == '\n':
            output_b += "27 "
            index += 1
            continue

        found = False
        for length in range(4, 0, -1):
            if index + length <= len(content):
                substr = content[index:index + length]
                if substr in mappings[length - 1]:
                    #output_b += f"\{length}.{mappings[length - 1][substr]} "
                    #output_b += f"\{100**\{length - 1} + (mappings[\{length - 1\}]\} "
                    if((length - 1) == 0):
                        output_b += f"\{mappings[\{length - 1\}][substr]} "

```

```
if((length- 1) == 1):
    output_b += f"100**({length- 1} + int(mappings[{length-1}][substr])) "
if((length- 1) == 2):
    output_b += f"100**({length- 1} + 100**({length- 2} + int(mappings[{length-1}][substr])) "
if((length- 1) == 3):
    output_b += f"100**({length- 1} + 100**({length- 2} + 100**({length- 3} +
int(mappings[{length-1}][substr])) "
index += length
found = True
break
if not found:
    output_b += "0 "
    index += 1
return output_b
```

```
#from PIL import Image
```

```
def save_image_as_pdf0(image, file_path):
    """
    Saves an image as a PDF file at the specified path.
```

Args:

image (PIL.Image): The image to save as a PDF.
file_path (str): The full path where the PDF should be saved.
"""

```
# Ensure the file path ends with '.pdf'
if not file_path.lower().endswith('.pdf'):
    file_path += '.pdf'
```

Save the image as a PDF

```
try:
    image.save(file_path, "PDF", resolution=100.0)
    print(f"Image successfully saved as PDF at {file_path}")
except Exception as e:
    print(f"Failed to save image as PDF. Error: {e}")
```

```
#####
#####
```

```
def request_user_input(prompt, input_type=int):
    try:
        value = input_type(input(prompt))
        if input_type is int and value <= 0:
            print("Please enter a positive integer.")
            return request_user_input(prompt, input_type)
        return value
    except ValueError:
        print(f"Invalid input. Please enter a valid {input_type.__name__}.")
        return request_user_input(prompt, input_type)
```

```

def generate_classical_circuit_from_line(line, filename):
    # Only take the part of the line after the colon if it exists
    gates = line.split(':')[1].strip().split(',')
    d = schemdraw.Drawing()
    gates = line.strip().split(',')
    last_element = None
    last_output = None

    for i, gate in enumerate(gates):
        gate = gate.strip()
        if gate == 'AND':
            element = logic.And(inputs=2, label=f'AND{i}')
        elif gate == 'OR':
            element = logic.Or(inputs=2, label=f'OR{i}')
        elif gate == 'NOT':
            element = logic.Not(label=f'NOT{i}')
        elif gate == 'NAND':
            element = logic.Nand(inputs=2, label=f'NAND{i}')
        elif gate == 'NOR':
            element = logic.Nor(inputs=2, label=f'NOR{i}')
        elif gate == 'XOR':
            element = logic.Xor(inputs=2, label=f'XOR{i}')
        elif gate == 'XNOR':
            element = logic.Xnor(inputs=2, label=f'XNOR{i}')
        else:
            print(f"Unrecognized gate: {gate}")
            continue

        # Add the element to the drawing with the correct position
        if last_element is not None:
            element.at(last_output)
            #d += element.at(last_element.anchors['out'])

        else:
            d += element

        #last_output = element.anchors['out']

        # Update last_element to the current one
        #####last_element = d.elements[-1] # Get the reference to the actual added element
        last_output = d.elements[-1].anchors['out']

    d.save(filename)
    return d

```

```

# Function to prompt the user for the number of cells (gate layers)
def get_user_input(prompt, input_type=int):
    try:

```

```

value = input_type(input(prompt))
if input_type is int and value <= 0:
    print("Please enter a positive integer.")
    return get_user_input(prompt, input_type)
return value
except ValueError:
    print(f"Invalid input. Please enter a valid {input_type.__name__}.")
    return get_user_input(prompt, input_type)

def generate_circuit_from_line(line, num_qubits):
    qc = QuantumCircuit(num_qubits)
    commands = line.strip().split(' ')[2:] # Skip the 'Circuit X:' part
    for cmd in commands:
        gate, args = cmd.split('(')
        args = args.strip(')').split(',')
        if gate in ['x', 'y', 'z', 'h', 's', 'sdg', 't', 'tdg']:
            getattr(qc, gate)(int(args[0]))
        elif gate in ['rx', 'ry', 'rz']:
            if 'pi/2' in args[0]:
                angle = np.pi / 2
            else:
                angle = float(args[0]) # Assuming other angles are directly specified
            getattr(qc, gate)(angle, int(args[1]))
        elif gate == 'cx':
            getattr(qc, gate)(int(args[0]), int(args[1]))
    return qc

def generate_cmp():
    # Create the 'instructionSet.txt' file
    name_m = input("Enter the name for your command matrix: ")
    with open(name_m, 'w') as f:
        # Lines 1 to 13107: string_var# = input('Enter string_var: ')
        for i in range(13107):
            f.write(f"string_var{i} = input('Enter string_var: ')\n")
        # Lines 13108 to 26214: var_given# = input('Enter the variable name to be used: ')
        for i in range(13107):
            f.write(f"var_given{i} = input('Enter the variable name to be used: ')\n")
        # Lines 26215 to 39321: var_glob# = globals()[var_given#]
        for i in range(13107):
            f.write(f"var_glob{i} = globals().get(var_given{i}, 'Variable not found') # Error handling: 'Variable not found'\n")
        # Lines 39322 to 52428: exec(var_glob#)
        for i in range(13107):
            f.write(f"if isinstance(var_glob{i}, str): exec(var_glob{i}) # Error handling: Execute only if it's a string\n")

```

```
# Lines 52429 to 65535: eval(var_glob#)
for i in range(13107):
    f.write(f"if isinstance(var_glob[i], str): result = eval(var_glob[i]) # Error handling: Evaluate only if it's
a string\n")

# Line 65536: Hyperlink
f.write("webbrowser.open('https://www.openai.com')")

# Note: For logging, you can add a line to write the executed or evaluated command to a log file.
# Note: For user authentication, you can add a line to check user credentials before executing or
evaluating a command.

def text_editor():
    print("Simple Text Editor")
    file_path = input("Enter the path of the text file to edit (or a new file name to create): ")

    # Try to open the file and read its contents into 'lines'
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()
        lines = [line.rstrip() for line in lines] # Remove newline characters
    except FileNotFoundError:
        print("File not found. Starting with an empty file.")
        lines = []

    while True:
        print("\n1. Add text")
        print("2. View text")
        print("3. Update a line")
        print("4. Delete a line")
        print("5. Save changes")
        print("6. Exit without saving")
        choice = input("Choose an option: ")

        if choice == '1':
            text = input("Enter your text: ")
            lines.append(text)
            print("Text added successfully.")

        elif choice == '2':
            if lines:
                print("\nCurrent Text:")
                for i, line in enumerate(lines, 1):
                    print(f"{i}. {line}")
            else:
                print("The text is empty.")

        elif choice == '3':
            line_num = int(input("Enter the line number to update: "))- 1
            if 0 <= line_num < len(lines):
                new_text = input("Enter the new text: ")
                lines[line_num] = new_text

        elif choice == '5':
            with open(file_path, 'w') as file:
                file.writelines(lines)
            print("Changes saved successfully.")

        elif choice == '6':
            break

    if choice != '6':
        print("Exiting the text editor.")
```

```

        print("Line updated successfully.")
    else:
        print("Invalid line number.")
    elif choice == '4':
        line_num = int(input("Enter the line number to delete: "))-1
        if 0 <= line_num < len(lines):
            del lines[line_num]
            print("Line deleted successfully.")
        else:
            print("Invalid line number.")
    elif choice == '5':
        with open(file_path, 'w') as file:
            for line in lines:
                file.write(f'{line}\n') # Add newline characters when writing
        print("Changes saved successfully.")
        break # Exit after saving
    elif choice == '6':
        print("Exiting without saving changes.")
        break
    else:
        print("Invalid choice. Please try again.")

```

```

def img_generator():
    print("random() : ", random.random())
    master = Tk()
    master.attributes('-fullscreen', True)
    #a = 250
    #b = 200
    print("Welcome\n")
    print("Tip: side width should be a factor of the image width, the same goes for the image height in relation to the side height of each pixel.")
    change1 = input("Enter side width of image block: ")
    change = int(change1)
    change2 = input("Enter side height of image block: ")
    change_h = int(change2)
    #a = 1880
    #b = 1050
    a1 = input("Enter width of image: ")
    a = int(a1)
    b1 = input ("Enter height of image: ")
    b = int(b1)
    pin_p1 = a/change
    pin_p2 = b/change_h
    w = Canvas(master, width= a, height= b)

#c = ["purple", "green", "gold", "red", "yellow", "orange", "pink", "brown", "cyan", "lime", "teal",
"magenta"]
c = []

```

```

while True:
    app_end = input("Enter colour Hex with # or colour name: ")
    c.append(app_end)

    conti_ = input("Enter 1 to add another colour, 0 to move on: ")
    if conti_ != '1': # Breaks the loop if input is not '1'
        break

    xc = 0
    zerox = 0
    zeroy = 0
    p = 1
    range_for = int((a/change)*(b/change_h))
    name = 1
    cells = ((a//change)*(b//change_h))
    upper = cells - 1
    nbr_comb = math.pow(len(c),cells)
    files = int(nbr_comb)
    #print(len(c))
    rown = 0
    img_fn_prefix = input("Enter Image filename prefix (omit the file extension name): ")
    file_count = 0
    for t in range(0,files):
        file_count = file_count + 1
        #if(file_count == 121):
        #    break
        switch = 1
        sw = 0
        #for x in range(range_for):
        #for x in range (0,cells):
        x = 0
        col = cells - 1
        #print(rown)
        while(x < cells):

            #f = open('%s.ps' % name, 'wb')
            #f.close
            if(switch == 1):
                row = 1
                nxleft = 0
                nxright = change
                nyleft = 0
                nyright = change_h
                zerox = 0
                zeroy = 0
            c_length = len(c)
            switch = 0
            #ran = random.randint(0,c_length- 1)

            rdiv = math.pow(len(c),col)
            cell = (rown/rdiv) % (len(c))

```

```

celled = int(cell)
print(celled)

#w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = "black", width = 0)
w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = "black", width = 0)
if(x <= cells):
    col = col- 1

w.grid(row = zeroy, column = zerox + change)
if(p >= pin_p1 and p%pin_p1 == 0):
    zeroy = zeroy + change_h
    zerox -=change
    #zerox = 0
    nyleft = change
    nxright = 0
    nyleft = nyleft + change_h
    nyright = nyright + change_h
    zerox = zerox + change
    p = p + 1
    #xc + 1
    nxright = nxright + change
    if(xc == 3):
        xc = 0
        x = x + 1
    rown = rown + 1
    ce = str(name)
    w.update()
    #w.postscript(file = ce + ".ps", colormode='color')
    w.postscript(file=img_fn_prefix + ce + ".ps", colormode='color', x=0, y=0, width=a, height=b)
    name = name + 1

def compile_image_file():
    nof = input("Input number of files: ")
    nof_ = int(nof)
    ins_x = input("Enter first file id: ")
    x = int(ins_x)
    donu = input("Do you have more than one file (y [Yes], n [No]? ")
    if(donu == 'y'):
        ins_y = input("Enter last file id: ")
        y = int(ins_y)
    ins = input("Enter path of files: ")

#def convert_to_png(path):
pre = input("Enter filename-prefix: ")
if(nof_ > 1):
    for i in range(x,y+1):
        ixy = str(i)
        #pre = input("Enter filename-prefix: ")
        path=ins + "\\\" + pre + ixy + ".ps"

```

```

img = Image.open(path)
img.save(pre + ins_x + ".png")
if(nof_ == 1):
    #pre = input("Enter filename-prefix: ")
    path = ins + "\\\" + pre + ins_x + ".ps"
    img = Image.open(path)
    img.save(pre + ins_x + ".png")

print("Done")

#master.mainloop()

def programming_engine():
    #####import pywebbrowser
    #create a custom event type
    MY_CUSTOM_EVENT = pygame.USEREVENT + 1

    # Initialize pygame
    pygame.init()

    # Initialize joystick module
    pygame.joystick.init()

    # Check if any joysticks are connected
    if pygame.joystick.get_count() > 0:
        # Get the first joystick
        joystick = pygame.joystick.Joystick(0)
        # Initialize the joystick
        joystick.init()

    # Set screen size
    #enter = int(input("Enter the dimension--> options 422, and multiples of 422"))
    enter = 422
    screen_width = enter
    screen_height = enter

bls = 52
screen = pygame.display.set_mode((screen_width, screen_height))
#screen = pygame.display.set_mode((screen_width, screen_height), pygame.RESIZABLE)
#####
#####Create Commands
bus_ = 255
amount_ = (bus_ + 1) * (bus_ + 1)
line_number_ = 0

create_f = open("Command_Template.txt", "w")
while(line_number_ < amount_):

```

```

#i = log10(line_number)
create_f.write("print(\"[Empty Command Slot] (Change using a text-editor to Update this slot in\n"
Command_Template.txt\") #remembering to rename Command_Template.txt\\n\")\n")
line_number_ = line_number_ + 1

create_f.close()

#LoadCommands
xin = 0
filnam = "Command_Template.txt"
with open(filnam, 'r') as file:
    commands = file.readlines()
    xin = xin + 1

star = [0]

for i, command in enumerate(commands):
    commands[i] = command.strip()
    #cmd = command.split(",")
    star.append(command)
del star[0]

#####
#####

#LoadCoordinates
xin2 = 0
with open('Coordinates_Python.txt', 'r') as file2:
    commands2 = file2.readlines()
    xin2 = xin2 + 1

starsx = [0]
starsy = [0]

for i2, command2 in enumerate(commands2):
    commands2[i2] = command2.strip()
    cmd2 = command2.split(",")
    starsx.append(int(cmd2[1]))
    varys = int(cmd2[2].strip())
    starsy.append(varys)

del starsx[0]
del starsy[0]

#####
for i3, commandy in starsy:
    new = starsy[i3].strip()

```

```
starsy[i3] = int(new)
"""

print(starsx)
print(starsy)
#####
# Load images
block_images = []
for i in range(256):
    block_images.append(pygame.image.load(f"block{i+1}.png"))

# Maze layout
maze_layout = [
    [0, 1, 2, 3],
    [4, 5, 6, 7],
    [8, 9, 10, 11],
    [12, 13, 14, 15]
]

#Function to check if a position is inside the maze
def is_inside_maze(x, y, maze_width, maze_height, bls):
    return 0 <= x < maze_width * bls and 0 <= y < maze_height * bls

# Load player images
player_image = pygame.image.load("player.png")
player_image2 = pygame.image.load("player2.png")

#scaled_block_images = [pygame.transform.scale(img, (int(img.get_width() * scale_factor),
int(img.get_height() * scale_factor))) for img in block_images]
#scaled_player_image = pygame.transform.scale(player_image, (int(player_image.get_width() *
scale_factor), int(player_image.get_height() * scale_factor)))

# Set player position
player_x = 0
player_y = 0
con_program = int(input("Continue? 1 [yes], 0 [no]: "))
if(con_program == 1):
    print("Ok ...")
if(con_program == 0):
    print("Exiting ...")
    exit()
if(con_program != 0 and con_program != 1):
    print("Error")
    exit()
# Set initial input type
```

```
typei = "k"
print("\n\nIn Local Command Mode\n\n")
print("Switch Command Matrix [s],\n mouse [m],\nkeyboard arrows [k],\nD-Pad (Xbox One Controller)
[d]\nPress [SPACEBAR] to call a command from Commands_Template.txt (Commands can be changed using
a text-editor) ... ")
print("For mode 'd' A (sub-matrix start region) Then A (sub-matrix end region). Press start for help
(Secure Internet Access Required).")
# some input variable
input_variable = 0
# add the custom event to the event queue
pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": input_variable}))
#etch loop
etch = 0

#maze width
maze_width = 16
maze_height = 16
# Main game loop
running = True
#new screen size settings
##screen_width, screen_height = pygame.display.get_surface().get_size()
scale_factor_x = screen_width / maze_width
#scale_factor_y = screen_height / maze_height
#scale_factor = min(scale_factor_x, scale_factor_y)
scale_factor = scale_factor_x

bls = int(bls * 0.5)

scaled_block_images = []

# Before the main loop
is_selecting_submatrix = False
submatrix_start_pos = None
selected_command = None
execute_command_prompt = False

for ni in range(256):
    s_i = pygame.transform.scale(block_images[ni], (bls, bls))
    block_images[ni] = s_i

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
#####
        elif event.type == pygame.KEYDOWN:
            # Check for key press event
            if event.key == pygame.K_m and typei != "m":
```

```

# Update input type
typei = "m"
elif event.key == pygame.K_k and typei != "k":
    # Update input type
    typei = "k"
elif event.key == pygame.K_s and typei != "s":
    # Update input type
    typei = "s"
elif event.key == pygame.K_d and typei != "d":
    # Update input type
    typei = "d"
elif event.type == pygame.KEYDOWN:
    if event.key == pygame.K_SPACE:
        #update input_type
        typei = "n"
        #check for status
        enter_cma = int(input("Enter command? 1[yes], 0 [no]: "))
        """
        # user input variable
        if(enter_cma == 0):
            typei = "k"
            player_x = 0
            player_y = 0
            etch = 1
        """
        if(enter_cma == 0):
            typei = "k"
            player_x = 0
            player_y = 0
        if(enter_cma == 1):
            input_variable = int(input("Enter command number ID (0 to 65535): "))
        # add the custom event to the event queue
        pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": input_variable}))
        #pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": cma}))
    """
    elif event.type == pygame.MOUSEBUTTONDOWN:
        # Get mouse position
        mouse_x, mouse_y = pygame.mouse.get_pos()

        # Check if the clicked position is inside the maze
        if is_inside_maze(mouse_x, mouse_y, len(maze_layout[0]), len(maze_layout), bls):
            # Calculate the clicked cell
            cell_x = mouse_x // bls
            cell_y = mouse_y // bls

            # Execute the desired action (e.g., print a message with the image ID)
            print(f"Hello, world! Image ID: {maze_layout[cell_y][cell_x]}")
    """

```

```

# check for custom event type
for event in pygame.event.get(MY_CUSTOM_EVENT):
    # access the input variable from the event object
    cma = event.input_variable
    # do something with the input variable
    #pass...
    if(typei == 'n'):

        #enter_cma = int(input("Enter command? 1[yes], 0 [no]: "))
        if(enter_cma == 1 and enter_cma != 0):
            player_x = 0
            player_y = 0

            # Clear screen
            screen.fill((255, 255, 255))
            # Update display
            #pygame.display.update()

        #cma = int(input("Enter command number ID (0 to 65535): "))
        try:
            exec(star[cma], globals())
        except:
            print("Try again, there was an error")
        #update command index pointer
        bus = 255 + 1
        if(cma <= 255):
            cmd_index = 0
            command_point = cma
        elif(cma > 255):
            cmd_index = cma//bus
            command_point = cma
        for tip in range(0, cmd_index):
            command_point = command_point- (bus)
        #print(command_point)

        prompt = "Executing Command " + str(cma) + " from index: " + str(cmd_index)
        print(prompt)
        #print(starsx)

        player_x = starsy[cmd_index]
        player_y = starsx[cmd_index]
        command_point_x = starsy[command_point]
        command_point_y = starsx[command_point]

        # Render player image at new position--> screen.blit(player_image, (player_x, player_y))

        #####

```

```

# Draw blocks
for y in range(int(screen_height/bls)):
    for x in range(int(screen_width /bls)):
        relative_x = x - player_x
        relative_y = y - player_y
        screen.blit(block_images[relative_x + relative_y * int(screen_width/bls)], (x * bls, y * bls))
        #screen.blit(scaled_block_images[relative_x + relative_y * int(screen_width // (bls *
scale_factor))], (x * bls * scale_factor, y * bls * scale_factor))

if(typei == 'n'):
    #Draw2 player
    screen.blit(player_image2, (player_x * bls, player_y * bls))
    #Draw Command pointer
    screen.blit(player_image, (command_point_x * bls, command_point_y * bls))
    #screen.blit(scaled_player_image, (player_x * bls * scale_factor, player_y * bls *
scale_factor))

    # Update display
    pygame.display.update()

#cm = int(input("Enter command number ID (0 to 65535):"))

#####
"""

if(typei == 'd'):
    # Check for Xbox controller input
    if pygame.joystick.get_count() > 0:
        joystick = pygame.joystick.Joystick(0)
        joystick.init()
    # Check for D-Pad input
    dpad = joystick.get_hat(0)
    if dpad == (1, 0):
        player_x += 1
    elif dpad == (-1, 0):
        player_x-= 1
    elif dpad == (0, 1):
        player_y-= 1
    elif dpad == (0,-1):
        player_y += 1
    if player_x < 0:
        player_x = 0
    if player_x >= int(screen_width / bls):
        player_x = int(screen_width / bls)- 1
    if player_y < 0:
        player_y = 0
    if player_y >= int(screen_height / bls):
        player_y = int(screen_height / bls)- 1
"""

```

```

# Assuming your commands are laid out in a grid with a known width (e.g., 16 for a 16x16 grid)
comp = 16 # Adjust this based on your actual layout
copmy = 256
# Inside the main loop, in the 'if(typei == 'd')):' block
if(typei == 'd'):
    # Check for Xbox controller input
    if pygame.joystick.get_count() > 0:
        joystick = pygame.joystick.Joystick(0)
        joystick.init()
        # Check for D-Pad input
        dpad = joystick.get_hat(0)
        if dpad == (1, 0):
            player_x += 1
        elif dpad == (-1, 0):
            player_x -= 1
        elif dpad == (0, 1):
            player_y -= 1
        elif dpad == (0,-1):
            player_y += 1
        if player_x < 0:
            player_x = 0
        if player_x >= int(screen_width / bls):
            player_x = int(screen_width / bls)- 1
        if player_y < 0:
            player_y = 0
        if player_y >= int(screen_height / bls):
            player_y = int(screen_height / bls)- 1

    # Handling button presses for A, X, B, Y
    buttons = joystick.get_numbuttons()
    for i in range(buttons):
        if joystick.get_button(i):
            if i == 0: # A button
                if not is_selecting_submatrix:
                    is_selecting_submatrix = True
                    submatrix_start_pos = (player_x, player_y)
                    print("Sub-matrix selection started at position:", submatrix_start_pos)
            else:
                is_selecting_submatrix = False
                print("Sub-matrix selected:", submatrix_start_pos, "to", (player_x, player_y))

    # Assuming index 7 represents the Start button; adjust if necessary
    elif i == 7: # Start button
        print("Start button pressed. Executing command 65535 to access documentation (Secure Internet Access Required).")
        # Assuming command 65535 takes the user to the documentation
        exec(star[65535])
        # Optionally, set a flag or take additional action as needed

```

```

if(typei == 's'):

#####
#LoadCommands
xin = 0
filnam = input("Enter name of Command Matrix file, 'filename.txt': ")
quet = int(input("Enter 1 if file already exists [Read mode], Else Enter 2 [Write mode], choose
carefully: "))

bus = 255
amount = (bus + 1) * (bus + 1)
line_number = 0
#ai = log10(amount)
#if(quet == 2):
#    create_f = open(filnam, "w")
#    while(line_number < amount):
#        #i = log10(line_number)
#        create_f.write("print(\"[Empty Command Slot] Change using a text-editor to Update this slot
in Command_Template.txt after renaming it accordingly.\")\n")
#        line_number = line_number + 1

#    create_f.close()

with open(filnam, 'r') as file:
    commands = file.readlines()
    xin = xin + 1

star = [0]

for i, command in enumerate(commands):
    commands[i] = command.strip()
    #cmd = command.split(",")
    star.append(command)
del star[0]

#reset state of matrix
typei = "k"
player_x = 0
player_y = 0

if(typei == 'k'):
    # Get user keyboard input
    keys = pygame.key.get_pressed()

    # Update move flag based on keyboard input
    if keys[pygame.K_LEFT] or keys[pygame.K_RIGHT] or keys[pygame.K_UP] or
    keys[pygame.K_DOWN]:
        move = True

```

```

else:
    move = False

# Update player position based on keyboard input
if(move):
    if keys[pygame.K_LEFT]:
        player_x-= 1
    if keys[pygame.K_RIGHT]:
        player_x += 1
    if keys[pygame.K_UP]:
        player_y-= 1
    if keys[pygame.K_DOWN]:
        player_y += 1
    if player_x < 0:
        player_x = 0
    if player_x >= int(screen_width / bls):
        player_x = int(screen_width / bls)- 1
    if player_y < 0:
        player_y = 0
    if player_y >= int(screen_height / bls):
        player_y = int(screen_height / bls)- 1

if(typei == 'm'):
    # Get mouse position
    mouse_x, mouse_y = pygame.mouse.get_pos()

    # Update player position based on mouse position
    player_x = mouse_x // bls
    player_y = mouse_y // bls

# Clear screen
screen.fill((255, 255, 255))

# Draw blocks
if(typei == 'k' or typei == 'm' or typei == 'd'):
    for y in range(int(screen_height/bls)):
        for x in range(int(screen_width/bls)):
            relative_x = x- player_x
            relative_y = y- player_y
            screen.blit(block_images[relative_x + relative_y * int(screen_width // (bls))], (x * bls, y * bls))
    if(typei == 'k' or typei == 'm' or typei == 'd'):
        screen.blit(player_image, (player_x * bls, player_y * bls))

if(typei == 'k' or typei == 'm' or typei == 'd'):
    # Update display
    pygame.display.update()

```

```

# Quit pygame
pygame.quit()

def help_print():
    print("\nWelcome to Amalec, here are your options ...\\n\\n")
    print("\\n\\tProof is the absence of doubt. Guess not move. By way of the unbreakable Law. \\n\\tThis is the
solution. That is which solves without the cause of one problem exclusively or problems.\\n\\nWritten by Mr.
Dominic Alexander Cooper")
    print("\\n\\nEnter 0: Image generator (Required for the use of 2)")
    print("Enter 1: Image Compiler")
    print("Enter 2: Programming Engine")
    print("Enter 3: Cosmology (Useable with 0, 1 and 2)")
    print("Enter 4: Sequential Language Generation (Warning, you must filter out the profane. Also a user .txt
file must exist within the working directory that lists (one character per line) Unicode characters: ")
    print("Enter 5: Text Editor")
    print("Enter 6: Generate Command Matrix Plugin for 2")
    print("Enter 7: Exit Program")
    print("Enter 8: 3D CAD CAM Engine")
    print("Enter 9: Quantum Circuit Qiskit WorkBench")
    print("Enter 10: Classical Circuit SchemDraw WorkBench")
    print("Enter 11: Cosmological Simulation by Text")
    print("Enter 12: Large Language Model")
    print("Enter 13: Data Script by Custom Structures")
    print("Enter 14: Generate Binary Unique Strings to a custom.txt file")
    print("Enter 16: outb file to binaryout file (Extension of 12)")
    print("Enter 17: Text Database")
    print("Enter 18: Proximal Prime Numbers")
    print("Enter 19: Configurations for Mode 2")
    print("Enter 21: Proximal Primes Animator")
    print("Enter 22: Manual Turing Complete Finite State Machine")
    print("Enter 23: Automatic Turing Complete Finite State Machine")
    print("Enter 24: Python Enabled Command Line Interface")
    print("Enter 25: Automatic Control Engineering")
    print("Enter 26: Automated Design and Technology")
    print("Enter 27: Construct your own Linguistic Language")
    print("Enter 28: Convert Text Encoding (From 12) to a Convergent Image")
    print("Enter 29: Mathematics Lab")
    print("Enter 30: Multi-Language Character Mapper")

while(True):

    entrance = int(input("\\n\\n\\tEnter your mode of operation (15 for available options): "))

    if(entrance == 0):
        img_generator()
    if(entrance == 1):
        compile_image_file()
    if(entrance == 2):
        programming_engine()

```

```

if(entrance == 3):
    current_col = 0
    current_row = 0
    resized_images = [] # Global variable to store resized images

    # Rest of your code...
    # Global variable for the image selection frame
    image_selection_frame = None
    selected_image = None # Global variable to hold the currently selected image
    # Global variable to keep track of the current mode
    current_mode = "draw_char" # Possible values: "draw_char", "image_mode"

def toggle_mode():
    global current_mode
    if current_mode == "draw_char":
        current_mode = "image_mode"
        canvas.bind("<Button-1>", on_canvas_click_for_image) # Bind the image drawing function
    else:
        current_mode = "draw_char"
        canvas.bind("<Button-1>", lambda event: draw_char(event.y // square_size, event.x // square_size)) # Bind draw_char function

def select_image(img):
    global selected_image
    selected_image = img
    # You can add additional logic here, e.g., updating the UI to indicate the selected image

def resize_and_add_image(file_path, size):
    try:
        with Image.open(file_path) as img:
            # Resize and add to the list
            #resized_img = img.resize(size, Image.Resampling.LANCZOS) # For Pillow versions 8.0.0 and later
            #resized_img = img.resize(size, Image.LANCZOS) # For older versions of Pillow
            resized_img = img.resize(size, Image.LANCZOS)
            resized_images.append(resized_img)
            #update_image_selection_area()
    except Exception as e:
        print(f"Error loading image: {e}")

def on_canvas_click_for_image(event):
    col = event.x // square_size
    row = event.y // square_size
    draw_image_on_canvas(row, col)

def draw_image_on_canvas(row, col):
    global selected_image_path, square_size

```

```

if selected_image_path:
    try:
        with Image.open(selected_image_path) as img:
            resized_image = img.resize((square_size, square_size), Image.LANCZOS)
            tk_image = ImageTk.PhotoImage(resized_image)

            x = col * square_size
            y = row * square_size
            canvas.create_image(x, y, image=tk_image, anchor='nw')

            if not hasattr(canvas, 'images'):
                canvas.images = []
            canvas.images.append(tk_image) # Keep a reference
    except Exception as e:
        print(f"Error loading image: {e}")

    ...
def update_image_selection_area():
    global image_selection_frame

    # Clear existing buttons in the frame
    for widget in image_selection_frame.winfo_children():
        widget.destroy()

    # Create buttons for each image
    for img in resized_images:
        tk_image = ImageTk.PhotoImage(img)
        btn = tk.Button(image_selection_frame, image=tk_image, command=lambda img=img:
select_image(img))
        btn.image = tk_image # Keep a reference to avoid garbage collection
        btn.pack(side='left')
    ...

def select_and_add_images():
    global selected_image_path

    # Ask the user to enter the file path
    file_path = input("Enter the image file path: ")
    selected_image_path = file_path # Store the path of the selected image

    # Define a target size for the images
    wid = int(input("Target Width: "))
    hei = int(input("Target Height: "))
    target_size = (wid, hei) # You can change this size as needed

    if file_path:
        # Resize the image and add it to the canvas
        resize_and_add_image(file_path, target_size)

```

```
def resize_and_add_image(file_path, size):
    try:
        with Image.open(file_path) as img:
            # Resize and add to the list
            resized_img = img.resize(size, Image.Resampling.LANCZOS) # For Pillow versions 8.0.0 and later
            resized_images.append(resized_img)
            #update_image_selection_area()
    except Exception as e:
        print(f"Error loading image: {e}")

def select_and_resize_image():
    # Ask the user to select an image file
    file_path = filedialog.askopenfilename(filetypes=[("PNG files", "*.png")])
    if not file_path:
        return # User cancelled the dialog

    # Prompt the user for the new size
    new_size = simpledialog.askstring("Resize Image", "Enter new size (width,height):")
    if not new_size:
        return # User cancelled the dialog

    try:
        # Parse the size input and resize the image
        width, height = map(int, new_size.split(','))
        resize_image(file_path, (width, height))
    except Exception as e:
        tk.messagebox.showerror("Error", f"An error occurred: {e}")

def resize_image(input_path, size):
    try:
        # Construct a new file name based on the original path
        dir_name, file_name = os.path.split(input_path)
        name, ext = os.path.splitext(file_name)
        output_path = os.path.join(dir_name, f"{name}_resized{ext}")

        # Open, resize, and save the image
        with Image.open(input_path) as img:
            # Use Image.Resampling.LANCZOS for Pillow versions 8.0.0 and later
            img = img.resize(size, Image.Resampling.LANCZOS)
            # For older versions of Pillow, use Image.LANCZOS
            # img = img.resize(size, Image.LANCZOS)
            img.save(output_path)

        tk.messagebox.showinfo("Success", f"Image saved successfully to {output_path}")
    except Exception as e:
        tk.messagebox.showerror("Error", f"An error occurred: {e}")
```

```

def get_random_color():
    r = lambda: random.randint(0,255)
    return '#%02X%02X%02X' % (r(),r(),r())

def get_random_char():
    random_int = random.randint(0x0021, 0x007E)
    return chr(random_int)

def draw_grid():
    for i in range(grid_size):
        for j in range(grid_size):
            color = get_random_color()
            square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size,
            (i+1)*square_size, fill=color)
            char = get_random_char()
            text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2,
            text=char, font=("Arial", 18), anchor="center")

def draw_grid_IDE():
    global col
    try:
        for i in range(grid_size):
            for j in range(grid_size):
                color = "blue"
                square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size,
                (i+1)*square_size, fill=color, outline = "white")
                if typing_mode:
                    # In the function where you create the canvas...
                    canvas.bind("<Key>", on_key_press)
                    canvas.focus_set()
    except:
        print("Submit an app mode ...")

def toggle_color_mode():
    """Toggles between manual color mode and random color mode"""
    global color_mode_var
    color_mode_var.set(not color_mode_var.get())

font_size = 9

def draw_char1(canvas, char, row, col, font_size, color):
    global cell_size, font_color, square_size, grid_size, last_drawn, note, char_note

    try:
        x = col * square_size + square_size / 2
        y = row * square_size + square_size / 2

```

```

print(f"Drawing char '{char}' at ({x}, {y})" ) # Debug print to check coordinates
    canvas.create_rectangle(col * square_size, row * square_size, (col+1) * square_size, (row+1) * square_size, fill=color)
        canvas.create_text(x, y, text=char, fill=font_color, font=('Calibri', int(font_size)))
        note = char_note.get()
        # Log the color and character info
        logging.info(f"Position:{x};{y},Xp:{x},Yp:{y},Character:{char},Square color:{color},Font color:{font_color},Note:{note}")
        # Log the color and character info in a text file
        if(note == ""):
            note = "emptyNote"
        with open("color_log.txt", "a", encoding='utf-8') as login:
            print(f"{x},{y},{char},{color},{font_color},{note}", file=login)

except Exception as e:
    print(f"Something Went Wrong... Error: {e}")

```

```

def draw_char(i=None, j=None):
    try:
        """Draws a character from input field in a specific square"""
        global char_count, IDE_mode, typing_mode, last_drawn, font_color
        if(mode == 'IDE'):
            IDE_mode = True
        # Check if last_drawn is defined, otherwise define it
        if 'last_drawn' not in globals():
            last_drawn = []
        # If i, j are not provided, calculate them based on char_count
        if i is None or j is None:
            i, j = divmod(char_count, grid_size)
            # Handle out-of-grid situations
            if i >= grid_size or j >= grid_size:
                print('Out of grid!')
                return
        # Generate a random color if not in typing mode, white otherwise
        ##color = get_random_color() if not typing_mode else "white"

        # Check color mode
        if color_mode_var.get():
            # Manual color mode
            # Show a color picker and get the chosen color for the square
            color = askcolor(title="Choose square color")[1]

            # Ask for the font color
            color_result = askcolor(title="Choose font color")
            if color_result is not None:

```

```

        font_color = color_result[1]
    else:
        # Handle the case when the user cancelled the color selection
        font_color = "#000000" # default to black, for example

else:
    # Random color mode
    color = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
    color1 = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
    font_color = color1 #"#000000"

square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size,
(i+1)*square_size, fill=color)
char = char_entry.get()[:1]
note = char_note.get()
text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2, text=char,
font=("Arial", font_size), fill=font_color, anchor="center")

...
# Log the color and character info
logging.info(f"Position:{i};{j},Xp:{i},Yp:{j},Character:{char},Square color:{color},Font
color:{font_color}'")
# Log the color and character info in a text file
with open("color_log.txt", "a") as login:
    print(f"{i},{j},{char},{color},{font_color}", file=login)
"""

# Log the color and character info
logging.info(f"Position:{i};{j},Xp:{i},Yp:{j},Character:{char},Square color:{color},Font
color:{font_color}',Note:{note}'")
# Log the color and character info in a text file
if(note == ""):
    note = "emptyNote"
with open("color_log.txt", "a", encoding='utf-8') as login:
    print(f"{i},{j},{char},{color},{font_color},{note}", file=login)

if len(char_entry.get()) > 0: # Check if there's more than one character
    char_entry.delete(0, 1) # Delete the first character

last_drawn.append((square, text))

if not IDE_mode:
    char_count = (char_count + 1) % (grid_size * grid_size)
    if char_count == 0: # If we've filled the canvas, clear it
        canvas.delete('all')

return square, text

```

```

except:
    print("Submit an app mode ...")

def adjust_grid_and_font():
    global grid_size, square_size, font_size, canvas_width, current_row, current_col

    n_g_s = simpledialog.askstring("Change Grid Size (Grid Width)", "Enter New Grid Size (+Integer):")
    if(n_g_s != ""):
        new_grid_size = n_g_s
    n_f_s = simpledialog.askstring("Change Font Size (Default = 9)", "Enter New Font Size (+Integer):")
    if(n_f_s != ""):
        new_font_size = n_f_s
    # Update the global variables
    if(new_grid_size==None and new_font_size==None):
        grid_size = 10
        font_size = 9
    if((not new_grid_size==None) and (not new_font_size==None)):
        try:
            grid_size = int(new_grid_size)
            square_size = canvas_width / grid_size
            font_size = int(new_font_size)
        except:
            print("positive integers please")

    """
    # Get new values from input fields
    new_grid_size = grid_size_entry.get()
    new_font_size = font_size_entry.get()

    # Update the global variables
    grid_size = int(new_grid_size)
    square_size = canvas_width // grid_size
    font_size = int(new_font_size)
    """

    # Redraw the grid
    refresh_canvas()
    qit = input("Are you in typing mode, or want to enter typing mode? 1 for (No), 2 for (yes): ")
    if(qit == '2'):
        current_row = 0
        current_col = 0
        show_typing_mode_menu()

def refresh_canvas():
    """Clears the canvas and resets the char_count"""
    global char_count
    canvas.delete('all')
    char_count = 0
    draw_grid_IDE()

```

```
last_drawn.clear()

def undo_last():
    """Undoes the last drawing operation"""
    if last_drawn:
        square, text = last_drawn.pop()
        canvas.delete(square)
        canvas.delete(text)

def update_canvas():
    if not paused.get():
        canvas.delete('all')
        draw_grid()
    root.after(8000, update_canvas)

def toggle_typing_mode():
    global typing_mode, canvas

    typing_mode = not typing_mode
    if typing_mode:
        canvas.focus_set() # Set focus to the canvas for keyboard input


def on_canvas_click(event):
    try:
        global mode, typing_mode
        if mode == 'IDE' and not typing_mode:
            j = event.x // square_size
            i = event.y // square_size
            draw_char(i, j)
        else:
            paused.set(not paused.get())
    except:
        print("Submit an app mode ...")

def on_key_press(event):
    global char_count, typing_mode
    if typing_mode:
        char_entry.delete(0, 'end') # Clear the entry box
        try:
            utf8_char = event.char.encode('utf-8').decode('utf-8')
            char_entry.insert(0, utf8_char) # Insert the typed character
        except UnicodeDecodeError:
            print("Non UTF-8 character detected")
            return
        draw_char() # Draw the character
        char_count += 1 # Increment the count

    # If char_count exceeds the total number of squares in the grid, reset it
```

```
if char_count >= grid_size ** 2:  
    char_count = 0  
  
def submit_mode():  
  
    global char_count, mode, typing_mode  
  
    mode = mode_entry.get().upper()  
  
    ...  
  
    if mode in ['editor']:  
  
        typing_mode = False # Start in clicking mode  
        mode_entry.delete(0, 'end') # Clear the input field  
        mode_label.pack_forget()  
        mode_entry.pack_forget()  
        submit_button.pack_forget()  
  
        draw_grid()  
        char_label.pack()  
        char_entry.pack()  
        char_button.pack()  
  
        ...  
  
        # Draw the initial grid  
        if(mode != 'IDE'):  
            print("A valid mode ...")  
  
        ...  
  
        if(mode == 'normal'):  
  
            typing_mode = False # Start in clicking mode  
            mode_entry.delete(0, 'end') # Clear the input field  
            mode_label.pack_forget()  
            mode_entry.pack_forget()  
            submit_button.pack_forget()  
  
            draw_grid()  
            create_menu_1()  
  
        if mode == 'normal':  
            # Schedule the first update  
            root.after(8000, update_canvas)  
        ...  
  
        if mode == 'IDE':
```

```
typing_mode = False # Start in clicking mode
mode_entry.delete(0, 'end') # Clear the input field
mode_label.pack_forget()
mode_entry.pack_forget()
submit_button.pack_forget()

char_count = 0
draw_grid_IDE()

control_frame.pack()
char_label.pack()
char_entry.pack()
#grid_size_label.pack()
#grid_size_entry.pack()
#font_size_label.pack()
#font_size_entry.pack()
#adjust_button.pack()
char_note_label.pack()
char_note.pack()
#refresh_button.pack(side="left")
#undo_button.pack(side="left")
#save_button = tk.Button(root, text='Save', command=save_canvas, bg="white", padx=5, pady=0)
#save_button.pack(side="left")
#save_Sbutton.pack(side="left")
#load_button.pack(side="left")
create_menu()

return mode
```

```
def save_canvas():
    global IC_value, xSwitch, ing
    xSwitch = 1
    ing = 1
    if(ing == 0):
        ing = 2
    ing = 1
    if(xSwitch == 1 and ing == 1):
        IC_value = 0
        ing = 0
    """Save the current state of the canvas to a .png file"""
    #filename = simpledialog.askstring("Save Canvas", "Enter filename:")
    filename = f"ImageCanvas{IC_value}_rename"

if filename: # Only save the canvas if a filename was entered
    # Get the bounding box of the canvas
    x = root.winfo_rootx() + canvas.winfo_x()
```

```

y = root.winfo_rooty() + canvas.winfo_y()
x1 = x + canvas.winfo_width()
y1 = y + canvas.winfo_height()
time.sleep(3)
# Grab the image, crop it to the bounding box, and save it
ImageGrab.grab().crop((x, y, x1, y1)).save(filename + ".png")

def save_session():
    """Save the current session to a file using pickle"""
    global last_drawn
    # Ask the user to enter a filename
    filename = simpledialog.askstring("Save Session", "Enter filename:")
    if filename: # Only save the session if a filename was entered
        session_data = [(canvas.coords(square), canvas.itemcget(square, "fill"),
                          canvas.coords(text), canvas.itemcget(text, "text"), canvas.itemcget(text, "fill"))
                         for square, text in last_drawn]
        with open(filename + ".pkl", "wb") as f:
            pickle.dump(session_data, f)

def load_session():
    """Load a saved session from a file using pickle"""
    global last_drawn
    # Ask the user to enter a filename
    filename = simpledialog.askstring("Load Session", "Enter filename:")
    if filename: # Only try to load a session if a filename was entered
        try:
            with open(filename + ".pkl", "rb") as f:
                session_data = pickle.load(f)
                # Clear the canvas and redraw all elements from the loaded session
                canvas.delete('all')
                draw_grid_IDE()
                last_drawn = []
                for square_coords, square_fill, text_coords, text_content, text_fill in session_data:
                    square = canvas.create_rectangle(*square_coords, fill=square_fill)
                    text = canvas.create_text(*text_coords, text=text_content, fill=text_fill, font=("Arial",
                        font_size), anchor="center")
                    last_drawn.append((square, text))
        except FileNotFoundError:
            print(f"No session named '{filename}' found.")

#def exit_app():
#    exit()

def show_about():
    about_window = tk.Toplevel(root)
    about_window.title("About")
    about_msg = "This is a program created to learn and experiment with Tkinter. In IDE mode, you can draw characters on a grid (By typing or copying and pasting then clicking the squares you want to draw"

```

each character on), adjust the grid and font size, save and load sessions, and more, like saving the grid as an image (For example). C(num) was written with the aid of Chat GPT. Enjoy!\n\nNote: the workflow ... IDE to color_log.txt (Compiled by compile#.exe or compile#.py);\nindex_1.txt contains fields of study or research and development and\nindex_2.txt contains fields of study or research and development also
...\\n\\nBertotools Digital"

```
tk.Message(about_window, text=about_msg, width=500).pack()
tk.Button(about_window, text="OK", command=about_window.destroy).pack()

def compileGOSmX():
    MAX_LINE_LENGTH = 256
    MAX_TOPICS = 256

    global index_1, index_2

    index_1 = [
        "Axiom",
        "Theorem",
        "Lemma",
        "Proposition",
        "Corollary",
        "Conjecture",
        "Proof",
        "Premise",
        "Conclusion",
        "Hypothesis",
        "Counterexample",
        "Direct Proof",
        "Indirect Proof",
        "Proof by Contradiction (Reductio ad absurdum)",
        "Proof by Induction",
        "Proof by Contrapositive",
        "Deductive Reasoning",
        "Inference",
        "Assumption",
        "Statement",
        "Postulate",
        "Proof by Exhaustion",
        "Syllogism",
        "Constructive Proof",
        "Non-Constructive Proof",
        "Trivial Proof",
        "Vacuous Proof",
        "Biconditional",
        "Condition",
        "Sufficiency",
        "Necessity",
        "Quantifier",
        "Universal Quantifier",
        "Existential Quantifier",
        "Bound Variable",
```

"Free Variable",
"Predicate",
"Propositional Logic",
"Modus Ponens",
"Modus Tollens",
"Discrete Mathematics",
"Set Theory",
"Function",
"Bijection",
"Injection",
"Surjection",
"Equivalence Relation",
"Partial Order",
"Total Order",
"Well-Order",
"Reflexivity",
"Symmetry",
"Transitivity",
"Antisymmetry",
"Completeness",
"Compactness",
"Connectedness",
"Convergence",
"Divergence",
"Limit",
"Sequence",
"Series",
"Monotonicity",
"Cauchy Sequence",
"Infinite Set",
"Finite Set",
"Cardinality",
"Countable Set",
"Uncountable Set",
"Subset",
"Superset",
"Intersection",
"Union",
"Empty Set",
"Power Set",
"Cartesian Product",
"Equivalence Class",
"Partition",
"Field",
"Ring",
"Group",
"Abelian Group",
"Non-abelian Group",
"Matrix",
"Vector Space",

"Linear Transformation",
"Eigenvalue",
"Eigenvector",
"Norm",
"Inner Product",
"Orthogonality",
"Basis",
"Dimension",
"Rank",
"Nullity",
"Determinant",
"Graph Theory",
"Vertex",
"Edge",
"Connectivity",
"Cycle",
"Path",
"Degree",
"Subgraph",
"Tree",
"Forest",
"Planar Graph",
"Bipartite Graph",
"Directed Graph (Digraph)",
"Eulerian Graph",
"Hamiltonian Graph",
"Adjacency Matrix",
"Incidence Matrix",
"Isomorphism",
"Homeomorphism",
"Topology",
"Open Set",
"Closed Set",
"Boundary",
"Compact Space",
"Hausdorff Space",
"Continuity",
"Differential",
"Derivative",
"Integral",
"Partial Derivative",
"Multivariable Calculus",
"Laplace Transform",
"Fourier Transform",
"Taylor Series",
"Maclaurin Series",
"Conic Sections",
"Hyperbola",
"Ellipse",
"Parabola",

"Asymptote",
"Limits at Infinity",
"Complex Number",
"Imaginary Unit",
"Real Number",
"Rational Number",
"Irrational Number",
"Prime Number",
"Composite Number",
"GCD (Greatest Common Divisor)",
"LCM (Least Common Multiple)",
"Permutation",
"Combination",
"Probability",
"Statistics",
"Expected Value",
"Variance",
"Standard Deviation",
"Normal Distribution",
"Poisson Distribution",
"Binomial Distribution",
"Hypothesis Testing",
"Regression",
"Correlation",
"Matrix Algebra",
"Linear Algebra",
"Vector Calculus",
"Optimization",
"Algorithm",
"Computational Complexity",
"Big O Notation",
"Pigeonhole Principle",
"Principle of Inclusion-Exclusion",
"Turing Machine",
"Computability",
"Unsolvability",
"Parity",
"Diophantine Equations",
"Cryptography",
"Fermat's Last Theorem",
"Pythagorean Theorem",
"Triangle Inequality",
"Trigonometric Functions",
"Trigonometric Identities",
"Polar Coordinates",
"Euler's Formula",
"Riemann Zeta Function",
"P vs NP Problem",
"NP-complete Problem",
"Stochastic Process",

"Markov Chain",
"Random Variable",
"Conditional Probability",
"Bayes' Theorem",
"Monte Carlo Method",
"Fractal",
"Chaos Theory",
"Game Theory",
"Nash Equilibrium",
"Zero-Sum Game",
"Non-Zero-Sum Game",
"Linear Programming",
"Nonlinear Programming",
"Quadratic Programming",
"Dynamic Programming",
"Integer Programming",
"Graph Coloring",
"Network Flow",
"Spanning Tree",
"Bellman-Ford Algorithm",
"Dijkstra's Algorithm",
"Kruskal's Algorithm",
"Prim's Algorithm",
"Floyd-Warshall Algorithm",
"Euler's Method",
"Runge-Kutta Method",
"Numerical Integration",
"Numerical Differentiation",
"Bisection Method",
"Newton's Method",
"Secant Method",
"Fixed Point Iteration",
"Linear Interpolation",
"Polynomial Interpolation",
"Lagrange Interpolation",
"Splines",
"Fourier Series",
"Laplace's Equation",
"Heat Equation",
"Wave Equation",
"Schrodinger Equation",
"Ordinary Differential Equation (ODE)",
"Partial Differential Equation (PDE)",
"Boundary Value Problem",
"Initial Value Problem",
"Green's Theorem",
"Stoke's Theorem",
"Divergence Theorem",
"Curl",
"Gradient",

"Divergence",
"Tensor",
"Manifold",
"Topological Space",
"Measure Theory",
"Lebesgue Integral",
"Borel Set",
"Hilbert Space",
"Banach Space",
"Category Theory",
"Functor",
"Natural Transformation",
"Sheaf",
"Homotopy",
"Homology",
"Cohomology",
"Galois Theory",
"Algebraic Geometry",
"Topological K-Theory",
"Knot Theory",
"Lattice Theory"

]

index_2 = [
 "Biochemistry",
 "Biophysics",
 "Molecular biology",
 "Genetics",
 "Immunology",
 "Cell biology",
 "Microbiology",
 "Neuroscience",
 "Pharmacology",
 "Bioinformatics",
 "Biotechnology",
 "Proteomics",
 "Genomics",
 "Structural biology",
 "Virology",
 "Systems biology",
 "Developmental biology",
 "Evolutionary biology",
 "Synthetic biology",
 "Metabolomics",
 "Epigenetics",
 "Tissue engineering",
 "Nanotechnology",
 "Materials science",
 "Quantum physics",
 "Condensed matter physics",

"Particle physics",
"Astrophysics",
"Cosmology",
"Optics",
"Atomic and molecular physics",
"Fluid mechanics",
"Thermodynamics",
"Environmental science",
"Climate science",
"Geology",
"Oceanography",
"Atmospheric science",
"Ecology",
"Conservation biology",
"Botany",
"Zoology",
"Entomology",
"Marine biology",
"Paleontology",
"Anthropology",
"Archaeology",
"Psychology",
"Cognitive science",
"Social psychology",
"Linguistics",
"Artificial intelligence",
"Machine learning",
"Computer vision",
"Natural language processing",
"Human-computer interaction",
"Robotics",
"Computer graphics",
"Data science",
"Mathematical modeling",
"Mathematical physics",
"Number theory",
"Algebraic geometry",
"Differential equations",
"Computational physics",
"Mathematical biology",
"Operations research",
"Biostatistics",
"Epidemiology",
"Cancer research",
"Diabetes research",
"Heart disease research",
"Infectious diseases research",
"Immunotherapy",
"Stem cell research",
"Gene therapy",

"Drug discovery",
"Precision medicine",
"Health informatics",
"Renewable energy",
"Energy storage",
"Sustainable materials",
"Environmental engineering",
"Water management",
"Transportation engineering",
"Civil engineering",
"Chemical engineering",
"Aerospace engineering",
"Biomedical engineering",
"Electrical engineering",
"Mechanical engineering",
"Robotics engineering",
"Quantum computing",
"Cryptography",
"Cybersecurity",
"Network engineering",
"Telecommunications",
"Human genetics",
"Forensic science",
"Space exploration and research",
"Planetary science",
"Astrobiology",
"Astrochemistry",
"Astrogeology",
"Astroinformatics",
"Exoplanet research",
"Stellar evolution",
"Galactic astronomy",
"Observational astronomy",
"Computational astrophysics",
"Quantum chemistry",
"Computational chemistry",
"Organic chemistry",
"Inorganic chemistry",
"Physical chemistry",
"Environmental chemistry",
"Analytical chemistry",
"Agricultural science",
"Food science",
"Nutritional science",
"Exercise physiology",
"Sports science",
"Biomechanics",
"Plant physiology",
"Plant genetics",
"Plant pathology",

"Soil science",
"Hydrology",
"Geochemistry",
"Geophysics",
"Geomorphology",
"Remote sensing",
"Geotechnical engineering",
"Petroleum engineering",
"Aerospace materials",
"Nanomaterials",
"Polymer science",
"Computational materials science",
"Photonics",
"Physical optics",
"Quantum optics",
"Neuroengineering",
"Brain imaging",
"Cognitive neuroscience",
"Neuroinformatics",
"Psychophysics",
"Developmental psychology",
"Personality psychology",
"Clinical psychology",
"Industrial-organizational psychology",
"Educational psychology",
"Psycholinguistics",
"Human genetics",
"Evolutionary genetics",
"Population genetics",
"Genetic engineering",
"Genetic counseling",
"Epigenomics",
"Cardiovascular research",
"Respiratory research",
"Gastroenterology research",
"Endocrinology research",
"Nephrology research",
"Hematology research",
"Ophthalmology research",
"Orthopedic research",
"Dermatology research",
"Veterinary medicine",
"Animal behavior",
"Conservation ecology",
"Wildlife biology",
"Environmental microbiology",
"Agricultural economics",
"Development economics",
"Behavioral economics",
"Econometrics",

"Financial mathematics",
"Operations management",
"Supply chain management",
"Industrial engineering",
"Human-computer interaction",
"Virtual reality",
"Augmented reality",
"Data mining",
"Text mining",
"Big data analytics",
"Computational linguistics",
"Quantum information science",
"Quantum cryptography",
"Biometrics",
"Information retrieval",
"Software engineering",
"Computer networks",
"Embedded systems",
"Human-robot interaction",
"Control systems",
"Biopharmaceuticals",
"Drug delivery systems",
"Clinical trials",
"Regenerative medicine",
"Agricultural biotechnology",
"Plant breeding",
"Animal breeding",
"Food technology",
"Sensory science",
"Poultry science",
"Aquaculture",
"Marine ecology",
"Limnology",
"Population ecology",
"Landscape ecology",
"Evolutionary ecology",
"Environmental toxicology",
"Environmental chemistry",
"Environmental microbiology",
"Ecotoxicology",
"Green chemistry",
"Space physics",
"Space weather",
"Astrostatistics",
"Computational fluid dynamics",
"Mathematical optimization",
"Operations research",
"Human genetics",
"Functional genomics",
"Molecular genetics",

```
"Cancer genetics",
"Psychiatric genetics",
"Population genomics",
"Bioengineering",
"Biomaterials",
"Biomechatronics",
"Cardiovascular engineering",
"Neural engineering",
"Rehabilitation engineering",
"Genetic engineering",
"Environmental engineering",
"Water resources engineering",
"Structural engineering",
"Robotics engineering",
"Quantum information theory",
"Quantum simulation",
"Quantum sensing",
"Geographical information systems (GIS)",
"Urban planning",
"Renewable energy systems",
"Solar cell technology",
"Wind energy research",
"Energy policy and economics",
"Computational neuroscience",
"Neurobiology",
"Cognitive neuroscience",
"Systems neuroscience",
"Human-robot interaction",
"Evolutionary psychology",
"Social network analysis"
]
```

```
def square_print_topic_indices(r_value, g_value, b_value, file):
    index1 = r_value % len(index_1)
    index2 = g_value % len(index_1)
    index3 = b_value % len(index_1)
    ""
    file.write(f"\t\t\t{topics[index1]}\n")
    file.write(f"\t\t\t{topics[index2]}\n")
    file.write(f"\t\t\t{topics[index3]}\n")
    ""
    file.write(f"\t\t\t{index_1[index1]}\n")
    file.write(f"\t\t\t{index_1[index2]}\n")
    file.write(f"\t\t\t{index_1[index3]}\n")

squareRatio = str(index1) + ":" + str(index2) + ":" + str(index3)
return squareRatio
```

```

def font_print_topic_indices(r_value, g_value, b_value, file):
    index1_ = r_value % len(index_2)
    index2_ = g_value % len(index_2)
    index3_ = b_value % len(index_2)
    file.write(f"\t\t{index1_}\n")
    file.write(f"\t\t{index2_}\n")
    file.write(f"\t\t{index3_}\n")

fontRatio = str(index1_) + ":" + str(index2_) + ":" + str(index3_)
return fontRatio


def get_index(i, j, grid_size):
    return i * grid_size + j

def extract_rgb_values(color):
    if len(color) != 7 or color[0] != '#':
        raise ValueError('Input should be a hex color code in the format "#RRGGBB"')
    try:
        red = int(color[1:3], 16)
        green = int(color[3:5], 16)
        blue = int(color[5:7], 16)
        return red, green, blue
    except ValueError:
        raise ValueError('Invalid color code. RGB values should be hex digits (0-9, A-F)')

def GO():
    log_file = "color_log.txt"
    ##      topic_file = "index_1.txt"
    ##      topic2_file = "index_2.txt" # Corrected this line to read from a different file
    gridWidth = int(input("Enter gridWidth: "))
    # Read topic files to get the topics and topic2s
    ##      with open(topic_file, "r", encoding='utf8') as file:
    ##          topics = [line.strip() for line in file]
    ##
    ##      with open(topic2_file, "r", encoding='utf8') as file:
    ##          topic2s = [line.strip() for line in file]

    numTopics = len(index_1)
    numTopic2s = len(index_2)

    # Read color log file
    with open(log_file, "r") as file:
        lines = file.readlines()

    lineNumber = 0
    # Save output to a file
    output_file = "output.txt"

```

```

with open(output_file, "w") as file:
    print()
for line in lines:
    parts = line.strip().split(",")
    if len(parts) >= 5:
        xcoor = int(parts[1])
        ycoor = int(parts[0])
        index_ = (gridWidth * ycoor) + xcoor
        character = parts[2]
        squareColor = parts[3]
        fontColor = parts[4]
        notes = parts[5]

        # Extract RGB values from squareColor
        red, green, blue = extract_rgb_values(squareColor)

        # Extract RGB values from fontColor
        fontRed, fontGreen, fontBlue = extract_rgb_values(fontColor)

        # Calculate the grid size
        grid_size = int(parts[0])

        # Select topics based on RGB values from squareColor
        topicIndices = [(red + get_index(i, j, grid_size)) % numTopics for i in range(grid_size) for j in
range(grid_size)]

    lineNumber += 1

    with open(output_file, "a") as file:
        file.write(f"\nCompiling Line/s: {lineNumber}\n\n")
        file.write(f"\nLine({lineNumber})\n\n")
        file.write(f"\t0\tLineNumber: {lineNumber}\tx-coordinate: {xcoor}\ty-coordinate: {ycoor}\t"
Character: {character}\tRGB: {squareColor}\tRGB: {fontColor}\t Note: {notes}\t GridDimensions: "
{gridWidth}x{gridWidth}\n\n")
        #print(topicIndices)
        file.write(f"\nLine/s({lineNumber}) Output:\n\n")
        file.write(f"\n#define Line({lineNumber}){\n\n")
        # Inside the main function
        file.write(f"\t1_0(squareColor){\n\n")
        file.write(f"\t\t{character}\n")
        file.write(f"\t\t(R: {red}, G: {green}, B: {blue})\n")
        s_ratio = square_print_topic_indices(red, green, blue, file)
        #square_print_topic_indices(red, green, blue, topics, file) # Modify this line
        file.write("\t)\n")
        file.write("\t;\n\n")
        file.write(f"\t1_1(fontColor){\n\n")
        file.write(f"\t\t{character}\n")
        file.write(f"\t\t(R: {fontRed}, G: {fontGreen}, B: {fontBlue})\n")
        f_ratio = font_print_topic_indices(fontRed, fontGreen, fontBlue, file)

```

```

#font_print_topic_indices(fontRed, fontGreen, fontBlue, topic2s, file) # And this line
file.write("\t\t)\n")
file.write(f"\treturn squareRatio({s_ratio}), fontRatio({f_ratio}), sqaureIndex({index_})\n")
#file.write(f"\treturn fontRatio({f_ratio})\n")
file.write(")\n\n"

```

GO()

```

def create_menu():
    global control_frame
    def home():
        canvas.pack_forget()
        menubar.destroy()
        control_frame.pack()
        char_entry.pack_forget()
        char_label.pack_forget()
        #grid_size_label.pack_forget()
        #grid_size_entry.pack_forget()
        #font_size_label.pack_forget()
        #font_size_entry.pack_forget()
        #adjust_button.pack_forget()
        char_note.pack_forget()
        char_note_label.pack_forget()

        root.destroy()
        switch = 1
        setup_mode(switch)

    # Create a menubar
    menubar = tk.Menu(root)

    # Create an options menu and add it to the menubar
    options_menu = tk.Menu(menubar, tearoff=0)
    #menubar.add_cascade(label="Options", menu=options_menu)

    # Create a dropdown menu and add it to the menubar
    dropdown = tk.Menu(menubar, tearoff=0)
    menubar.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"

    # Add commands to dropdown menu

    # Add checkbutton to options menu
    dropdown.add_command(label="Toggle Manual/Random Colour", command=manual_colour)

    dropdown.add_command(label="Refresh", command=refresh_canvas)
    dropdown.add_command(label="Undo", command=undo_last)
    dropdown.add_command(label="Save Session", command=save_session)
    dropdown.add_command(label="Load Session", command=load_session)

```

```
dropdown.add_command(label="Save as Image", command=save_canvas)
dropdown.add_command(label="Compile Canvas", command=compileGOSmX)
# Add the new command for resizing images
dropdown.add_command(label="Select and Resize Image", command=select_and_resize_image)
# Add the command for selecting images
dropdown.add_command(label="Select Images", command=select_and_add_images)
# Add the toggle mode command
dropdown.add_command(label="Toggle Draw Chars/Images", command=toggle_mode)
#dropdown.add_command(label="Exit", command=exit_app)
#dropdown.add_command(label="Toggle Type/Click Mode", command=toggle_typing_mode)
dropdown.add_command(label="About Program", command=show_about)
dropdown.add_command(label="Go Home", command=home)
dropdown.add_command(label="Edit Grid Dimensions", command=adjust_grid_and_font)
dropdown.add_command(label="Typing Mode", command=show_typing_mode_menu)
```

```
# Set the menubar
root.config(menu=menubar)
```

```
def create_menu_1():
    global control_frame
    def home1():
        canvas.pack_forget()
        menubar1.destroy()
        root.destroy()
        switch = 1
        setup_mode(switch)
```

```
# Create a menubar
menubar1 = tk.Menu(root)
```

```
# Create an options menu and add it to the menubar
options_menu1 = tk.Menu(menubar1, tearoff=0)
#menubar.add_cascade(label="Options", menu=options_menu)
```

```
# Create a dropdown menu and add it to the menubar
dropdown = tk.Menu(menubar1, tearoff=0)
menubar1.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"
```

```
# Add commands to dropdown menu
```

```
# Add checkbutton to options menu
dropdown.add_command(label="About Program", command=show_about)
dropdown.add_command(label="Change Mode", command=home1)
```

```
# Set the menubar
root.config(menu=menubar1)
```

```
def manual_colour():
    # Toggle the value of color_mode_var
    current_value = color_mode_var.get()
    color_mode_var.set(not current_value)

    ...
# Create a tkinter window
root = tk.Tk()
root.title("General Operating System")
root.iconbitmap("logo.ico")
"""

def show_typing_mode_menu():
    global color_
    while True:
        print("1. Use default UTF-8 mapping")
        print("4. Exit")
        choice = input("Choose an option: ")

        if choice == "1":
            # Use default UTF-8 mapping
            print("Using default UTF-8 mapping.")
            color_ = input("Use Default Colour Scheme (1): ")
            if(color_ == '1'):
                # Initialize bg_color and font_color with default values at the global scope
                bg_color = "#FFFFFF" # Default white background color
                font_color = "#000000" # Default black font color
                break
            else:
                print("Invalid choice. Please try again.")

    canvas.bind("<Key>", type_character)
    canvas.focus_set()

def set_background_color():
    global color_
    #bg_color = tk.colorchooser.askcolor()[1]
    color = input("Enter background color (#FFFFFF for example):")

def set_font_color():
    global font_color, font_size
    #font_color = tk.colorchooser.askcolor()[1]
    font_color = input("Enter background color (#000000 for example):")
```

```
font_size = input("Enter font_size: ")

def type_character(event):
    global current_row, current_col, col, font_size, font_color, color, canvas_width, square_size
    color = "#FFFFFF"
    font_color = "#000000"
    char = event.char
    if char: # Ignore non-character events
        try:
            draw_char1(canvas, char, current_row, current_col, font_size, color)
            current_col += 1
            if (current_col >= canvas_width/square_size):
                current_col = 0
                current_row += 1
            if(current_row >= canvas_width/square_size):
                current_col = 0
                current_row = 0
        except Exception as e:
            print("Something Went Wrong...")
            print(e)
```

```
def setup_mode(sw):

    global mode_entry, mode_label, square_size, grid_size, char_count, last_drawn
    global root, color_mode_var, logging, paused, canvas_width, canvas_height, canvas
    global grid_size_label, grid_size_entry, font_size_label, font_size_entry, adjust_button
    global save_Sbutton, load_button, char_label, char_entry, char_note_label, char_note, char_button
    global refresh_button, undo_button, control_frame, submit_button, col, color_
    global image_selection_frame # Use the global variable

    current_row = 0
    current_col = 0
    color_ = 1

    # Define the size of the squares and the grid
    #square_size = 70
    square_size = int(input("Enter sub-square size: "))
    #grid_size = 10
    grid_size = int(input("Enter grid size by the number of sub-squares: "))
    char_count = 0
    last_drawn = []
```

```
window_height = square_size * grid_size + (square_size * 2)
window_width = square_size * grid_size

if(sw == 1):
    # Create a tkinter window
    root = tk.Tk()
    # Define the frame for displaying the image selection
    #image_selection_frame = tk.Frame(root)
    #image_selection_frame.pack(side='top') # Adjust the side as per your layout
    root.title("C")
    #root.iconbitmap("logo.ico")
    sw = 0
    switch = 0

###

# Get screen width and height
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

# Calculate position of the window
position_top = int(screen_height / 2 - window_height / 2)
position_right = int(screen_width / 2 - window_width / 2)

# Set the dimensions of the window and where it is placed
root.geometry(f'{window_width}x{window_height}+{position_right}+{position_top}')

###

# Create a BooleanVar for the color mode and set it to False initially
color_mode_var = BooleanVar(value=False)

# Setup logging
logging.basicConfig(filename='logging.txt', level=logging.INFO)

paused = tk.BooleanVar()
paused.set(False)

canvas_width = grid_size * square_size
canvas_height = grid_size * square_size
canvas = tk.Canvas(root, width=canvas_width, height=canvas_height)
col = canvas_width
canvas.pack()
canvas.bind("<Button-1>", on_canvas_click)

# In the function where you create the canvas...
canvas.bind("<Key>", on_key_press)
canvas.focus_set()
```

```

# Create input fields and buttons for grid and font size input
grid_size_label = tk.Label(root, text="Grid size:")
grid_size_entry = tk.Entry(root, width=5)
font_size_label = tk.Label(root, text="Font size:")
font_size_entry = tk.Entry(root, width=5)
adjust_button = tk.Button(root, text='Adjust Grid & Font', command=adjust_grid_and_font)

# Create the save and load buttons
save_Sbutton = tk.Button(root, text='Save Session', command=save_session, bg="white", padx=5,
pady=0)
load_button = tk.Button(root, text='Load Session', command=load_session, bg="white", padx=5,
pady=0)

# Create input fields and buttons for mode and character input
mode_label = tk.Label(root, text="Enter ... 'IDE' for an\nInteractive Development Environment\nor
Learning anything!")
mode_entry = tk.Entry(root)
submit_button = tk.Button(root, text='LEARN', command=submit_mode)

"""

filename = simpledialog.askstring("Save Canvas", "Enter filename:")
if filename:
"""

char_label = tk.Label(root, text="Enter symbol/s to draw: ")
char_entry = tk.Entry(root, width=12)
char_note_label = tk.Label(root, text="Enter Note: ")
char_note = tk.Entry(root, width=12)
#char_button = tk.Button(root, text='Draw Character', command=draw_char)

refresh_button = tk.Button(root, text='Refresh', command=refresh_canvas, bg="white", padx=5,
pady=0)
undo_button = tk.Button(root, text='Undo', command=undo_last, bg="white", padx=5, pady=0)

control_frame = tk.Frame(root)

mode_label.pack()
mode_entry.pack()
submit_button.pack()
# create_menu(root)

# Run the tkinter main loop
root.mainloop()

global switch
switch = 1

```

```

setup_mode(switch)

if(entrance == 4):
    #language()
def get_subranges_from_user():
    subranges = []
    number_of_subranges = int(input("Enter the number of subranges: "))

    for i in range(number_of_subranges):
        lower_bound = int(input("Enter the lower bound of subrange {i + 1}: "))
        upper_bound = int(input("Enter the upper bound of subrange {i + 1}: "))
        subranges.append((lower_bound, upper_bound))

    return subranges

def main():
    print("(pt) == Universal Set = Computer")

    # Read the file
    gettext = input("Enter filename of the .txt file containing single characters per line (Unicode
    Characters): ")
    #with open("alpha.txt", "r", encoding="utf-8") as alpha_file:
    with open(gettext, "r", encoding="utf-8") as alpha_file:
        file_characters = [line.strip() for line in alpha_file if line.strip()]

    subranges = get_subranges_from_user()
    a = []

    for subrange in subranges:
        lower_bound = max(0, subrange[0]- 1)
        upper_bound = min(len(file_characters)- 1, subrange[1]- 1)
        a.extend(file_characters[lower_bound:upper_bound + 1])

    print("Characters have been added to the array from the specified subranges.")

choice = input("Enter a to use your subrange of characters. Else, Enter e to Exit: ")
if choice.lower() == 'a':
    z = int(input("Enter the size of your array: "))
    k = z - 1

    noc = int(input("\nEnter n: "))
    print(f"\nNumber Of FILE Cells = {noc}")
    n = noc

    id = 0

    with open("LSC-RENAME.txt", "w", encoding="utf-8") as p:
        ai = int(input("\nEnter nth File System (Number of Cells Per File of the nth File System). Let the
        value be equal to n for a progressionless session: "))

```

```

for n in range(n, ai + 1):
    nbr_comb = int(math.pow(k + 1, n))

    for row in range(nbr_comb):
        p.write(f"\n\n{n}CF{id}\n\n")
        id += 1
        for col in range(n - 1, -1, -1):
            rdiv = int(math.pow(k + 1, col))
            cell = (row // rdiv) % (k + 1)

            if cell < len(a):
                if col == 0:
                    p.write(f"{a[cell]}")
                else:
                    p.write(f"{a[cell]} ")

if __name__ == "__main__":
    main()

if(entrance == 5):
    text_editor()
if(entrance == 6):
    generate_cmp()
if(entrance == 7):
    break
if(entrance == 8):
    class Point:
        def __init__(self, material, x, y, z, x_i, y_i, z_i, id_number):
            self.material = material
            self.x = x # x-coordinate relative to the origin
            self.y = y # y-coordinate relative to the origin
            self.z = z # z-coordinate relative to the origin
            self.x_i = x_i # x-coordinate of the origin
            self.y_i = y_i # y-coordinate of the origin
            self.z_i = z_i # z-coordinate of the origin
            self.id_number = id_number

        def __str__(self):
            return f"Point(ID: {self.id_number}, Material: {self.material}, Relative: ({self.x}, {self.y}, {self.z}), Origin: ({self.x_i}, {self.y_i}, {self.z_i}))"

        def log_entry(self):
            return f"{self.material} {self.x} {self.y} {self.z} {self.x_i} {self.y_i} {self.z_i} {self.id_number}\n"

    class Structure:
        def __init__(self):
            self.points = []
            self.next_id = 1
            self.load_points_from_log()

```

```

def load_points_from_log(self):
    try:
        with open("logs.txt", "r") as log_file:
            for line in log_file:
                parts = line.split()
                if len(parts) == 8:
                    material, x, y, z, x_i, y_i, z_i, id_number = parts
                    # Convert string to appropriate types
                    point = Point(
                        material,
                        #int(x), int(y), int(z),
                        #int(x_i), int(y_i), int(z_i),
                        float(x), float(y), float(z),
                        float(x_i), float(y_i), float(z_i),
                        int(id_number)
                    )
                    self.points.append(point)
                    # Update the next ID to be larger than any ID already used
                    self.next_id = max(self.next_id, point.id_number + 1)
    except FileNotFoundError:
        print("logs.txt file not found. Starting with an empty structure.")

def add_point(self, material, x, y, z, x_i, y_i, z_i):
    point = Point(material, x, y, z, x_i, y_i, z_i, self.next_id)
    self.points.append(point)
    self.next_id += 1
    # Log the new point
    with open("logs.txt", "a") as log_file:
        log_file.write(point.log_entry())
    """

def add_point(self, material, x, y, z, x_i, y_i, z_i):
    point = Point(material, x, y, z, x_i, y_i, z_i, self.next_id)
    self.points.append(point)
    with open("logs.txt", "a") as log_file:
        log_file.write(point.log_entry())
    self.next_id += 1 # Prepare ID for the next point
"""

def __str__(self):
    return "\n".join(str(point) for point in self.points)

def plot_structure(self, point_id=None):
    x_coords = [point.x + point.x_i for point in self.points]
    y_coords = [point.y + point.y_i for point in self.points]
    z_coords = [point.z + point.z_i for point in self.points]

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

```

```

scatter = ax.scatter(x_coords, y_coords, z_coords, color='red')

if point_id is not None:
    # If a point ID is specified, zoom in on that point
    point = next((p for p in self.points if p.id_number == point_id), None)
    if point:
        ax.set_xlim(point.x + point.x_i - 5, point.x + point.x_i + 5)
        ax.set_ylim(point.y + point.y_i - 5, point.y + point.y_i + 5)
        ax.set_zlim(point.z + point.z_i - 5, point.z + point.z_i + 5)
        # Highlight the selected point
        ax.scatter([point.x + point.x_i], [point.y + point.y_i], [point.z + point.z_i], color='blue', s=100)
    else:
        print(f"No point found with ID {point_id}")

# Set the axes labels
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
plt.title('3D Plot of Points in Structure')

# Show the plot
plt.show()

class Interface:
    def __init__(self):
        self.structure = Structure()

    def process_command(self, command):
        parts = command.split()
        if parts[0] == "add_point":
            # Expecting command format: add_point material x y z x_i y_i z_i
            if len(parts) != 8:
                print("Invalid number of arguments for add_point. Expected format: add_point material x y z x_i y_i z_i")
            return
        try:
            #self.structure.add_point(parts[1], int(parts[2]), int(parts[3]), int(parts[4]), int(parts[5]),
            int(parts[6]), int(parts[7]))
            self.structure.add_point(parts[1], float(parts[2]), float(parts[3]), float(parts[4]), float(parts[5]),
            float(parts[6]), float(parts[7]))
        except ValueError:
            print("Invalid input ...")

        elif parts[0] == "plot":
            #self.load_point_from_log()
            self.structure.plot_structure()

        elif parts[0] == "zoom_on_point":
            #self.load_point_from_log()
            if len(parts) < 2 or not parts[1].isdigit():

```

```

        print("Invalid command. Usage: zoom_on_point <point_id>")
        return
    point_id = int(parts[1])
    self.structure.plot_structure(point_id=point_id)

else:
    print("Unknown command")

def run(self):
    print("Available Commands:")
    print("\tadd_point material x y z x_i y_i z_i")
    print("\tplot")
    print("\tzoom_on_point <point_id>")
    print("\tprint_structure")
    print("\texit")
    while True:
        command = input("Enter command: ")
        if command.lower() == "exit":
            break
        if command.lower() == "print_structure":
            print(self.structure)
        else:
            self.process_command(command)

# Main loop
interface = Interface()
interface.run()

if(entrance == 9):
    while(5 > 4):
        mode = get_user_input("Enter 1 to generate new circuits, 2 to generate .png from existing file, 3 to
exit: ", int)

        if mode == 1:
            # Generate new circuits and save them to a file
            qubits = int(input("Enter number of qubits per circuit configuration (Positive Integer): "))
            gates = ['x', 'y', 'z', 'h', 's', 'sdg', 't', 'tdg', 'rx', 'ry', 'rz', 'cx']
            num_qubits = qubits # Number of qubits in the circuit
            num_cells = get_user_input("Enter the number of cells (gate layers) per circuit: ")

            with open('source.txt', 'w') as file:
                gate_combinations = product(gates, repeat=num_cells * num_qubits)

                for idx, combo in enumerate(gate_combinations):
                    qc = QuantumCircuit(num_qubits)
                    gate_sequence = ""
                    for i, gate in enumerate(combo):
                        qubit = i % num_qubits
                        if gate in ['x', 'y', 'z', 'h', 's', 'sdg', 't', 'tdg']:

```

```

        getattr(qc, gate)(qubit)
        gate_sequence += f"{gate}({qubit}) "
    elif gate == 'cx':
        getattr(qc, gate)(qubit, (qubit + 1) % num_qubits)
        gate_sequence += f"{gate}({qubit}, {(qubit + 1) % num_qubits}) "
    elif gate in ['rx', 'ry', 'rz']:
        getattr(qc, gate)(np.pi/2, qubit)
        gate_sequence += f"{gate}({pi/2},{qubit}) "

    file.write(f"Circuit {idx}: {gate_sequence.strip()}\n")
    print(f"Saved configuration for Circuit {idx}")

elif mode == 2:
    # Generate .png files from an existing .txt file
    filename = get_user_input("Enter the name of the source file (including .txt extension): ", str)
    qbits = get_user_input("Enter number of qubits per circuit configuration (Positive Integer): ")
    num_qubits = qbits # Use this for constructing QuantumCircuit instances

    print("Enter the line numbers or ranges (e.g., 1-3, 5, 7-10) to generate .png files:")
    line_numbers = input("Line numbers/ranges: ").split(',') # Direct input call for complex string input

    with open(filename, 'r') as file:
        lines = file.readlines()
        for number in line_numbers:
            if '-' in number:
                start, end = map(int, number.split('-'))
                for i in range(start, end + 1):
                    qc = generate_circuit_from_line(lines[i - 1], num_qubits)
                    circuit_drawer(qc, output='mpl', filename=f"circuit_{i - 1}.png")
                    plt.close() # Close the figure after saving
                    print(f"Saved circuit_{i - 1}.png")
            else:
                i = int(number)
                qc = generate_circuit_from_line(lines[i - 1], num_qubits)
                circuit_drawer(qc, output='mpl', filename=f"circuit_{i - 1}.png")
                plt.close() # Close the figure after saving
                print(f"Saved circuit_{i - 1}.png")

    elif mode == 3:
        break

if(entrance == 10):

    while True:
        mode = request_user_input("Enter 1 to generate new circuits, 2 to generate .png from existing file, 3 to exit: ", int)

        if mode == 1:
            gates = ['AND', 'OR', 'NOT', 'NAND', 'NOR', 'XOR', 'XNOR']
            num_elements = request_user_input("Enter the number of elements (gates) per circuit: ")

```

```

with open('source_classical.txt', 'w') as file:
    gate_combinations = product(gates, repeat=num_elements)

    for idx, combo in enumerate(gate_combinations):
        circuit_line = ', '.join(combo)
        file.write(f"Circuit {idx}: {circuit_line}\n")
        print(f"Saved configuration for Circuit {idx}")

elif mode == 2:
    filename = request_user_input("Enter the name of the source file (including .txt extension): ", str)

    print("Enter the line numbers or ranges (e.g., 1-3, 5, 7-10) to generate .png files:")
    line_numbers = input("Line numbers/ranges: ").split(',')

    with open(filename, 'r') as file:
        lines = file.readlines()
        for number in line_numbers:
            if '-' in number:
                start, end = map(int, number.split('-'))
                for i in range(start, end + 1):
                    try:
                        line_to_process = lines[i - 1].split(':')[1].strip()
                        circuit_diagram = generate_classical_circuit_from_line(line_to_process, f"circuit_{i-1}.png")
                        print(f"Saved circuit_{i-1}.png")
                    except Exception as e:
                        print(f"An error occurred while processing line {i}: {e}")
            else:
                i = int(number)
                try:
                    line_to_process = lines[i - 1].split(':')[1].strip()
                    circuit_diagram = generate_classical_circuit_from_line(line_to_process, f"circuit_{i-1}.png")
                    print(f"Saved circuit_{i-1}.png")
                except Exception as e:
                    print(f"An error occurred while processing line {i}: {e}")

elif mode == 3:
    break

if(entrance == 11):

    # Define the path for the CSV file
    file_path = 'cell_data.csv'

    # Define the headers for the CSV file
    headers = ['x', 'y', 'z', 'function_id', 'function_word', 'function_definition']

```

```

# Check if the file exists; if not, create it and write the headers
if not os.path.exists(file_path):
    with open(file_path, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(headers)

# Start the program loop
while True:
    # Prompt the user to enter the cell data or type 'end_program' to exit
    print("Enter the cell's data (or type 'end_program' to exit):")
    user_input = input("Format: x,y,z,function_id,function_word,function_definition\\n")

    # Check if the user wants to end the program
    if user_input.lower() == 'end_program':
        break

    # Split the input into components
    data = user_input.split(',')

    # Check if the input is valid
    if len(data) != 6:
        print("Invalid input format. Please try again.")
        continue

    # Append the data to the CSV file
    with open(file_path, mode='a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(data)

        print("Data added to the file.")

    if(entrance == 12):
        main0()
    if(entrance == 13):
        countables()
    if(entrance == 14):
        # Specify the file path for the 26-bit combinations
        name_ext = input("Enter filename with .txt extension: ")
        file_path_26_bit = "llanguageMod/binary-configs/" + name_ext
        n_ = int(input("Enter bit length per binary string: "))

        # Generate and save the 26-bit combinations
        generate_and_save_n_bit_combinations(file_path_26_bit, n_)

    if(entrance == 15):
        help_print()
    if(entrance == 16):

        base = int(input("Enter the integer base of origin file (between 2 and 16):"))

        input_file = input("Enter origin file path: ")
        output_file = input("Enter output file path: ")

```

```
try:
    convert_file(input_file, output_file, base)
    print(f"Successfully converted {input_file} to {output_file}")
except FileNotFoundError:
    print(f"Error: File not found. Please check the paths for {input_file} and {output_file}")

if(entrance == 17):
    while(5 > 2):
        inty = int(input("0 for main_menu, 1 to continue: "))
        if(inty == 0):
            break
        if(inty == 1):
            main2()
if(entrance == 18):
    while(1 < 2):
        intt = int(input("0 for main_menu, 1 to continue: "))
        if(intt == 0):
            break
        if(intt == 1):
            main18()
if(entrance == 19):
    print("""\n\nThe following is an example mapping for the font index:\n
0 a
1 b
2 c
3 d
4 e
5 f
6 g
7 h
8 i
9 j
10 k
11 l
12 m
13 n
14 o
15 p
16 q
17 r
18 s
19 t
20 u
21 v
22 w
23 x
24 y
25 z
26 '
```

27,
28/
29<
30>
31?
32;
33:
34@
35#
36~
37]
38[
39{
40}
41'
42~
43|
44|
45!
46"
47€
48\$
49%
50^
51&
52*
53(
54)
55-
56_
57+
58=
59.
60A
61B
62C
63D
64E
65F
66G
67H
68I
69J
70K
71L
72M
73N
74O
75P
76Q

```
77 R
78 S
79 T
80 U
81 V
82 W
83 X
84 Y
85 Z
86 0
87 1
88 2
89 3
90 4
91 5
92 6
93 7
94 8
95 9
96 \\
""")  
grid_size = int(input("Enter the grid size (number of squares along one side): "))  
square_size = int(input("Enter the pixel length of each sub-square: "))  
font_size = int(input("Enter the font size for the symbols: "))  
file_name = input("Enter the name of the configuration file to save (.txt): ")  
  
configurations = []  
for i in range(grid_size * grid_size):  
    print(f"Configuration for square {i+1}")  
    symbol_index = int(input("Enter symbol index (0 to 96): "))  
    square_color_index = 0 + int(16777216 / (symbol_index + 1))  
    font_color_index = 16777216 - int(16777216 / (symbol_index + 1))  
    configurations.append((square_color_index, symbol_index, font_color_index))  
  
save_configuration_to_file19(file_name, grid_size, square_size, font_size, configurations)  
print(f"Configuration saved to {file_name}")  
if(entrance == 21):  
    while(5 > 2):  
        intys = int(input("0 for main_menu, 1 to continue: "))  
        if(intys == 0):  
            break  
        if(intys == 1):  
            # Main  
            initial_a = int(input("Enter initial a (a > b): "))  
            initial_b = int(input("Enter initial b (b < a): "))  
            user_iterations = int(input("Enter number of additional user iterations: "))  
  
            fsm = ProximalPrimeFSM(initial_a, initial_b, user_iterations)  
            fsm.run()  
            coords = prepare_coordinates21(fsm.primes)
```

```

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ani = FuncAnimation(fig, animate21, frames=len(coords), repeat=False)
plt.show()

if(entrance == 22):
    while(5 > 2):
        intys22 = int(input("0 for main_menu, 1 to continue: "))
        if(intys22 == 0):
            break
        if(intys22 == 1):
            main22()
if(entrance == 23):
    while(5 > 2):
        intys23 = int(input("0 for main_menu, 1 to continue: "))
        if(intys23 == 0):
            break
        if(intys23 == 1):
            main23()
if(entrance == 24):
    while(5 > 2):
        intys23 = int(input("0 for main_menu, 1 to continue: "))
        if(intys23 == 0):
            break
        if(intys23 == 1):
            interactive_cli()
if(entrance == 25):
    while(5 > 2):
        intys25 = int(input("0 for main menu, 1 to continue: "))
        if(intys25 == 0):
            break
        if(intys25 == 1):
            #import tkinter as tk
            #from tkinter import simpledialog, messagebox

class BlockDiagramApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Block Diagram Creator")
        self.canvas = tk.Canvas(self.root, bg="white")
        self.canvas.pack(fill=tk.BOTH, expand=True)

        self.grid_size = self.get_grid_size()
        self.draw_grid()

        self.canvas.bind("<Button-1>", self.add_circle)
        self.canvas.bind("<Button-3>", self.add_rectangle)
        self.canvas.bind("<Button-2>", self.start_line)
        self.root.bind("<Shift_L>", self.toggle_turn)

```

```

self.elements = []
self.line_start = None
self.current_line = None
self.turn_points = []
self.label_for_line = None
self.placing_line_label = False

def get_grid_size(self):
    size = simpledialog.askinteger("Grid Size", "Enter the grid size (pixels):", minvalue=10,
maxvalue=100)
    if size is None:
        size = 20 # Default grid size
    return size

def snap_to_grid(self, x, y):
    grid_size = self.grid_size
    snapped_x = round(x / grid_size) * grid_size
    snapped_y = round(y / grid_size) * grid_size
    return snapped_x, snapped_y

def draw_grid(self):
    self.canvas.delete("grid_line") # Clear existing grid lines if any
    width = self.canvas.winfo_width()
    height = self.canvas.winfo_height()
    grid_size = self.grid_size

    for i in range(0, width, grid_size):
        self.canvas.create_line([(i, 0), (i, height)], tag='grid_line', fill='lightgray')

    for i in range(0, height, grid_size):
        self.canvas.create_line([(0, i), (width, i)], tag='grid_line', fill='lightgray')

    self.canvas.tag_lower("grid_line") # Move grid lines to the background

def add_circle(self, event):
    if self.placing_line_label:
        self.place_label(event)
        return

    x, y = self.snap_to_grid(event.x, event.y)
    label = self.get_label()
    if label is not None:
        circle = self.canvas.create_oval(x-20, y-20, x+20, y+20, fill="lightblue")
        text = self.canvas.create_text(x, y, text=label)
        self.elements.append(("circle", x, y, label, circle, text))

def add_rectangle(self, event):
    if self.placing_line_label:
        self.place_label(event)
        return

```

```

x, y = self.snap_to_grid(event.x, event.y)
label = self.get_label()
if label is not None:
    rect = self.canvas.create_rectangle(x-30, y-20, x+30, y+20, fill="lightgreen")
    text = self.canvas.create_text(x, y, text=label)
    self.elements.append(("rectangle", x, y, label, rect, text))

def start_line(self, event):
    if self.placing_line_label:
        self.place_label(event)
    return

x, y = self.snap_to_grid(event.x, event.y)
if self.line_start is None:
    self.line_start = (x, y)
    self.current_line = []
    self.turn_points = [(x, y)]
else:
    self.add_line_segment(x, y)
    self.turn_points.append((x, y))

def add_line_segment(self, x, y):
    last_x, last_y = self.turn_points[-1]
    if self.current_line is not None:
        line = self.canvas.create_line(last_x, last_y, x, y, fill="black", width=2)
        label = self.get_label()
        if label is not None:
            self.label_for_line = label
            self.placing_line_label = True
            self.canvas.bind("<Button-1>", self.place_label)
            self.elements.append(("line", last_x, last_y, x, y, line))

    #if len(self.turn_points) >= 2:
    self.line_start = None
    self.current_line = None

def place_label(self, event):
    if self.label_for_line is not None:
        x, y = event.x, event.y
        text = self.canvas.create_text(x, y, text=self.label_for_line, fill="red")
        self.elements.append(("line_label", x, y, self.label_for_line, text))
        self.label_for_line = None
        self.placing_line_label = False
        self.canvas.unbind("<Button-1>")
        self.canvas.bind("<Button-1>", self.add_circle)
        self.canvas.bind("<Button-3>", self.add_rectangle)

def toggle_turn(self, event):
    if self.line_start is not None and self.turn_points:

```

```

        self.turn_points.append(self.turn_points[-1])

    def get_label(self):
        return simpledialog.askstring("Input", "Enter the label for the component or line:")

    def resize_canvas(self, event):
        self.draw_grid()

    #if __name__ == "__main__":
    root = tk.Tk()
    app = BlockDiagramApp(root)
    root.geometry("800x600")
    root.bind("<Configure>", app.resize_canvas)
    root.mainloop()

if(entrance == 26):
    while(5 > 2):
        intys26 = int(input("0 for main menu, 1 to continue: "))
        if(intys26 == 0):
            break
        if(intys26 == 1):
            class Point:
                def __init__(self, x, y, z):
                    self.x = x
                    self.y = y
                    self.z = z

                def __repr__(self):
                    return f"Point(x={self.x}, y={self.y}, z={self.z})"

            class PointStructure:
                def __init__(self, index, origin):
                    self.index = index
                    self.origin = origin
                    self.points = {}

                def add_point(self, point_index, x, y, z):
                    self.points[point_index] = Point(x, y, z)

                def apply_transformation(self, transformation):
                    for point in self.points.values():
                        transformation(point)

                def __repr__(self):
                    return f"PointStructure(index={self.index}, origin={self.origin}, points={self.points})"

            class Transformation:
                @staticmethod
                def translate(dx, dy, dz):
                    def transformation(point):
                        point.x += dx

```

```

        point.y += dy
        point.z += dz
    return transformation

    @staticmethod
    def scale(sx, sy, sz):
        def transformation(point):
            point.x *= sx
            point.y *= sy
            point.z *= sz
        return transformation

    @staticmethod
    def rotate_x(angle):
        def transformation(point):
            y = point.y * math.cos(angle) - point.z * math.sin(angle)
            z = point.y * math.sin(angle) + point.z * math.cos(angle)
            point.y = y
            point.z = z
        return transformation

    @staticmethod
    def rotate_y(angle):
        def transformation(point):
            x = point.x * math.cos(angle) + point.z * math.sin(angle)
            z = -point.x * math.sin(angle) + point.z * math.cos(angle)
            point.x = x
            point.z = z
        return transformation

    @staticmethod
    def rotate_z(angle):
        def transformation(point):
            x = point.x * math.cos(angle) - point.y * math.sin(angle)
            y = point.x * math.sin(angle) + point.y * math.cos(angle)
            point.x = x
            point.y = y
        return transformation

def get_user_input():
    structures = []

    while True:
        print("\nCreate a new point structure")
        index = int(input("Enter structure index: "))
        x0 = float(input("Enter x-coordinate of the origin: "))
        y0 = float(input("Enter y-coordinate of the origin: "))
        z0 = float(input("Enter z-coordinate of the origin: "))
        origin = Point(x0, y0, z0)
        structure = PointStructure(index, origin)

```

```

while True:
    point_index = int(input("Enter point index (or-1 to stop adding points): "))
    if point_index ==-1:
        break
    x = float(input("Enter x-coordinate of the point: "))
    y = float(input("Enter y-coordinate of the point: "))
    z = float(input("Enter z-coordinate of the point: "))
    structure.add_point(point_index, x, y, z)

structures.append(structure)

another = input("Would you like to add another point structure? (yes/no): ").strip().lower()
if another != 'yes':
    break

return structures

def apply_user_transformations(structures):
    while True:
        structure_index = int(input("Enter the index of the point structure to transform (or-1 to stop):"))
        if structure_index ==-1:
            break
        structure = next((s for s in structures if s.index == structure_index), None)
        if not structure:
            print(f"No point structure found with index {structure_index}")
            continue

        print("Choose a transformation: translate, scale, rotate_x, rotate_y, rotate_z")
        transformation_type = input("Enter the transformation type: ").strip().lower()

        if transformation_type == 'translate':
            dx = float(input("Enter translation along x: "))
            dy = float(input("Enter translation along y: "))
            dz = float(input("Enter translation along z: "))
            transformation = Transformation.translate(dx, dy, dz)

        elif transformation_type == 'scale':
            sx = float(input("Enter scaling factor along x: "))
            sy = float(input("Enter scaling factor along y: "))
            sz = float(input("Enter scaling factor along z: "))
            transformation = Transformation.scale(sx, sy, sz)

        elif transformation_type == 'rotate_x':
            angle = float(input("Enter rotation angle around x-axis (in radians): "))
            transformation = Transformation.rotate_x(angle)

        elif transformation_type == 'rotate_y':
            angle = float(input("Enter rotation angle around y-axis (in radians): "))

```

```
transformation = Transformation.rotate_y(angle)

elif transformation_type == 'rotate_z':
    angle = float(input("Enter rotation angle around z-axis (in radians): "))
    transformation = Transformation.rotate_z(angle)

else:
    print("Invalid transformation type")
    continue

structure.apply_transformation(transformation)
print("\nUpdated Point Structure:")
print(structure)

def main():
    print("Welcome to the Point Structure Transformation Program")

    structures = get_user_input()

    print("\nInitial Point Structures:")
    for structure in structures:
        print(structure)

    apply_user_transformations(structures)

    print("\nFinal Point Structures:")
    for structure in structures:
        print(structure)

#if __name__ == "__main__":
#    main()
if(entrance == 27):
    while(5 > 2):
        intys27 = int(input("0 for main menu, 1 to continue: "))
        if(intys27 == 0):
            break
        if(intys27 == 1):
            #import os

            def define_encodings():
                encodings = {}
                if os.path.exists('langc.txt'):
                    encodings = load_encodings()

            print("Enter the visual encoding for each character. Type 'done' when finished.")
            while True:
                char = input("Enter character (or type 'done' to finish): ").strip()
                if char.lower() == 'done':
                    break
                if len(char) != 1:
```

```
print("Please enter a single character.")
continue
encoding = input(f"Enter encoding for '{char}': ").strip()
encodings[char] = encoding

# Save encodings to langc.txt
with open('langc.txt', 'w', encoding='utf-8') as f:
    for char, encoding in encodings.items():
        f.write(f"{char}:{encoding}\n")

print("Encodings saved to langc.txt")

def load_encodings(file_path='langc.txt'):
    encodings = {}
    if not os.path.exists(file_path):
        return encodings
    with open(file_path, 'r', encoding='utf-8') as f:
        for line in f:
            char, encoding = line.strip().split(':')
            encodings[char] = encoding
    return encodings

def process_input_string(input_file='input_s.txt', output_file='encode_out.txt'):
    if not os.path.exists(input_file):
        print(f"Input file '{input_file}' not found.")
        return

    encodings = load_encodings()
    if not encodings:
        print("No encodings found. Please define encodings first.")
        return

    with open(input_file, 'r', encoding='utf-8') as infile, open(output_file, 'w', encoding='utf-8') as outfile:
        for line in infile:
            encoded_line = []
            for char in line:
                if char in encodings:
                    encoded_line.append(encodings[char])
                else:
                    encoded_line.append('0') # Unrecognized character
            outfile.write(''.join(encoded_line) + '\n')

    print(f"Encoded output saved to {output_file}")

def view_encodings():
    encodings = load_encodings()
    if not encodings:
        print("No encodings found.")
    return
```

```

print("Current encodings:")
for char, encoding in encodings.items():
    print(f"'{char}': {encoding}")

def clear_encodings():
    if os.path.exists('langc.txt'):
        os.remove('langc.txt')
    print("Encodings cleared.")

def main_menus():
    while True:
        print("\nMenu:")
        print("1. Define Encodings")
        print("2. Process Input String (input_s.txt)")
        print("3. View Encodings")
        print("4. Clear All Encodings")
        print("5. Print Help Documentation")
        print("6. Exit")

    choice = input("Enter your choice: ").strip()
    if choice == '1':
        define_encodings()
    elif choice == '2':
        input_file = input("Enter the input file name (default: input_s.txt): ").strip() or 'input_s.txt'
        output_file = input("Enter the output file name (default: encode_out.txt): ").strip() or 'encode_out.txt'
        process_input_string(input_file, output_file)
    elif choice == '3':
        view_encodings()
    elif choice == '4':
        clear_encodings()
    elif choice == '6':
        print("Exiting...")
        break
    elif choice == '5':
        print("""Help Documentation ...

```

Example Visual Numerical Mapping For Reading Any Language
via Pattern Recognition.

```

1 /
2 \
3 |
4 -
5 .
6 )
7 (
8 Reading Order (Top to Bottom, Left to Right)

```

```

        """
    )
else:
    print("Invalid choice. Please try again.")

#if __name__ == "__main__":
main_menus()

if(entrance == 28):
    while(5 > 2):
        intys28 = int(input("0 for main menu, 1 to continue: "))
        if(intys28 == 0):
            break
        if(intys28 == 1):
            # Predefined color palette with 12 unique colors
            COLOR_PALETTE = [
                '#FF5733', '#33FF57', '#3357FF', '#F1C40F', '#8E44AD',
                '#1ABC9C', '#E74C3C', '#2ECC71', '#3498DB', '#9B59B6',
                '#FF33A1', '#A1FF33'
            ]
class App:
    def __init__(self, master):
        self.master = master
        master.title("Hex Color Picker")

        self.label = Label(master, text="Choose Text File")
        self.label.pack()

        self.open_file_button = Button(master, text="Open File", command=self.open_file)
        self.open_file_button.pack()

        self.close_button = Button(master, text="Generate Image", command=self.generate_image)
        self.close_button.pack()

        self.save_pdf_button = Button(master, text="Save as PDF", command=self.save_as_pdf)
        self.save_pdf_button.pack()

        self.data = []
        self.color_encoding = {}

    def open_file(self):
        file_path = filedialog.askopenfilename(filetypes=[("Text files", "*.txt"), ("All files", "*.*")])
        if file_path:
            try:
                with open(file_path, 'r') as file:
                    self.data = [float(num) for num in file.read().split()]
                    self.label.config(text="File successfully opened.")
            except Exception as e:
                self.label.config(text=f"Error opening file: {e}")
        else:

```

```

        self.label.config(text="No file selected.")

def generate_image(self):
    print("Generating ...")
    if hasattr(self, 'data') and self.data:
        self.label.config(text="Generating, please wait...")
        self.master.update_idletasks() # Force update of the GUI

    # Convert data to colors
    color_data = [self.data_to_color(datum) for datum in self.data]

    self.label.config(text="Data converted to colors. Creating image...")
    self.master.update_idletasks() # Force update of the GUI

    create_image(color_data, self.data)

    self.label.config(text="Image successfully generated.")
else:
    self.label.config(text="Please select a file first.")

def data_to_color(self, datum):
    # Perform the modular arithmetic-based color encoding
    first_digit = int(str(datum)[0])-1
    fractional_part = int(str(datum).split('.')[1])
    cpoint = fractional_part % 12
    pointer = (first_digit + cpoint) % 12
    return COLOR_PALETTE[pointer]

def save_as_pdf(self):
    if hasattr(self, 'generated_image'):
        file_path = filedialog.asksaveasfilename(defaultextension='.pdf', filetypes=[("PDF files",
        "*.pdf")])
        if file_path:
            # A5 dimensions in pixels (previously calculated)
            a5_width_pixels = 1748
            a5_height_pixels = 2480

            # Call the function to save the image as a PDF
            save_image_as_pdf(self.generated_image, file_path, a5_width_pixels,
            a5_height_pixels)
            self.label.config(text="Saved as PDF.")
        else:
            self.label.config(text="No file selected.")
    else:
        self.label.config(text="Generate an image first.")

def create_image(color_data, data):
    # Count unique string encodings
    unique_encodings = set(data)
    num_unique_encodings = len(unique_encodings)

```

```

total_encodings = len(data) # This will count all the encodings
print(f"Number of unique encodings: {num_unique_encodings}")
print(f"Total number of encodings: {total_encodings}")

# Request image dimensions and pixel size from user
pixel_side_length = int(input("Enter the square side length of each pixel to be created: "))
num_pixels_width = int(input("Enter the pixel width of the image, by number of pixels: "))
num_pixels_height = int(input("Enter the pixel height of the image, by number of pixels: "))

img_width = pixel_side_length * num_pixels_width
img_height = pixel_side_length * num_pixels_height

# Create an empty white canvas
image = Image.new('RGB', (img_width, img_height), color="white")

# Create a drawing context for the image
draw = ImageDraw.Draw(image)

# Save the image in the object for later PDF conversion
app.generated_image = image

data_idx = 0 # Index to track data position
for i in range(num_pixels_height):
    for j in range(num_pixels_width):
        if data_idx < len(color_data):
            hex_value = color_data[data_idx]
            top_left = (j * pixel_side_length, i * pixel_side_length)
            bottom_right = ((j + 1) * pixel_side_length, (i + 1) * pixel_side_length)
            draw.rectangle([top_left, bottom_right], fill=hex_value)
            data_idx += 1

image.show()
image.save('generated_image.png')

return image

def save_image_as_pdf(image, file_path, a5_width_pixels, a5_height_pixels, margin_mm=10):
    margin_pixels = int(margin_mm * (300 / 25.4))
    drawable_width = a5_width_pixels - (2 * margin_pixels)
    drawable_height = a5_height_pixels - (2 * margin_pixels)
    num_pages_horizontal = math.ceil(image.width / drawable_width)
    num_pages_vertical = math.ceil(image.height / drawable_height)

    pdf_pages = []

    for y in range(num_pages_vertical):
        for x in range(num_pages_horizontal):
            left = x * drawable_width
            upper = y * drawable_height
            right = min(left + drawable_width, image.width)

```

```

lower = min(upper + drawable_height, image.height)
crop_area = (left, upper, right, lower)

cropped_image = image.crop(crop_area)
pdf_page = Image.new('RGB', (a5_width_pixels, a5_height_pixels), 'white')

paste_position = (
    margin_pixels + max(0, (drawable_width - cropped_image.width) // 2),
    margin_pixels + max(0, (drawable_height - cropped_image.height) // 2)
)

pdf_page.paste(cropped_image, paste_position)
pdf_pages.append(pdf_page)

pdf_pages[0].save(file_path, save_all=True, append_images=pdf_pages[1:], resolution=100.0)

root = Tk()
app = App(root)
root.mainloop()

if(entrance == 29):
    while(5 > 2):
        intys29 = int(input("0 for main menu, 1 to continue: "))
        if(intys29 == 0):
            break
        if(intys29 == 1):

class MplCanvas(FigureCanvas):
    def __init__(self, parent=None, width=5, height=4, dpi=100, projection='3d'):
        fig = Figure(figsize=(width, height), dpi=dpi)
        self.axes = fig.add_subplot(111, projection=projection)
        super(MplCanvas, self).__init__(fig)

class ShapeManagerGUI(QtWidgets.QMainWindow):
    def __init__(self):
        super(ShapeManagerGUI, self).__init__()
        self.setWindowTitle("Shape Manager")
        self.setGeometry(100, 100, 1200, 600)
        self.canvas = MplCanvas(self)
        self.setCentralWidget(self.canvas)
        self.shapes = {}
        self.init_ui()

    def init_ui(self):
        self.control_panel = QtWidgets.QWidget(self)
        self.control_layout = QtWidgets.QVBoxLayout()

        self.control_layout.addWidget(QtWidgets.QLabel("Label:"))
        self.label_input = QtWidgets.QLineEdit(self)
        self.control_layout.addWidget(self.label_input)

```

```

self.control_layout.addWidget(QtWidgets.QLabel("Dimension (1D, 2D, 3D):"))
self.dimension_input = QtWidgets.QComboBox(self)
self.dimension_input.addItems(["1D", "2D", "3D"])
self.control_layout.addWidget(self.dimension_input)

self.control_layout.addWidget(QtWidgets.QLabel("Function:"))
self.function_input = QtWidgets.QLineEdit(self)
self.control_layout.addWidget(self.function_input)

self.add_shape_btn = QtWidgets.QPushButton("Add Shape", self)
self.add_shape_btn.clicked.connect(self.add_shape)
self.control_layout.addWidget(self.add_shape_btn)

self.draw_shape_btn = QtWidgets.QPushButton("Draw Shape", self)
self.draw_shape_btn.clicked.connect(self.draw_shape)
self.control_layout.addWidget(self.draw_shape_btn)

self.save_png_btn = QtWidgets.QPushButton("Save as PNG", self)
self.save_png_btn.clicked.connect(self.save_as_png)
self.control_layout.addWidget(self.save_png_btn)

self.layer_shape_btn = QtWidgets.QPushButton("Layer Shapes", self)
self.layer_shape_btn.clicked.connect(self.layer_shapes)
self.control_layout.addWidget(self.layer_shape_btn)

self.save_session_btn = QtWidgets.QPushButton("Save Session", self)
self.save_session_btn.clicked.connect(self.save_session)
self.control_layout.addWidget(self.save_session_btn)

self.load_session_btn = QtWidgets.QPushButton("Load Session", self)
self.load_session_btn.clicked.connect(self.load_session)
self.control_layout.addWidget(self.load_session_btn)

self.doc_panel = QtWidgets.QTextEdit(self)
self.doc_panel.setReadOnly(True)
self.control_layout.addWidget(self.doc_panel)

self.control_panel.setLayout(self.control_layout)
self.dock_widget = QtWidgets.QDockWidget("Control Panel", self)
self.dock_widget.setWidget(self.control_panel)
self.addDockWidget(QtCore.Qt.RightDockWidgetArea, self.dock_widget)

self.update_documentation()

def update_documentation(self):
    doc_text = "Current State Shapes:\n\n"
    for label, shape in self.shapes.items():
        doc_text += f"Label: {label}\nDimension: {shape['dimension']}\nFunction: {shape['function']}\n\n"
    self.doc_panel.setText(doc_text)

```

```

def add_shape(self):
    label = self.label_input.text().strip()
    dimension = self.dimension_input.currentText()
    function = self.function_input.text().strip()
    if label and function:
        self.shapes[label] = {'dimension': dimension, 'function': function}
        QtWidgets.QMessageBox.information(self, "Shape Added", f"Shape '{label}' added
successfully.")
        self.update_documentation()
    else:
        QtWidgets.QMessageBox.warning(self, "Invalid Input", "Please ensure all fields are filled
correctly.")

def draw_shape(self):
    label = self.label_input.text().strip()
    if label in self.shapes:
        shape = self.shapes[label]
        dimension = shape['dimension']
        function = shape['function']
        try:
            self.canvas.axes.clear()
            if dimension == '1D':
                self.draw_1d(function)
            elif dimension == '2D':
                self.draw_2d(function)
            elif dimension == '3D':
                self.draw_3d(function)
            self.canvas.draw()
        except Exception as e:
            QtWidgets.QMessageBox.critical(self, "Error", "Failed to draw shape: " + str(e))
    else:
        QtWidgets.QMessageBox.warning(self, "Shape Not Found", "No shape found with label: "
+ label)

def draw_1d(self, function):
    x = np.linspace(-10, 10, 400)
    y = eval(function, {'x': x, 'np': np})
    self.canvas.axes.plot(x, y)
    self.canvas.axes.set_title("1D Plot")

def draw_2d(self, function):
    x = np.linspace(-5, 5, 100)
    y = np.linspace(-5, 5, 100)
    X, Y = np.meshgrid(x, y)
    Z = eval(function, {'X': X, 'Y': Y, 'np': np})
    self.canvas.axes.contourf(X, Y, Z, cmap='viridis')
    self.canvas.axes.set_title("2D Contour Plot")

def draw_3d(self, function):

```

```

x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = eval(function, {'X': X, 'Y': Y, 'np': np})
self.canvas.axes.plot_surface(X, Y, Z, cmap='viridis')
self.canvas.axes.set_title("3D Surface Plot")

def layer_shapes(self):
    # First, explore the current state shapes
    self.explore_state_shapes(list(self.shapes.keys()))

    # After exploring, prompt the user to enter the layering sequence
    labels, okPressed = QtWidgets.QInputDialog.getText(self, "Layer Shapes", "Enter labels to
layer (separated by space):")
    if okPressed and labels:
        new_label, okPressed = QtWidgets.QInputDialog.getText(self, "New Label", "Enter new
label for the layered shape:")
        if okPressed and new_label:
            try:
                # Check that all shapes to be layered are of the same dimension
                dimensions = [self.shapes[label.strip()]['dimension'] for label in labels.split() if
label.strip() in self.shapes]
                if len(set(dimensions)) > 1:
                    QtWidgets.QMessageBox.critical(self, "Error", "All shapes must have the same
dimension to be layered.")
            return

                dimension = dimensions[0]
                layered_function = ' + '.join(['(' + self.shapes[label.strip()]['function'] + ')' for label in
labels.split() if label.strip() in self.shapes])
                self.shapes[new_label] = {'dimension': dimension, 'function': layered_function}
                QtWidgets.QMessageBox.information(self, "Layered Shape Created", f"Layered
shape '{new_label}' created.")
                self.update_documentation()
            except Exception as e:
                QtWidgets.QMessageBox.critical(self, "Error", str(e))

    # Draw the layered shapes on a single plot
    self.draw_layered_shapes(labels.split())

def draw_layered_shapes(self, labels):
    if labels:
        dimensions = [self.shapes[label.strip()]['dimension'] for label in labels if label.strip() in
self.shapes]
        if len(set(dimensions)) > 1:
            QtWidgets.QMessageBox.critical(self, "Error", "All shapes must have the same
dimension to be layered.")
        return

    dimension = dimensions[0]

```

```

self.canvas.axes.clear()
for label in labels:
    if label.strip() in self.shapes:
        shape = self.shapes[label.strip()]
        function = shape['function']
        try:
            if dimension == '1D':
                self.draw_1d_layered(function)
            elif dimension == '2D':
                self.draw_2d_layered(function)
            elif dimension == '3D':
                self.draw_3d_layered(function)
        except Exception as e:
            QtWidgets.QMessageBox.critical(self, "Error", f"Failed to draw shape
'{label.strip()}': " + str(e))
        self.canvas.draw()

def draw_1d_layered(self, function):
    x = np.linspace(-10, 10, 400)
    y = eval(function, {'x': x, 'np': np})
    self.canvas.axes.plot(x, y)

def draw_2d_layered(self, function):
    x = np.linspace(-5, 5, 100)
    y = np.linspace(-5, 5, 100)
    X, Y = np.meshgrid(x, y)
    Z = eval(function, {'X': X, 'Y': Y, 'np': np})
    self.canvas.axes.contour(X, Y, Z, cmap='viridis')

def draw_3d_layered(self, function):
    x = np.linspace(-5, 5, 100)
    y = np.linspace(-5, 5, 100)
    X, Y = np.meshgrid(x, y)
    Z = eval(function, {'X': X, 'Y': Y, 'np': np})
    self.canvas.axes.plot_surface(X, Y, Z, cmap='viridis')

def explore_state_shapes(self, labels):
    explorer_window = QtWidgets.QWidget()
    explorer_window.setWindowTitle("State Shape Explorer")
    explorer_layout = QtWidgets.QVBoxLayout()

    for label in labels:
        if label.strip() in self.shapes:
            shape = self.shapes[label.strip()]
            dimension = shape['dimension']
            function = shape['function']
            canvas = MplCanvas(explorer_window, width=5, height=4, dpi=100, projection='3d' if
dimension == '3D' else None)
            try:
                if dimension == '1D':

```

```

        self.draw_1d_explorer(function, canvas)
    elif dimension == '2D':
        self.draw_2d_explorer(function, canvas)
    elif dimension == '3D':
        self.draw_3d_explorer(function, canvas)
    explorer_layout.addWidget(canvas)
except Exception as e:
    QtWidgets.QMessageBox.critical(self, "Error", f"Failed to draw shape '{label.strip()}':
" + str(e))

explorer_window.setLayout(explorer_layout)
explorer_window.show()

def draw_1d_explorer(self, function, canvas):
    x = np.linspace(-10, 10, 400)
    y = eval(function, {'x': x, 'np': np})
    canvas.axes.plot(x, y)
    canvas.axes.set_title("1D Plot")

def draw_2d_explorer(self, function, canvas):
    x = np.linspace(-5, 5, 100)
    y = np.linspace(-5, 5, 100)
    X, Y = np.meshgrid(x, y)
    Z = eval(function, {'X': X, 'Y': Y, 'np': np})
    canvas.axes.contourf(X, Y, Z, cmap='viridis')
    canvas.axes.set_title("2D Contour Plot")

def draw_3d_explorer(self, function, canvas):
    x = np.linspace(-5, 5, 100)
    y = np.linspace(-5, 5, 100)
    X, Y = np.meshgrid(x, y)
    Z = eval(function, {'X': X, 'Y': Y, 'np': np})
    canvas.axes.plot_surface(X, Y, Z, cmap='viridis')
    canvas.axes.set_title("3D Surface Plot")

def save_as_png(self):
    label = self.label_input.text().strip()
    if label in self.shapes and self.canvas.figure:
        options = QtWidgets.QFileDialog.Options()
        fileName, _ = QtWidgets.QFileDialog.getSaveFileName(self, "Save Shape as PNG", "", "
"PNG Files (*.png);All Files (*)", options=options)
        if fileName:
            self.canvas.figure.savefig(fileName)
            QtWidgets.QMessageBox.information(self, "Export Successful", f"Shape saved as PNG:
{fileName}")
        else:
            QtWidgets.QMessageBox.warning(self, "Shape Not Found", "No shape found with label: "
+ label)

def save_session(self):

```

```

options = QtWidgets.QFileDialog.Options()
fileName, _ = QtWidgets.QFileDialog.getSaveFileName(self, "Save Session", "", "Session
Files (*.pkl);;All Files (*)", options=options)
if fileName:
    with open(fileName, 'wb') as f:
        pickle.dump(self.shapes, f)
    QtWidgets.QMessageBox.information(self, "Session Saved", "Session saved successfully.")

def load_session(self):
    options = QtWidgets.QFileDialog.Options()
    fileName, _ = QtWidgets.QFileDialog.getOpenFileName(self, "Load Session", "", "Session
Files (*.pkl);;All Files (*)", options=options)
    if fileName:
        try:
            with open(fileName, 'rb') as f:
                self.shapes = pickle.load(f)
            QtWidgets.QMessageBox.information(self, "Session Loaded", "Session loaded
successfully.")
            self.update_documentation()
        except FileNotFoundError:
            QtWidgets.QMessageBox.warning(self, "Session Not Found", "No session found.")

def main():
    app = QtWidgets.QApplication(sys.argv)
    main_window = ShapeManagerGUI()
    main_window.show()
    sys.exit(app.exec_())

#if __name__ == "__main__":
#    main()

if(entrance == 30):
    while(5 > 2):
        intys30 = int(input("0 for main menu, 1 to continue: "))
        if(intys30 == 0):
            break
        if(intys30 == 1):
            class CharacterMapper(QWidget):
                def __init__(self):
                    super().__init__()

                # Initialize the character map from a text file and add special characters
                self.char_map = self.load_characters_from_file('alpha.txt')
                self.add_special_characters()

                # Reverse map for character to number conversion
                self.reverse_char_map = {v: k for k, v in self.char_map.items()}

            # Now initialize the UI
            self.initUI()

```

```
def initUI(self):
    self.setWindowTitle('Multi-Language Character Mapper')

    self.layout = QVBoxLayout()

    self.string_input_label = QLabel('Enter String to Convert to Number Sequence:', self)
    self.layout.addWidget(self.string_input_label)

    self.string_input_field = QTextEdit(self)
    self.layout.addWidget(self.string_input_field)

    self.convert_button = QPushButton('Convert String to Number Sequence', self)
    self.convert_button.clicked.connect(self.convert_string_to_numbers)
    self.layout.addWidget(self.convert_button)

    self.input_label = QLabel('Enter Number Sequence (comma-separated):', self)
    self.layout.addWidget(self.input_label)

    self.input_field = QLineEdit(self)
    self.layout.addWidget(self.input_field)

    self.map_button = QPushButton('Map to Characters', self)
    self.map_button.clicked.connect(self.show_characters)
    self.layout.addWidget(self.map_button)

    self.result_label = QLabel('Mapped Characters:', self)
    self.layout.addWidget(self.result_label)

    self.result_scroll_area = QScrollArea(self)
    self.result_scroll_area.setWidgetResizable(True)
    self.result_scroll_content = QWidget()
    self.result_scroll_layout = QVBoxLayout(self.result_scroll_content)

    self.result_text_label = QLabel("", self.result_scroll_content)
    self.result_text_label.setWordWrap(True)
    self.result_scroll_layout.addWidget(self.result_text_label)

    self.result_scroll_area.setWidget(self.result_scroll_content)
    self.layout.addWidget(self.result_scroll_area)

    self.available_chars_label = QLabel('Available Characters:', self)
    self.layout.addWidget(self.available_chars_label)

    self.chars_scroll_area = QScrollArea(self)
    self.chars_scroll_area.setWidgetResizable(True)
    self.chars_scroll_content = QWidget()
    self.chars_scroll_layout = QVBoxLayout(self.chars_scroll_content)

    self.chars_text_label = QLabel("", self.chars_scroll_content)
    self.chars_text_label.setWordWrap(True)
```

```

        self.chars_scroll_layout.addWidget(self.chars_text_label)

        self.chars_scroll_area.setWidget(self.chars_scroll_content)
        self.layout.addWidget(self.chars_scroll_area)

        self.setLayout(self.layout)

        self.update_available_characters()
        self.show()

def load_characters_from_file(self, filename):
    char_map = {}
    try:
        with open(filename, 'r', encoding='utf-8') as file:
            for idx, line in enumerate(file):
                character = line.strip()
                if character: # Skip empty lines
                    char_map[idx + 1] = character
    except FileNotFoundError:
        print(f"File {filename} not found. Ensure the file is in the correct path.")
    except Exception as e:
        print(f"An error occurred while reading the file: {e}")
    return char_map

def add_special_characters(self):
    special_chars = {
        ' ': 'SPACE',
        '\t': 'TAB',
        '\n': 'NEWLINE',
        '\0': 'NULL'
    }
    for key, value in special_chars.items():
        self.char_map[len(self.char_map) + 1] = key

def convert_string_to_numbers(self):
    input_text = self.string_input_field.toPlainText()
    number_sequence = []
    for char in input_text:
        if char in self.reverse_char_map:
            number_sequence.append(str(self.reverse_char_map[char]))
    self.input_field.setText(','.join(number_sequence))

def show_characters(self):
    input_text = self.input_field.text()
    try:
        # Split the input text by comma and convert to integers
        numbers = [int(num.strip()) for num in input_text.split(',')]
        # Map each number to the corresponding character
        characters = [self.char_map.get(num, 'Character not found') for num in numbers]
        # Join the characters to form the result string
    
```

```
result = ''.join(characters)
self.result_text_label.setText(f'Characters: {result}')
except ValueError:
    self.result_text_label.setText('Please enter valid numbers separated by commas')

def update_available_characters(self):
    available_chars = ', '.join(f'{key}: {value}' for key, value in self.char_map.items())
    self.chars_text_label.setText(available_chars)

#if __name__ == '__main__':
app = QApplication(sys.argv)
ex = CharacterMapper()
sys.exit(app.exec_())
```

README.md |

```
# cpv
# A Python application for generating quantum circuits and performing arithmetic geometry using state
shapes, and much more.
# Amalec: A Comprehensive Multi-Tool Python Application
```

Overview

Welcome to Amalec, a versatile Python application that integrates a variety of tools and features designed to assist with image generation, programming, cosmology, quantum computing, classical circuit design, language generation, and more.

Amalec is created to solve problems across different domains with a single, unified application. Developed by Mr. Dominic Alexander Cooper, Amalec aims to provide users with a robust set of tools for both professional and educational purposes.

Features

Here are the features and functionalities offered by Amalec:

0. Image Generator

****Description**:** Generates images which are required for the use of other features in the application, particularly the Programming Engine (Option 2).

1. Image Compiler

****Description**:** Compiles multiple images into a single file or presentation.

2. Programming Engine

****Description**:** A comprehensive engine for programming tasks, leveraging the images generated by the Image Generator.

3. Cosmology

****Description**:** Tools and functionalities for cosmological research and simulations. Can be used in conjunction with Options 0, 1, and 2.

4. Sequential Language Generation

Description: Generates sequences of language. **Warning**: This tool must filter out profane content. A user `.txt` file must exist within the working directory, listing Unicode characters (one per line).

5. Text Editor

Description: A built-in text editor for writing and editing text files.

6. Generate Command Matrix Plugin for the Programming Engine

Description: Creates command matrix plugins to extend the capabilities of the Programming Engine.

7. Exit Program

Description: Exits the Amalec application.

8. 3D CAD CAM Engine

Description: A 3D CAD/CAM engine for computer-aided design and manufacturing.

9. Quantum Circuit Qiskit WorkBench

Description: A workbench for creating and simulating quantum circuits using Qiskit.

10. Classical Circuit SchemDraw WorkBench

Description: A workbench for designing and simulating classical circuits using SchemDraw.

11. Cosmological Simulation by Text

Description: Simulates cosmological phenomena using text-based inputs and outputs.

12. Large Language Model

Description: Integrates a large language model for various language processing tasks.

13. Data Script by Custom Structures

Description: Generates data scripts using custom data structures.

14. Generate Binary Unique Strings to a custom.txt File

Description: Creates unique binary strings and saves them to a custom `'.txt` file.

16. Convert .outb File to BinaryOut File (Extension of Option 12)

Description: Converts `'.outb` files to `binaryout` files, extending the functionality of the Large Language Model.

17. Text Database

Description: Manages and manipulates text databases.

18. Proximal Prime Numbers

Description: Calculates and analyzes proximal prime numbers.

19. Configurations for Mode 2

Description: Provides configuration settings for the Programming Engine (Option 2).

21. Proximal Primes Animator

Description: Animates the behavior and distribution of proximal prime numbers.

22. Manual Turing Complete Finite State Machine

Description: Allows the user to manually construct a Turing-complete finite state machine.

23. Automatic Turing Complete Finite State Machine

Description: Automatically generates a Turing-complete finite state machine.

24. Python Enabled Command Line Interface

Description: Provides a command line interface with Python capabilities.

25. Automatic Control Engineering

Description: Tools and functionalities for automatic control engineering.

26. Automated Design and Technology

Description: Facilitates automated design and technological development tasks.

27. Construct Your Own Linguistic Language

Description: Enables users to construct their own linguistic languages.

28. Convert Text Encoding to a Convergent Image (From Option 12)

Description: Converts text encoding, derived from the Large Language Model, into a convergent image.

29. Mathematics Lab

Description: A lab environment for conducting mathematical experiments and visualizations.

Installation

1. **Clone the repository**:

```sh

git clone https://github.com/DOMIAEGDE/cpy.git

2. \*\*Navigate to the directory in terminal or console.\*\*

3. \*\*pip install -r requirements.txt\*\*

## License

This project is licensed under the MIT License- see the [LICENSE](LICENSE) file for details.

setup.py |

```
from setuptools import setup, find_packages
```

```
setup(
```

```
 name="Amalec",
```

```
 version="0.1.0",
```

```
 packages=find_packages(),
```

```
 install_requires=[
```

```
 "beautifulsoup4",
```

```
 "keyboard",
```

```
 "matplotlib",
```

```
 "numpy",
```

```
 "pygame",
```

```
 "Pillow",
```

```
"plotly",
"psutil",
"pyautogui",
"PyQt5",
"qiskit",
"requests",
"schemdraw",
"schedule",
],
entry_points={
 'console_scripts': [
 'quantum_arithmetic=C:main',
],
},
author="Mr. Dominic Alexander Cooper",
author_email="dacgde.cooper@gmail.com",
description="A Python application for generating quantum circuits and performing arithmetic geometry using state shapes, and much more.",
license="MIT",
keywords="quantum circuits arithmetic geometry image generation cosmology language generation",
url="https://github.com/domiaxegde/cpy",
)
```

requirements.txt |

beautifulsoup4  
keyboard  
matplotlib  
numpy  
pygame  
Pillow  
plotly  
psutil  
pyautogui  
PyQt5  
qiskit  
requests  
schemdraw  
schedule

Coordinates\_Python.txt |

1,0,0  
2,0,1  
3,0,2  
4,0,3  
5,0,4  
6,0,5  
7,0,6  
8,0,7

9,0,8  
10,0,9  
11,0,10  
12,0,11  
13,0,12  
14,0,13  
15,0,14  
16,0,15  
17,1,0  
18,1,1  
19,1,2  
20,1,3  
21,1,4  
22,1,5  
23,1,6  
24,1,7  
25,1,8  
26,1,9  
27,1,10  
28,1,11  
29,1,12  
30,1,13  
31,1,14  
32,1,15  
33,2,0  
34,2,1  
35,2,2  
36,2,3  
37,2,4  
38,2,5  
39,2,6  
40,2,7  
41,2,8  
42,2,9  
43,2,10  
44,2,11  
45,2,12  
46,2,13  
47,2,14  
48,2,15  
49,3,0  
50,3,1  
51,3,2  
52,3,3  
53,3,4  
54,3,5  
55,3,6  
56,3,7  
57,3,8  
58,3,9

59,3,10  
60,3,11  
61,3,12  
62,3,13  
63,3,14  
64,3,15  
65,4,0  
66,4,1  
67,4,2  
68,4,3  
69,4,4  
70,4,5  
71,4,6  
72,4,7  
73,4,8  
74,4,9  
75,4,10  
76,4,11  
77,4,12  
78,4,13  
79,4,14  
80,4,15  
81,5,0  
82,5,1  
83,5,2  
84,5,3  
85,5,4  
86,5,5  
87,5,6  
88,5,7  
89,5,8  
90,5,9  
91,5,10  
92,5,11  
93,5,12  
94,5,13  
95,5,14  
96,5,15  
97,6,0  
98,6,1  
99,6,2  
100,6,3  
101,6,4  
102,6,5  
103,6,6  
104,6,7  
105,6,8  
106,6,9  
107,6,10  
108,6,11

109,6,12  
110,6,13  
111,6,14  
112,6,15  
113,7,0  
114,7,1  
115,7,2  
116,7,3  
117,7,4  
118,7,5  
119,7,6  
120,7,7  
121,7,8  
122,7,9  
123,7,10  
124,7,11  
125,7,12  
126,7,13  
127,7,14  
128,7,15  
129,8,0  
130,8,1  
131,8,2  
132,8,3  
133,8,4  
134,8,5  
135,8,6  
136,8,7  
137,8,8  
138,8,9  
139,8,10  
140,8,11  
141,8,12  
142,8,13  
143,8,14  
144,8,15  
145,9,0  
146,9,1  
147,9,2  
148,9,3  
149,9,4  
150,9,5  
151,9,6  
152,9,7  
153,9,8  
154,9,9  
155,9,10  
156,9,11  
157,9,12  
158,9,13

159,9,14  
160,9,15  
161,10,0  
162,10,1  
163,10,2  
164,10,3  
165,10,4  
166,10,5  
167,10,6  
168,10,7  
169,10,8  
170,10,9  
171,10,10  
172,10,11  
173,10,12  
174,10,13  
175,10,14  
176,10,15  
177,11,0  
178,11,1  
179,11,2  
180,11,3  
181,11,4  
182,11,5  
183,11,6  
184,11,7  
185,11,8  
186,11,9  
187,11,10  
188,11,11  
189,11,12  
190,11,13  
191,11,14  
192,11,15  
193,12,0  
194,12,1  
195,12,2  
196,12,3  
197,12,4  
198,12,5  
199,12,6  
200,12,7  
201,12,8  
202,12,9  
203,12,10  
204,12,11  
205,12,12  
206,12,13  
207,12,14  
208,12,15

209,13,0  
210,13,1  
211,13,2  
212,13,3  
213,13,4  
214,13,5  
215,13,6  
216,13,7  
217,13,8  
218,13,9  
219,13,10  
220,13,11  
221,13,12  
222,13,13  
223,13,14  
224,13,15  
225,14,0  
226,14,1  
227,14,2  
228,14,3  
229,14,4  
230,14,5  
231,14,6  
232,14,7  
233,14,8  
234,14,9  
235,14,10  
236,14,11  
237,14,12  
238,14,13  
239,14,14  
240,14,15  
241,15,0  
242,15,1  
243,15,2  
244,15,3  
245,15,4  
246,15,5  
247,15,6  
248,15,7  
249,15,8  
250,15,9  
251,15,10  
252,15,11  
253,15,12  
254,15,13  
255,15,14  
256,15,15

maincpp-output.py |

```
import json
from reportlab.lib.pagesizes import letter
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, PageBreak

def read_json_file(filename):
 with open(filename, 'r') as file:
 data = json.load(file)
 return data

def create_pdf(data, output_filename):
 doc = SimpleDocTemplate(output_filename, pagesize=letter)
 styles = getSampleStyleSheet()
 custom_styles = {
 "title": ParagraphStyle(
 'Title',
 parent=styles['Title'],
 fontSize=24,
 spaceAfter=12
),
 "body": ParagraphStyle(
 'BodyText',
 parent=styles['BodyText'],
 fontSize=12,
 spaceAfter=12,
 leading=15
),
 "subtitle": ParagraphStyle(
 'Subtitle',
 parent=styles['Title'],
 fontSize=18,
 spaceAfter=6
)
 }
 story = []

 for structure_id, details in data.items():
 story.append(Paragraph(structure_id, custom_styles['title']))
 story.append(Spacer(1, 12))
 story.append(Paragraph(f"Timestamp: {details['timestamp']}", custom_styles['subtitle']))
 story.append(Spacer(1, 12))

 properties = details["properties"]
 for key, value in properties.items():
 # Handle multi-line property values by replacing newlines with
 tags
 if isinstance(value, str):
 value = value.replace("\n", '
')
 story.append(Paragraph(f"{{key}}: {{value}}", custom_styles['body']))

 doc.build(story)
```

```
story.appendSpacer(1, 12))

story.appendPageBreak()

doc.build(story)

if __name__ == "__main__":
 input_filename = input("Enter input filename.json (With extension): ")
 json_data = read_json_file(input_filename)

 # Print JSON data to verify structure
 print(json.dumps(json_data, indent=4))

 create_pdf(json_data, 'output.pdf')

main.cpp |

/*
 * Installation Instructions for Mathematicion
 *
 * Prerequisites:
 * 1. C++ Compiler (g++ or Visual Studio)
 * 2. CMake (if using a CMake project)
 * 3. vcpkg (for managing C++ libraries)
 * 4. JSON for Modern C++ library (nlohmann/json)
 *
 * Step 1: Install vcpkg
 * - Clone the vcpkg repository:
 * git clone https://github.com/microsoft/vcpkg.git
 * - Navigate to the vcpkg directory:
 * cd vcpkg
 * - Bootstrap vcpkg:
 * .\bootstrap-vcpkg.bat (on Windows)
 * ./bootstrap-vcpkg.sh (on Unix)
 * - Integrate vcpkg with your shell:
 * .\vcpkg integrate install
 *
 * Step 2: Install JSON for Modern C++ library
 * - Use vcpkg to install the nlohmann-json package:
 * .\vcpkg install nlohmann-json
 *
 * Step 3: Compile the Program
 * - Open a command prompt (cmd) or PowerShell.
 * - Navigate to the directory containing the source code (e.g., main.cpp).
 * - Compile the program using g++:
 * g++-ID:/vcpkgm/vcpkg/installed/x64-windows/include main.cpp-o Mathematicion.exe
 * (Replace D:/vcpkgm/vcpkg/installed/x64-windows/include with the correct path to your vcpkg
 * installation)
 * - If using MSVC (Visual Studio):
 * - Open the Developer Command Prompt for Visual Studio.
```

```

* - Navigate to the directory containing the source code.
* - Compile the program using cl:
* cl /EHsc main.cpp /I D:/vcpkgm/vcpkg/installed/x64-windows/include /link /OUT:Mathematicion.exe
* (Replace D:/vcpkgm/vcpkg/installed/x64-windows/include with the correct path to your vcpkg
installation)
*
* Step 4: Run the Program
* - After successful compilation, run the executable:
* .\Mathematicion.exe
*
* Troubleshooting:
* - Ensure all paths are correct and point to the appropriate directories.
* - Ensure all dependencies are installed and accessible.
* - Check for any compilation errors and resolve missing include paths or libraries.
*
* Example:
* > mkdir config
* > .\Mathematicion.exe
*
* You should now be able to interact with the menu and perform various operations as described in the
program.
*
* For further help and documentation, refer to the official documentation of the JSON for Modern C++
library and vcpkg.
*/
#include <iostream>
#include <string>
#include <set>
#include <unordered_map>
#include <memory>
#include <chrono>
#include <sstream>
#include <iomanip>
#include <fstream>
#include "D:/vcpkgm/vcpkg/installed/x64-windows/include/nlohmann/json.hpp"
#include <filesystem>

using json = nlohmann::json;

// A function to get the current timestamp in the desired format
std::string getCurrentTimestamp() {
 auto now = std::chrono::system_clock::now();
 auto in_time_t = std::chrono::system_clock::to_time_t(now);

 std::stringstream ss;
 ss << std::put_time(std::gmtime(&in_time_t), "%Y-%m-%dT%H:%M:%S");
 return ss.str();
}

// Base class for PS (Property Set)

```

```
class PropertySet {
public:
 virtual void display() const = 0;
 virtual json to_json() const = 0;
 virtual void delete_property(const std::string& key) = 0;
 virtual ~PropertySet() = default;
};

// An example of a user-defined property set using JSON
class DynamicPropertySet : public PropertySet {
private:
 json properties;
public:
 DynamicPropertySet(const json& props) : properties(props) {}

 void display() const override {
 std::cout << properties.dump(4) << std::endl;
 }

 json to_json() const override {
 return properties;
 }

 void delete_property(const std::string& key) override {
 properties.erase(key);
 }
};

// Cooper class to encapsulate the structure
class Cooper {
private:
 std::string id;
 std::unique_ptr<PropertySet> propertySet;
 std::string timestamp;

 static std::set<std::string> idSet;

public:
 Cooper(std::string id, std::unique_ptr<PropertySet> ps)
 : id(std::move(id)), propertySet(std::move(ps)), timestamp(getCurrentTimestamp()) {
 if (idSet.find(this->id) != idSet.end()) {
 throw std::invalid_argument("ID already exists.");
 }
 idSet.insert(this->id);
 }

 const std::string& getId() const { return id; }
 const std::string& getTimestamp() const { return timestamp; }
 const PropertySet* getPropertySet() const { return propertySet.get(); }
}
```

```

void display() const {
 std::cout << "ID: " << id << std::endl;
 std::cout << "Timestamp: " << timestamp << std::endl;
 std::cout << "Properties: ";
 propertySet->display();
}

static void resetIdSet() {
 idSet.clear();
}

static void updateIdSet(const std::set<std::string>& newIdSet) {
 idSet = newIdSet;
}

void save(const std::string& filename) const {
 json j;
 std::ifstream inFile(filename);
 if (inFile.is_open()) {
 inFile >> j;
 inFile.close();
 }

 j[id]["timestamp"] = timestamp;
 j[id]["properties"] = propertySet->to_json();

 std::ofstream outFile(filename);
 if (outFile.is_open()) {
 outFile << j.dump(4);
 }
}

static std::unordered_map<std::string, std::unique_ptr<Cooper>> loadAll(const std::string& filename) {
 std::ifstream file(filename);
 if (!file.is_open()) {
 throw std::runtime_error("Could not open file for reading");
 }
 json j;
 file >> j;

 std::unordered_map<std::string, std::unique_ptr<Cooper>> cooperMap;

 for (auto& [id, data] : j.items()) {
 std::string timestamp = data["timestamp"];
 json properties = data["properties"];

 auto propertySet = std::make_unique<DynamicPropertySet>(properties);
 auto cooper = std::make_unique<Cooper>(id, std::move(propertySet));
 cooper->timestamp = timestamp; // Ensure the timestamp is preserved
 }
}

```

```

 cooperMap[id] = std::move(cooper);
 }

 return cooperMap;
}

void updatePropertySet(const json& newProps) {
 propertySet = std::make_unique<DynamicPropertySet>(newProps);
}

void deleteProperty(const std::string& key) {
 propertySet->delete_property(key);
}

void changeId(const std::string& newId) {
 if (idSet.find(newId) != idSet.end()) {
 throw std::invalid_argument("ID already exists.");
 }
 idSet.erase(id);
 id = newId;
 idSet.insert(id);
}
};

std::set<std::string> Cooper::idSet;

void displayMenu() {
 std::cout << "Menu:\n";
 std::cout << "1. Create new structure\n";
 std::cout << "2. Display structure\n";
 std::cout << "3. Update structure properties\n";
 std::cout << "4. Delete structure\n";
 std::cout << "5. Change structure ID\n";
 std::cout << "6. Save structure to file\n";
 std::cout << "7. Load structure from file\n";
 std::cout << "8. Delete a property from structure\n";
 std::cout << "9. Help documentation\n";
 std::cout << "10. Exit\n";
}

void helpDocumentation() {
 std::cout << "Help Documentation:\n";
 std::cout << "1. Create new structure: Allows you to create a new mathematical structure by entering properties interactively.\n";
 std::cout << "2. Display structure: Displays the details of a selected structure.\n";
 std::cout << "3. Update structure properties: Allows you to update the properties of an existing structure.\n";
 std::cout << "4. Delete structure: Deletes an existing structure from the system.\n";
 std::cout << "5. Change structure ID: Changes the ID of an existing structure, ensuring no duplicates.\n";
 std::cout << "6. Save structure to file: Saves the details of a selected structure to a JSON file.\n";
}

```

```

 std::cout << "7. Load structure from file: Loads a structure from a JSON file and adds it to the system.\n";
 std::cout << "8. Delete a property from structure: Deletes a specific property from an existing
structure.\n";
 std::cout << "9. Help documentation: Displays this help documentation.\n";
 std::cout << "10. Exit: Exits the application.\n";
}

json getUserProperties() {
 json userProps;
 std::string key;
 std::string value;
 bool booleanValue;
 int intValue;

 std::cout << "Enter the properties for the structure. Type 'done' when finished." << std::endl;
 while (true) {
 std::cout << "Enter property name (or type 'done' to finish): ";
 std::getline(std::cin, key);
 if (key == "done") break;

 std::cout << "Enter property type (string, int, bool): ";
 std::string type;
 std::getline(std::cin, type);

 if (type == "string") {
 std::cout << "Enter string value (type 'END' to finish multi-line input): ";
 std::string line;
 value.clear();
 while (std::getline(std::cin, line)) {
 if (line == "END") break;
 value += line + "\n";
 }
 userProps[key] = value;
 } else if (type == "int") {
 std::cout << "Enter integer value: ";
 std::cin >> intValue;
 userProps[key] = intValue;
 std::cin.ignore(); // Ignore the newline character left in the input buffer
 } else if (type == "bool") {
 std::cout << "Enter boolean value (0 for false, 1 for true): ";
 std::cin >> booleanValue;
 userProps[key] = booleanValue;
 std::cin.ignore(); // Ignore the newline character left in the input buffer
 } else {
 std::cout << "Invalid type. Please enter 'string', 'int', or 'bool'." << std::endl;
 }
 }

 return userProps;
}

```

```

int main() {
 try {
 std::unordered_map<std::string, std::unique_ptr<Cooper>> cooperMap;
 int choice;
 std::string id;
 std::string filename;

 while (true) {
 displayMenu();
 std::cout << "Enter your choice: ";
 std::cin >> choice;
 std::cin.ignore(); // Ignore the newline character left in the input buffer

 switch (choice) {
 case 1: {
 std::cout << "Enter structure ID: ";
 std::getline(std::cin, id);
 auto props = getUserProperties();
 cooperMap[id] = std::make_unique<Cooper>(id,
std::make_unique<DynamicPropertySet>(props));
 break;
 }
 case 2: {
 std::cout << "Enter structure ID to display: ";
 std::getline(std::cin, id);
 if (cooperMap.find(id) != cooperMap.end()) {
 cooperMap[id]->display();
 } else {
 std::cout << "Structure ID not found." << std::endl;
 }
 break;
 }
 case 3: {
 std::cout << "Enter structure ID to update: ";
 std::getline(std::cin, id);
 if (cooperMap.find(id) != cooperMap.end()) {
 auto newProps = getUserProperties();
 cooperMap[id]->updatePropertySet(newProps);
 } else {
 std::cout << "Structure ID not found." << std::endl;
 }
 break;
 }
 case 4: {
 std::cout << "Enter structure ID to delete: ";
 std::getline(std::cin, id);
 if (cooperMap.find(id) != cooperMap.end()) {
 cooperMap.erase(id);
 std::set<std::string> newIdSet;
 }
 }
 }
 }
 }
}

```

```

 for (const auto& pair : cooperMap) {
 newIdSet.insert(pair.first);
 }
 Cooper::updateIdSet(newIdSet);
 } else {
 std::cout << "Structure ID not found." << std::endl;
 }
 break;
}
case 5: {
 std::cout << "Enter current structure ID: ";
 std::getline(std::cin, id);
 if (cooperMap.find(id) != cooperMap.end()) {
 std::cout << "Enter new structure ID: ";
 std::string newId;
 std::getline(std::cin, newId);
 cooperMap[id]->changeId(newId);
 cooperMap[newId] = std::move(cooperMap[id]);
 cooperMap.erase(id);
 } else {
 std::cout << "Structure ID not found." << std::endl;
 }
 break;
}
case 6: {
 std::cout << "Enter structure ID to save: ";
 std::getline(std::cin, id);
 if (cooperMap.find(id) != cooperMap.end()) {
 std::cout << "Enter filename to save to: ";
 std::getline(std::cin, filename);
 cooperMap[id]->save("config/" + filename);
 } else {
 std::cout << "Structure ID not found." << std::endl;
 }
 break;
}
case 7: {
 std::cout << "Enter filename to load from: ";
 std::getline(std::cin, filename);
 cooperMap = Cooper::loadAll("config/" + filename);
 break;
}
case 8: {
 std::cout << "Enter structure ID: ";
 std::getline(std::cin, id);
 if (cooperMap.find(id) != cooperMap.end()) {
 std::string propertyKey;
 std::cout << "Enter property name to delete: ";
 std::getline(std::cin, propertyKey);
 cooperMap[id]->deleteProperty(propertyKey);
 }
}

```

```

 } else {
 std::cout << "Structure ID not found." << std::endl;
 }
 break;
}
case 9: {
 helpDocumentation();
 break;
}
case 10: {
 std::cout << "Exiting program." << std::endl;
 return 0;
}
default: {
 std::cout << "Invalid choice. Please try again." << std::endl;
}
}
}

} catch (const std::exception& e) {
 std::cerr << e.what() << std::endl;
}

return 0;
}

```

MechanicalComputer.py |

```

import FreeCAD, Part

Create a new document
doc = FreeCAD.newDocument("Stainless_Steel_Slab")

Create the stainless steel slab
slab = Part.makeBox(135, 63, 6)
slab_obj = doc.addObject("Part::Feature", "Slab")
slab_obj.Shape = slab

Create the first slot (one-third down from the top, starting from the top face)
slot1 = Part.makeBox(5, 63, 3, FreeCAD.Vector((135-5)/3, 0, 6- 3))

Create the second slot (two-thirds down from the top, starting from the top face)
slot2 = Part.makeBox(5, 63, 3, FreeCAD.Vector(2*(135-5)/3, 0, 6- 3))

Cut the slots from the slab
cut1 = slab.cut(slot1)
cut2 = cut1.cut(slot2)
slab_obj.Shape = cut2

Recompute the document
doc.recompute()

```

Computer.py |

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches

Slab dimensions
slab_length = 135
slab_width = 63
slab_thickness = 6

Slot dimensions
slot_length = 5
slot_width = 63
slot_depth = 3

Slot positions
slot1_x = (slab_length - slot_length) / 3
slot2_x = 2 * (slab_length - slot_length) / 3

Create figure and axis
fig, ax = plt.subplots(figsize=(10, 6))

Create the slab as a rectangle
slab = patches.Rectangle((0, 0), slab_length, slab_thickness, linewidth=1, edgecolor='black',
facecolor='gray')
ax.add_patch(slab)

Create the first slot
slot1 = patches.Rectangle((slot1_x, slab_thickness - slot_depth), slot_length, slot_depth, linewidth=1,
edgecolor='black', facecolor='white')
ax.add_patch(slot1)

Create the second slot
slot2 = patches.Rectangle((slot2_x, slab_thickness - slot_depth), slot_length, slot_depth, linewidth=1,
edgecolor='black', facecolor='white')
ax.add_patch(slot2)

Set the limits and aspect ratio
ax.set_xlim(-10, slab_length + 10)
ax.set_ylim(-10, slab_thickness + 10)
ax.set_aspect('equal')

Add labels and title
ax.set_xlabel('Length (mm)')
ax.set_ylabel('Thickness (mm)')
ax.set_title('Stainless Steel Slab with Slots')

Show the plot
plt.grid(True)
```

```

plt.show()

Enumerator.py | 1

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
import numpy as np

Slab dimensions
slab_length = 135
slab_width = 63
slab_thickness = 6

Slot dimensions
slot_length = 5
slot_width = 63
slot_depth = 3

Slot positions
slot1_x = (slab_length - slot_length) / 3
slot2_x = 2 * (slab_length - slot_length) / 3
slot_y = 0
slot_z = slab_thickness - slot_depth

Create figure
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

Create the vertices of the slab
vertices = np.array([
 [0, 0, 0], [slab_length, 0, 0], [slab_length, slab_width, 0], [0, slab_width, 0], # Bottom face
 [0, 0, slab_thickness], [slab_length, 0, slab_thickness], [slab_length, slab_width, slab_thickness], [0, slab_width, slab_thickness] # Top face
])

Define the 6 faces of the slab
faces = [
 [vertices[0], vertices[1], vertices[5], vertices[4]], # Front face
 [vertices[1], vertices[2], vertices[6], vertices[5]], # Right face
 [vertices[2], vertices[3], vertices[7], vertices[6]], # Back face
 [vertices[3], vertices[0], vertices[4], vertices[7]], # Left face
 [vertices[0], vertices[1], vertices[2], vertices[3]], # Bottom face
 [vertices[4], vertices[5], vertices[6], vertices[7]] # Top face
]

Add the faces of the slab to the plot
ax.add_collection3d(Poly3DCollection(faces, facecolors='gray', linewidths=1, edgecolors='black', alpha=0.5))

Define the slots as smaller boxes and add them to the plot
slot1_vertices = np.array([

```

```

[slot1_x, slot_y, slot_z], [slot1_x + slot_length, slot_y, slot_z], [slot1_x + slot_length, slot_y + slot_width,
slot_z], [slot1_x, slot_y + slot_width, slot_z],
[slot1_x, slot_y, slab_thickness], [slot1_x + slot_length, slot_y, slab_thickness], [slot1_x + slot_length,
slot_y + slot_width, slab_thickness], [slot1_x, slot_y + slot_width, slab_thickness]
])

slot2_vertices = np.array([
[slot2_x, slot_y, slot_z], [slot2_x + slot_length, slot_y, slot_z], [slot2_x + slot_length, slot_y + slot_width,
slot_z], [slot2_x, slot_y + slot_width, slot_z],
[slot2_x, slot_y, slab_thickness], [slot2_x + slot_length, slot_y, slab_thickness], [slot2_x + slot_length,
slot_y + slot_width, slab_thickness], [slot2_x, slot_y + slot_width, slab_thickness]
])

slot_faces1 = [
[slot1_vertices[0], slot1_vertices[1], slot1_vertices[5], slot1_vertices[4]],
[slot1_vertices[1], slot1_vertices[2], slot1_vertices[6], slot1_vertices[5]],
[slot1_vertices[2], slot1_vertices[3], slot1_vertices[7], slot1_vertices[6]],
[slot1_vertices[3], slot1_vertices[0], slot1_vertices[4], slot1_vertices[7]],
[slot1_vertices[0], slot1_vertices[1], slot1_vertices[2], slot1_vertices[3]],
[slot1_vertices[4], slot1_vertices[5], slot1_vertices[6], slot1_vertices[7]]
]

slot_faces2 = [
[slot2_vertices[0], slot2_vertices[1], slot2_vertices[5], slot2_vertices[4]],
[slot2_vertices[1], slot2_vertices[2], slot2_vertices[6], slot2_vertices[5]],
[slot2_vertices[2], slot2_vertices[3], slot2_vertices[7], slot2_vertices[6]],
[slot2_vertices[3], slot2_vertices[0], slot2_vertices[4], slot2_vertices[7]],
[slot2_vertices[0], slot2_vertices[1], slot2_vertices[2], slot2_vertices[3]],
[slot2_vertices[4], slot2_vertices[5], slot2_vertices[6], slot2_vertices[7]]
]

ax.add_collection3d(Poly3DCollection(slot_faces1, facecolors='white', linewidths=1, edgecolors='black'))
ax.add_collection3d(Poly3DCollection(slot_faces2, facecolors='white', linewidths=1, edgecolors='black'))

Set the limits and labels
ax.set_xlim([0, slab_length])
ax.set_ylim([0, slab_width])
ax.set_zlim([0, slab_thickness])
ax.set_xlabel('Length (mm)')
ax.set_ylabel('Width (mm)')
ax.set_zlabel('Thickness (mm)')
ax.set_title('3D View of Stainless Steel Slab with Slots')

Equal aspect ratio
ax.set_box_aspect([slab_length, slab_width, slab_thickness]) # Aspect ratio is 1:1:1

Show the plot
plt.show()

```

LICENSE |

MIT License

Copyright (c) 2024 DOMIAXEGDE

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

4.66787783 4.68778382 1.27 1.27 1.91 0 4.66631524 0 4.57670806 4.8190011 0 4.83141411 0 4.82190019  
4.8141358 1.27 1.92 0 4.81686467 4.76686312 1.3 1.27 1.93 0 4.18041920 4.15631524 1.27 1.94 0  
4.17041620 4.8170412 4.4131918 4.63192319 1.27 1.95 0 4.66141417 4.3081300 4.19041860 4.79241907  
4.14136319 2.2319 0 4.57861417 2.1018 0 4.22081907 0 3.190704 0 4.66631524 0 4.79171406 4.17001212  
3.81306 0 4.68130608 3.130458 1.27 1.96 0 4.12000813 4.2151559 4.14201915 4.20196315 1.24 1.27 1.97  
0 4.12000813 4.63021515 0 4.57670019 1.0 0 4.82191720 4.2192017 2.418 0 4.76001300 4.6041758 1.27  
1.98 0 4.76040207 4.130802 4.116614 4.12152019 4.4176315 1.24 0 4.57761403 3.41158 1.27 1.99 0  
4.66141215 4.20190417 3.631524 0 4.57761403 3.41158 1.27 2.9190 0 4.68132012 4.4170019 4.14176315  
1.24 0 4.57761403 3.41158 1.27 2.9191 0 4.75726668 3.778268 1.27 4.59595959 4.59595959 4.59595959  
4.59595959 4.59595959 4.59595959 4.59595959 4.59595959 4.59595959 4.59595959 2.5959 1.27  
4.69201302 4.19081413 2.11 0 4.69081104 1.18 0 4.210008 4.11000111 1.1 0 3.1937 1.27 4.7191915  
4.18373232 4.6081907 4.20016302 4.14123267 4.78767264 4.87687067 4.68320215 4.24630608 1.19 1.27  
4.59595959 4.59595959 4.59595959 4.59595959 4.59595959 4.59595959 4.59595959 4.59595959  
4.59595959 4.59595959 2.5959 1.27 1.27 4.66631524 0 1.47 1.27 1.27 4.5171412 0 4.19100813 3.190417 0  
4.8121514 2.1719 0 1.56 1.27 4.8121514 2.1719 0 4.19100813 3.190417 0 2.18 0 2.1910 1.27 4.8121514  
2.1719 0 4.17001303 2.1412 1.27 4.8121514 2.1719 0 2.1418 1.27 4.8121514 2.1719 0 4.12001907 1.27  
4.5171412 0 3.797275 0 4.8121514 2.1719 0 4.72120006 1.4 1.27 4.8121514 2.1719 0 4.18200115  
4.17140204 2.1818 1.27 4.8121514 2.1719 0 2.814 0 1.39 0 4.4231504 4.17081204 2.1319 1.27 4.8121514  
2.1719 0 4.15240600 2.1204 1.27 4.8121514 2.1719 0 4.8121514 4.17191108 1.1 1.27 4.8121514 2.1719 0  
4.17041620 4.4181918 1.27 4.8121514 2.1719 0 4.22040101 4.17142218 2.417 1.27 4.8121514 2.1719 0  
3.21821 1.27 4.5171412 0 3.11894 0 4.8121514 2.1719 0 4.65040020 4.19080520 4.11821420 1.15 1.27  
4.8121514 2.1719 0 4.3001904 4.19081204 1.27 4.8121514 2.1719 0 3.182418 1.27 4.8121514 2.1719 0  
4.9181413 1.27 4.8121514 2.1719 0 2.1704 1.27 4.5171412 0 3.797275 0 4.8121514 2.1719 0 4.72120006  
4.4701700 1.1 1.27 4.8121514 2.1719 0 4.15080210 2.1104 1.27 4.5171412 0 4.19100813 3.190417 0  
4.8121514 2.1719 0 4.18081215 4.11040308 4.111406 1.27 4.5171412 0 4.19100813 3.190417 0 4.8121514  
2.1719 0 4.76041320 1.31 0 4.66070402 4.10012019 4.19141331 0 4.65141411 4.4001385 2.17 0  
4.39691417 0 4.86081303 3.142218 0 2.7882 1.27 1.39 0 4.5171412 0 4.19100813 3.190417 0 4.8121514  
2.1719 0 4.76041320 1.31 0 4.66070402 4.10012019 3.191413 0 4.39691417 0 4.12000278 1.82 1.27 1.39 0  
4.5171412 0 4.19100813 3.190417 0 4.8121514 2.1719 0 4.65141411 4.4001385 2.17 0 4.39691417 0  
4.12000278 1.82 1.27 4.5171412 0 4.19100813 4.19041763 4.2141114 4.17020714 4.14180417 0  
4.8121514 2.1719 0 4.181002 4.14111417 1.27 4.8121514 2.1719 0 4.11140606 3.81306 1.27 4.8121514  
2.1719 0 4.19081204 1.27 4.8121514 2.1719 0 4.15240020 4.19140620 1.8 1.27 4.8121514 2.1719 0  
4.10042401 4.14001703 1.27 4.5171412 0 4.19100813 3.190417 0 4.8121514 2.1719 0 4.5081104  
4.3080011 2.1406 1.27 4.5171412 0 3.797275 0 4.8121514 2.1719 0 4.72120006 3.48310 1.27 4.8121514  
2.1719 0 4.12001915 4.11141911 4.8016315 4.24151114 1.19 0 2.18 0 3.151119 1.27 4.5171412 0  
4.12151160 4.19141411 4.10081918 4.63121511 4.14199303 0 4.8121514 2.1719 0 4.64230418 2.9367  
1.27 4.5171412 0 4.16081810 2.819 0 4.8121514 2.1719 0 4.80200013 4.19201266 4.8170220 2.819 1.27  
4.5171412 0 4.16081810 4.8196321 4.8182000 4.11082500 4.19081413 0 4.8121514 2.1719 0 4.2081702  
4.20081960 4.3170022 2.417 1.27 4.8121514 2.1719 0 4.18020704 4.12031700 1.22 1.27 4.8121514 2.1719  
0 4.18020704 4.12031700 4.22630411 4.4120413 2.1918 0 2.18 0 3.41112 1.27 4.5171412 0 4.18020704  
4.12031700 1.22 0 4.8121514 2.1719 0 4.11140608 1.2 1.27 4.5171412 0 4.8190417 4.19141411 1.18 0  
4.8121514 2.1719 0 4.15171403 3.200219 1.27 4.5171412 0 4.12001915 4.11141911 2.801 0 4.8121514  
2.1719 0 4.15241511 2.1419 0 2.18 0 3.151119 1.27 4.8121514 2.1719 0 4.13201215 1.24 0 2.18 0 2.1315  
1.27 4.8121514 2.1719 0 4.6111401 1.27 4.5171412 0 4.2141111 4.4021908 3.141318 0 4.8121514 2.1719  
0 4.3040500 4.20111903 3.80219 1.27 4.5171412 0 4.12001915 4.11141911 4.8016300 4.13081200  
4.19081413 0 4.8121514 2.1719 0 4.69201302 4.64130812 4.190814 1.13 1.27 4.8121514 2.1719 0  
4.15182019 2.811 1.27 4.8121514 2.1719 0 4.18020704 4.3201104 1.27 4.5171412 0 4.15001907 3.110801  
0 4.8121514 2.1719 0 4.79001907 1.27 4.8121514 2.1719 0 4.18072019 2.811 1.27 4.8121514 2.1719 0  
4.8121514 4.17191108 4.1632019 2.811 1.27 1.39 0 4.8121514 2.1719 0 4.15240020 4.19140620 1.8 1.27  
1.39 0 4.5171412 0 3.797275 0 4.8121514 2.1719 0 4.72120006 2.431 0 4.4231504 4.17081204 2.1319 1.27

1.27 4.5171412 0 3.797275 0 4.8121514 2.1719 0 4.72120006 2.431 0 4.72120006 4.4671700 1.22 1.27  
4.8121514 2.1719 0 4.15111419 4.11246306 4.17001507 4.60140109 4.4021918 0 2.18 0 2.614 1.27  
4.5171412 0 4.79248019 1.95 0 4.8121514 2.1719 0 4.80198608 4.3060419 2.1831 0 4.80196614 2.1704  
1.27 4.5171412 0 4.12001915 4.11141911 4.8016301 4.21004 4.13031863 4.1000210 4.4130360  
4.16199500 2.606 0 4.8121514 2.1719 0 4.69080620 4.17046600 4.13210018 4.80836406 1.6 0 2.18 0  
4.69080620 4.17046600 4.13210018 1.27 4.5171412 0 4.12001915 4.11141911 4.8016305 4.8062017 1.4 0  
4.8121514 2.1719 0 4.69080620 2.1704 1.27 4.5171412 0 4.79248019 4.95638019 4.86080306 3.41918 0  
4.8121514 2.1719 0 4.80641515 4.11080200 4.19081413 1.31 0 4.80860803 4.6041931 0 4.80856514  
4.23750024 4.14201931 0 4.80750813 4.4680308 2.1931 0 4.80750001 3.41131 0 4.80792018 4.7652019  
4.19141331 0 4.80820217 4.14111164 4.17040031 0 4.80830423 4.19680308 1.19 1.27 4.14186304  
4.13210817 4.14134245 4.80836064 4.79724541 0 1.62 0 4.45152416 3.199545 1.27 4.19041215  
4.11001904 1.18 0 1.62 0 1.43 1.27 0 0 0 0 4.50010018 4.8025037 0 4.42500803 2.5031 0 4.50130012  
3.45031 0 4.50210011 4.20045041 1.31 1.27 0 0 0 0 4.50030419 4.81104 3.35037 0 4.42500803 2.5031 0  
4.50130012 3.45031 0 4.50030418 4.2170815 4.19081413 2.5031 0 4.50190812 4.4181900 4.12155041  
1.27 1.44 1.27 1.27 4.15170813 4.19575050 4.50710417 1.4 0 2.818 0 1.0 0 4.11081819 0 2.1405 0  
3.141304 0 4.7201303 3.170403 0 4.3081819 4.8130219 0 4.5080411 2.318 0 4.22081907 2.813 0 3.190704  
0 4.17040011 1.12 0 2.1405 0 4.12001907 4.4120019 4.8020011 0 4.17041804 4.170207 0 3.1303 0  
4.3042104 4.11141512 4.4131937 1.27 1.27 2.9163 0 4.64110604 4.1170008 1.2 0 4.70041412 4.4191724  
1.27 2.9263 0 4.64110604 4.1170008 1.2 0 4.83141514 4.11140624 1.27 2.9363 0 4.64110604 4.1170008  
1.2 0 4.77201201 2.417 0 4.83070414 2.1724 1.27 2.9463 0 4.64110604 4.1170008 1.2 0 4.66141201  
4.8130019 4.14170802 1.18 1.27 2.9563 0 4.64130011 4.24190802 0 4.77201201 2.417 0 4.83070414  
2.1724 1.27 2.9663 0 4.64151511 3.80403 0 4.76001907 4.4120019 3.80218 1.27 2.9763 0 4.64151517  
4.14230812 4.190814 1.13 0 4.83070414 2.1724 1.27 2.9863 0 4.64170819 4.7120419 2.802 0 4.70041412  
4.4191724 1.27 2.9963 0 4.64182412 4.15191419 2.802 0 4.64130011 4.24180818 1.27 3.919063 0  
4.65081412 4.190704 4.12001908 2.218 1.27 3.919163 0 4.65170008 1.3 0 4.83070414 2.1724 1.27  
3.919263 0 4.66001102 4.20112018 0 2.1405 0 4.85001708 4.190814 2.1318 1.27 3.919363 0 4.66001904  
4.6141724 0 4.83070414 2.1724 1.27 3.919463 0 4.66070014 1.18 0 4.83070414 2.1724 1.27 3.919563 0  
4.66140308 2.1306 0 4.83070414 2.1724 1.27 3.919663 0 4.66141201 4.8130019 4.14170800 1.11 0  
4.83141514 4.11140624 1.27 3.919763 0 4.66141201 4.8130019 4.14170802 1.18 1.27 3.919863 0  
4.66141215 3.110423 0 4.64130011 4.24180818 1.27 3.919963 0 4.66141215 4.20190019 4.8141300 1.11 0  
4.64110604 3.11700 1.27 3.929063 0 4.66141215 4.20190019 4.8141300 1.11 0 4.70041412 4.4191724  
1.27 3.929163 0 4.66141215 4.20190019 4.8141300 1.11 0 4.77201201 2.417 0 4.83070414 2.1724 1.27  
3.929263 0 4.66141319 3.171411 0 4.83070414 2.1724 1.27 3.929363 0 4.66172415 4.19140617 4.150724  
1.27 3.929463 0 4.67080505 4.4170413 4.19080011 0 4.64110604 3.11700 1.27 3.929563 0 4.67080505  
4.4170413 4.19080011 0 4.68162000 4.19081413 1.18 1.27 3.929663 0 4.67080505 4.4170413 4.19080011  
0 4.70041412 4.4191724 1.27 3.929763 0 4.67080505 4.4170413 4.19080011 0 4.83141514 4.11140624  
1.27 3.929863 0 4.67081802 4.17041904 0 4.70041412 4.4191724 1.27 3.929963 0 4.67081802 4.17041904  
0 4.76001907 4.4120019 3.80218 1.27 3.939063 0 4.67241300 4.12080200 1.11 0 4.82241819 3.41218 1.27  
3.939163 0 4.68111108 4.15190802 0 4.66201721 2.418 1.27 3.939263 0 4.68132012 4.4170019 3.82104 0  
4.66141201 4.8130019 4.14170802 1.18 1.27 3.939363 0 4.68170614 3.30802 0 4.83070414 2.1724 1.27  
3.939463 0 4.68231504 4.17081204 4.13190011 0 4.76001907 4.4120019 3.80218 1.27 3.939563 0  
4.69081308 2.1904 0 4.70041412 4.4191724 1.27 3.939663 0 4.69112008 1.3 0 4.67241300 4.12080218  
1.27 3.939763 0 4.69142017 3.80417 0 4.64130011 4.24180818 1.27 3.939863 0 4.69170002 3.190011 0  
4.70041412 4.4191724 1.27 3.939963 0 4.69201302 4.19081413 2.11 0 4.64130011 4.24180818 1.27  
3.949063 0 4.69202525 1.24 0 4.76001907 4.4120019 3.80218 1.27 3.949163 0 4.70001204 0 4.83070414  
2.1724 1.27 3.949263 0 4.70041412 4.4191708 1.2 0 4.64130011 4.24180818 1.27 3.949363 0 4.70041412  
4.4191708 1.2 0 4.83141514 4.11140624 1.27 3.949463 0 4.70170015 1.7 0 4.83070414 2.1724 1.27  
3.949563 0 4.70171420 1.15 0 4.83070414 2.1724 1.27 3.949663 0 4.71001712 4.14130802 0 4.64130011  
4.24180818 1.27 3.949763 0 4.71141214 4.11140608 3.20011 0 4.64110604 3.11700 1.27 3.949863 0  
4.71141214 4.19141524 0 4.83070414 2.1724 1.27 3.949963 0 4.71241504 4.17011411 2.802 0 4.70041412

4.4191724 1.27 3.959063 0 4.72130208 4.3041302 1.4 0 4.70041412 4.4191724 1.27 3.959163 0  
4.72130514 4.17120019 3.81413 0 4.83070414 2.1724 1.27 3.959263 0 4.72131904 4.6170011 0  
4.68162000 4.19081413 1.18 1.27 3.959363 0 4.72131904 4.6170001 2.1104 0 4.82241819 3.41218 1.27  
3.959463 0 4.74598307 4.4141724 1.27 3.959563 0 4.74131419 0 4.83070414 2.1724 1.27 3.959663 0  
3.750804 0 4.64110604 4.1170018 1.27 3.959763 0 3.750804 0 4.70171420 2.1518 1.27 3.959863 0  
4.75081304 2.17 0 4.64110604 3.11700 1.27 3.959963 0 4.75081304 2.17 0 4.79171406 4.17001212  
3.81306 1.27 3.969063 0 4.75140608 1.2 1.27 3.969163 0 4.76001308 4.5141103 0 4.83070414 2.1724 1.27  
3.969263 0 4.76001907 4.4120019 4.8020011 0 4.65081411 3.140624 1.27 3.969363 0 4.76001907  
4.4120019 4.8020011 0 4.69081300 3.130204 1.27 3.969463 0 4.76001907 4.4120019 4.8020011 0  
4.75140608 1.2 1.27 3.969563 0 4.76001907 4.4120019 4.8020011 0 4.79072418 3.80218 1.27 3.969663 0  
4.76001917 2.823 0 4.83070414 2.1724 1.27 3.969763 0 4.76040018 3.201704 0 4.83070414 2.1724 1.27  
3.969863 0 4.76140304 1.11 0 4.83070414 2.1724 1.27 3.969963 0 4.77141302 4.14121220 4.19001908  
2.2104 0 4.70041412 4.4191724 1.27 3.979063 0 4.77141311 4.8130400 1.17 0 4.67241300 4.12080218  
1.27 3.979163 0 4.77201201 2.417 0 4.83070414 2.1724 1.27 3.979263 0 4.77201204 4.17080200 1.11 0  
4.64130011 4.24180818 1.27 3.979363 0 4.78150417 4.191417 0 4.64110604 4.1170018 1.27 3.979463 0  
4.78150417 4.191417 0 4.83070414 2.1724 1.27 3.979563 0 4.78151908 4.12082500 4.19081413 1.27  
3.979663 0 4.78170308 4.13001724 0 4.67080505 4.4170413 4.19080011 0 4.68162000 4.19081413 1.18  
1.27 3.979763 0 4.79001719 3.80011 0 4.67080505 4.4170413 4.19080011 0 4.68162000 4.19081413 1.18  
1.27 3.979863 0 4.79041702 4.14110019 3.81413 0 4.83070414 2.1724 1.27 3.979963 0 4.79041719  
4.20170100 4.19081413 0 4.83070414 2.1724 1.27 3.989063 0 4.79171401 4.10811 3.81924 0 4.83070414  
2.1724 1.27 3.989163 0 4.80200013 3.192012 0 4.64110604 3.11700 1.27 3.989263 0 4.80200013 3.192012  
0 4.70171420 2.1518 1.27 3.989363 0 4.80200013 3.192012 0 4.83141514 4.11140624 1.27 3.989463 0  
4.80200420 4.4081306 0 4.83070414 2.1724 1.27 3.989563 0 4.81040011 0 4.64130011 4.24180818 1.27  
3.989663 0 4.81041517 4.4180413 4.19001908 2.1413 0 4.83070414 2.1724 1.27 3.989763 0 4.81080412  
4.131308 2.13 0 4.70041412 4.4191724 1.27 3.989863 0 4.81081306 0 4.83070414 2.1724 1.27 3.989963 0  
3.820419 0 4.83070414 2.1724 1.27 3.999063 0 4.82081306 4.20110017 3.81924 0 4.83070414 2.1724 1.27  
3.999163 0 4.82150402 4.19170011 0 4.83070414 2.1724 1.27 3.999263 0 4.82190019 4.8181908 3.20011  
0 4.76040207 4.130802 1.18 1.27 3.999363 0 4.82191402 4.7001819 2.802 0 4.79171402 4.4181804 1.18  
1.27 3.999463 0 4.82241215 4.11040219 2.802 0 4.70041412 4.4191724 1.27 3.999563 0 4.82241819  
3.41218 0 4.83070414 2.1724 1.27 3.999663 0 4.83041318 2.1417 0 4.64130011 4.24180818 1.27 3.999763  
0 4.83141514 4.11140608 3.20011 0 4.70171420 2.1518 1.27 3.999863 0 4.83141514 4.11140624 1.27  
3.999963 0 4.83171415 4.8020011 0 4.70041412 4.4191724 1.27 4.91909063 0 4.86002104 4.11041918 0  
3.1303 0 4.76201119 4.8170418 4.14112019 3.81413 0 4.64130011 4.24180818 1.27 1.27 4.83070418 1.4 0  
4.5080411 2.318 0 4.17041517 4.4180413 1.19 0 1.0 0 4.21001819 0 4.11001303 4.18020015 1.4 0 2.1405  
0 4.12001907 4.4120019 4.8020011 0 4.8131620 3.81724 0 3.1303 0 4.7002104 0 4.4231904 4.13180821  
1.4 0 4.151511 4.8020019 4.8141318 0 4.21714 2.1818 0 4.21001708 3.142018 0 4.18020804 4.13190805  
2.802 0 3.1303 0 4.4130608 4.13040417 3.81306 0 4.3141200 4.8131863 1.27 1.27 4.71041704 0 2.818 0  
1.0 0 4.3041802 4.17081519 3.81413 0 2.1405 0 4.19220411 2.2104 0 4.1171400 1.3 0 4.5080411 2.318 0  
2.1405 0 4.18192003 1.24 0 4.19070019 0 4.4130214 4.12150018 1.18 0 4.12001324 0 2.1405 0 3.190704 0  
4.18150402 4.8001108 3.250403 0 4.170400 1.18 0 2.1405 0 4.12001907 4.4120019 4.8020011 0  
4.17041804 4.170207 0 3.1303 0 4.3042104 4.11141512 3.41319 0 4.9201819 0 4.11081819 3.40337 1.27  
1.27 2.9163 0 4.64110604 4.1170008 1.2 0 4.82191720 4.2192017 3.41837 0 4.83070818 0 4.5080411 1.3 0  
4.8130211 4.20030418 0 4.18192003 3.80418 0 2.813 0 4.110604 4.1170008 1.2 0 4.6041412 4.4191724  
1.31 0 4.110604 4.1170008 1.2 0 4.19141514 4.11140624 1.31 0 4.110604 4.1170008 1.2 0 4.13201201  
2.417 0 4.19070414 3.172431 0 3.1303 0 4.110604 4.1170008 1.2 0 4.2141201 4.8130019 4.14170802  
2.1863 0 2.7219 0 4.8132114 4.11210418 0 3.190704 0 4.4231511 4.14170019 3.81413 0 2.1405 0  
4.12001907 4.4120019 4.8020011 0 4.18191720 4.2192017 2.418 0 4.19070019 0 3.1704 0 4.5201303  
4.120413 4.19001111 1.24 0 4.110604 4.1170008 1.2 0 2.813 0 4.13001920 3.170431 0 4.8132114  
4.11210813 1.6 0 4.14150417 4.190814 2.1318 0 4.22081907 2.813 0 4.18041918 0 4.19070019 0  
4.5141111 2.1422 0 4.18150402 4.8050802 0 4.230814 3.121831 0 4.11081004 0 4.6171420 3.151831 0

4.17081306 2.1831 0 3.1303 0 4.5080411 3.31863 0 4.83070418 1.4 0 4.18192003 3.80418 0 3.1704 0  
4.2172002 3.80011 0 3.51417 0 4.20130304 4.17181900 4.13030813 1.6 0 4.6041412 4.4191708 3.20011 0  
4.15171415 4.4171908 2.418 0 4.19071714 3.200607 0 4.110604 4.1170008 1.2 0 4.4231517 4.4181808  
4.14131863 1.27 1.27 2.9263 0 4.77201201 2.417 0 4.83070414 3.172437 0 4.66141215 4.17081808 2.1306  
0 4.130011 4.24190802 0 4.13201201 2.417 0 4.19070414 3.172431 0 4.110604 4.1170008 1.2 0  
4.13201201 2.417 0 4.19070414 3.172431 0 3.1303 0 4.170819 4.7120419 2.802 0 4.6041412 4.4191724  
1.31 0 4.19070818 0 4.5080411 1.3 0 4.5140220 3.180418 0 2.1413 0 3.190704 0 4.15171415 4.4171908  
2.418 0 3.1303 0 4.17041100 4.19081413 4.18070815 1.18 0 2.1405 0 4.13201201 4.4171831 0 4.15001719  
4.8022011 4.171124 0 3.190704 0 4.8131904 4.6041718 1.63 0 2.7219 0 4.20190811 4.8250418 0  
4.19040207 4.13081620 2.418 0 4.5171412 0 1.0 0 4.1171400 1.3 0 4.17001306 1.4 0 2.1405 0 4.12001907  
4.4120019 4.8020011 0 4.3081802 4.8151108 3.130418 0 2.1914 0 4.18141121 1.4 0 4.15171401  
4.11041218 0 4.17041100 3.190403 0 2.1914 0 4.3082108 4.18080108 4.11081924 1.31 0 4.2141306  
4.17200413 4.2041831 0 3.1303 0 3.190704 0 4.3081819 4.17080120 4.19081413 0 2.1405 0 4.15170812  
3.41863 1.27 1.27 2.9363 0 4.66141201 4.8130019 4.14170802 1.18 0 3.1303 0 4.70170015 1.7 0  
4.83070414 3.172437 0 4.83070818 0 4.8130211 4.20030418 0 4.4132012 4.4170019 3.82104 0 4.2141201  
4.8130019 4.14170802 2.1831 0 4.110604 4.1170008 1.2 0 4.2141201 4.8130019 4.14170802 2.1831 0  
3.1303 0 4.6170015 1.7 0 4.19070414 3.172463 0 2.7219 0 4.18192003 3.80418 0 4.2141201 4.8130019  
4.14170800 1.11 0 4.18191720 4.2192017 2.418 0 3.1303 0 4.19070408 1.17 0 4.110604 4.1170008 1.2 0  
4.15171415 4.4171908 2.418 0 2.1914 0 4.20130304 4.17181900 2.1303 0 4.2141305 4.8062017 4.190814  
2.1318 0 3.1303 0 4.17041100 4.19081413 1.18 0 4.11081004 0 4.19071418 1.4 0 4.5142013 1.3 0 2.813 0  
4.6170015 1.7 0 4.18191720 4.2192017 3.41831 0 4.3041808 4.6131831 0 3.1303 0 4.2140304 2.1863 1.27  
1.27 2.9463 0 4.83141514 4.11140608 3.20011 0 4.82192003 4.8041837 0 4.83070818 0 4.5080411 1.3 0  
4.2142104 2.1718 0 4.170400 1.18 0 4.18200207 0 2.18 0 4.110604 4.1170008 1.2 0 4.19141514  
4.11140624 1.31 0 4.3080505 4.4170413 4.19080011 0 4.19141514 4.11140624 1.31 0 3.1303 0 4.6041412  
4.4191708 1.2 0 4.19141514 4.11140624 1.63 0 2.7219 0 4.3040011 1.18 0 4.22081907 0 4.15171415  
4.4171908 2.418 0 4.19070019 0 3.1704 0 4.15170418 4.4172104 1.3 0 4.19071714 3.200607 0 4.3040514  
4.17120019 4.8141318 1.31 0 4.19220818 4.19081306 2.1831 0 3.1303 0 4.18191704 4.19020708 3.130618  
0 2.1405 0 4.14010904 4.2191831 0 4.8132104 4.18190806 4.190813 1.6 0 4.2141302 4.4151918 0  
4.11081004 0 4.2141319 4.8132008 3.192431 0 4.2141215 4.21913 4.4181831 0 3.1303 0 4.2141313  
4.4021904 4.3130418 2.1863 1.27 1.27 2.9563 0 4.67080505 4.4170413 4.19080011 0 4.70041412  
4.4191724 0 3.1303 0 4.70041412 4.4191708 1.2 0 4.64130011 4.24180818 1.37 0 4.83070818 0 4.4130214  
4.12150018 3.180418 0 4.3080505 4.4170413 4.19080011 0 4.6041412 4.4191724 1.31 0 4.3080505  
4.4170413 4.19080011 0 4.19141514 4.11140624 1.31 0 3.1303 0 4.6041412 4.4191708 1.2 0 4.130011  
4.24180818 1.63 0 2.7219 0 4.8132114 4.11210418 0 3.190704 0 3.201804 0 2.1405 0 4.2001102  
4.20112018 0 3.1303 0 4.110604 3.11700 0 2.1914 0 4.18192003 1.24 0 4.15171401 4.11041218 0 2.813 0  
4.6041412 4.4191724 1.31 0 4.5140220 4.18081306 0 2.1413 0 4.2201721 3.41831 0 4.18201705 4.20418  
1.31 0 3.1303 0 4.7080607 4.4175903 4.8120413 4.18081413 2.11 0 4.130011 4.14062004 2.1863 1.27 1.27  
2.9663 0 4.643130011 4.24190802 2.11 0 4.82192003 4.8041837 0 4.69080411 2.318 0 4.11081004 0  
4.2141215 3.110423 0 4.130011 4.24180818 1.31 0 4.5201302 4.19081413 2.11 0 4.130011 4.24180818  
1.31 0 3.1303 0 4.7001712 4.14130802 0 4.130011 4.24180818 0 4.5001111 0 4.20130304 1.17 0  
4.19070818 0 4.2001904 4.6141724 1.63 0 4.83070424 0 4.8132114 3.112104 0 3.190704 0 4.3041900  
4.8110403 0 4.8132104 4.18190806 4.190814 1.13 0 2.1405 0 4.5201302 4.19081413 2.1831 0 4.19070408  
1.17 0 4.18150002 3.41831 0 3.1303 0 4.14190704 1.17 0 4.17041100 3.190403 0 4.12001907 4.4120019  
4.8020011 0 4.4131908 4.19080418 1.31 0 4.15171421 4.8030813 1.6 0 1.0 0 4.3040415 0 4.20130304  
4.17181900 4.13030813 1.6 0 2.1405 0 4.19070408 1.17 0 4.1040700 4.21081417 0 3.1303 0 4.15171415  
4.4171908 3.41863 1.27 1.27 2.9763 0 4.64151511 3.80403 0 4.76001907 4.4120019 4.8020011 0  
4.82020804 4.13020418 1.37 0 4.83070818 0 4.8130211 4.20030418 0 4.151511 3.80403 0 4.12001907  
4.4120019 4.8021831 0 4.12001907 4.4120019 4.8020011 0 4.15072418 4.8021831 0 4.12001907  
4.4120019 4.8020011 0 4.5081300 4.13020431 0 3.1303 0 4.1081412 4.190704 4.12001908 3.21831 0  
4.5140220 4.18081306 0 2.1413 0 3.190704 0 4.151511 4.8020019 3.81413 0 2.1405 0 4.12001907

4.4120019 4.8020011 0 4.12041907 3.140318 0 2.124 0 4.3080505 4.4170413 1.19 0 4.5080411 2.318 0  
4.18200207 0 2.18 0 4.18020804 3.130204 0 3.1303 0 4.4130608 4.13040417 4.8130663 0 4.83070818 0  
4.170400 0 4.151511 3.80418 0 4.19070414 4.17080418 0 3.1303 0 4.19040207 4.13081620 2.418 0  
4.5171412 0 3.190704 0 4.15201704 0 4.15001719 1.18 0 2.1405 0 4.12001907 4.4120019 3.80218 0  
2.1914 0 4.15170002 4.19080200 1.11 0 4.15171401 4.11041218 1.63 1.27 1.27 2.9863 0 4.66141215  
4.20190019 4.8141300 1.11 0 4.76001907 4.4120019 4.8021837 0 4.83070818 0 4.5080411 1.3 0  
4.2142104 2.1718 0 4.2141215 4.20190019 4.8141300 1.11 0 4.110604 4.1170031 0 4.2141215 4.20190019  
4.8141300 1.11 0 4.6041412 4.4191724 1.31 0 3.1303 0 4.13201204 4.17080200 1.11 0 4.130011  
4.24180818 1.63 0 2.7219 0 4.3040011 1.18 0 4.22081907 0 4.12001907 4.4120019 4.8020011 0  
4.17041804 4.170207 0 2.813 0 4.170400 1.18 0 4.19070019 0 4.17041620 3.81704 0 4.11001706  
4.4591802 3.1104 0 4.2141215 4.20190019 3.81413 0 3.1303 0 4.110614 4.17081907 3.120802 0  
4.15170402 4.8180814 2.1331 0 4.14051904 1.13 0 3.51417 0 4.18081220 4.11001908 4.14131831 0  
4.14151908 4.12082500 4.19081413 2.1831 0 3.1303 0 4.2141215 3.110423 0 4.2001102 4.20110019  
4.8141318 1.63 1.27 1.27 2.9963 0 4.67081802 4.17041904 0 4.76001907 4.4120019 4.8021837 0  
4.72130211 4.20030813 1.6 0 4.3081802 4.17041904 0 4.6041412 4.4191724 1.31 0 4.2141201 4.8130019  
4.14170800 1.11 0 4.19141514 4.11140624 1.31 0 3.1303 0 4.3081802 4.17041904 0 4.12001907  
4.4120019 3.80218 0 4.8191804 3.110531 0 4.19070818 0 4.5080411 1.3 0 4.8132104 4.18190806  
4.190418 0 4.12001907 4.4120019 4.8020011 0 4.18191720 4.2192017 2.418 0 4.19070019 0 3.1704 0  
4.5201303 4.120413 4.19001111 1.24 0 4.3081802 4.17041904 0 4.17001907 2.417 0 4.19070013 0  
4.2141319 4.8132014 3.201863 0 2.7219 0 3.70018 0 4.151511 4.8020019 4.8141318 0 2.813 0 4.2141215  
4.20190417 0 4.18020804 4.13020431 0 4.2172415 4.19140617 4.150724 1.31 0 3.1303 0 4.8130514  
4.17120019 3.81413 0 4.19070414 3.172463 1.27 1.27 3.919063 0 4.67241300 4.12080200 1.11 0  
4.82241819 3.41218 0 3.1303 0 4.68170614 3.30802 0 4.83070414 3.172437 0 4.82192003 4.24081306 0  
4.170400 1.18 0 4.11081004 0 4.3241300 4.12080200 1.11 0 4.18241819 4.4121831 0 4.2070014 1.18 0  
4.19070414 3.172431 0 3.1303 0 4.4170614 3.30802 0 4.19070414 3.172431 0 4.19070818 0 4.5080411 1.3  
0 4.4231511 4.14170418 0 4.18241819 3.41218 0 4.19070019 0 4.4211411 2.2104 0 4.14210417 0  
4.19081204 0 4.20214 4.17030813 1.6 0 2.1914 0 4.18150402 4.8050802 0 4.17201104 2.1863 0 2.7219 0  
4.4230012 4.8130418 0 3.71422 0 4.19070418 1.4 0 4.18241819 3.41218 0 4.1040700 3.210431 0  
4.4211411 3.210431 0 3.1303 0 4.17041815 3.141303 0 2.1914 0 4.21001708 3.142018 0 4.8131520 2.1918  
0 4.14210417 0 3.190704 0 4.11141306 0 4.19041712 1.63 1.27 1.27 3.919163 0 4.76001907 4.4120019  
4.8020011 0 4.75140608 1.2 0 3.1303 0 4.69142013 4.3001908 4.14131837 0 4.66142104 4.17081306 0  
4.11140608 2.231 0 3.180419 0 4.19070414 3.172431 0 3.1303 0 4.12140304 1.11 0 4.19070414 3.172431  
0 4.19070818 0 4.5080411 1.3 0 4.18192003 3.80418 0 3.190704 0 4.5141712 2.11 0 4.1001808 1.18 0  
2.1405 0 4.12001907 4.4120019 4.8021863 0 2.7219 0 4.8132104 4.18190806 4.190418 0 3.190704 0  
4.15170813 4.2081511 2.418 0 2.1405 0 4.12001907 4.4120019 4.8020011 0 4.17040018 4.14130813 2.631  
0 3.190704 0 4.13001920 2.1704 0 2.1405 0 4.12001907 4.4120019 4.8020011 0 4.14010904 4.2191831 0  
3.1303 0 3.190704 0 4.19070414 4.17041908 3.20011 0 4.20130304 4.17150813 4.13081306 1.18 0 2.1405  
0 4.12001907 4.4120019 4.8020011 0 4.19070414 4.17080418 1.63 1.27 1.27 3.919263 0 4.78151908  
4.12082500 4.19081413 0 3.1303 0 4.66141319 3.171411 0 4.83070414 3.172437 0 4.68130214  
4.12150018 4.18081306 0 4.11081304 2.17 0 4.05171406 4.17001212 4.8130631 0 4.14151908  
4.12082500 4.19081413 1.31 0 4.2141319 3.171411 0 4.19070414 3.172431 0 3.1303 0 4.18241819  
3.41218 0 4.19070414 3.172431 0 4.19070818 0 4.5080411 1.3 0 2.818 0 4.2141302 4.4171304 1.3 0  
4.22081907 0 4.5081303 3.81306 0 3.190704 0 4.1041819 0 4.15141818 4.8011104 0 4.18141120  
4.19081413 0 2.1914 0 1.0 0 4.15171401 4.11041231 0 4.6082104 1.13 0 4.2141318 4.19170008 3.131918  
0 3.1303 0 4.14010904 4.2190821 3.41831 0 3.1303 0 4.2141319 4.17141111 3.81306 0 3.190704 0  
4.1040700 4.21081417 0 2.1405 0 4.3241300 3.120802 0 4.18241819 3.41218 0 2.813 0 2.13 0 4.14151908  
3.120011 0 4.12001313 3.41763 1.27 1.27 4.68000207 0 2.1405 0 4.19070418 1.4 0 4.5080411 2.318 0  
2.818 0 4.21001819 0 3.1303 0 4.8131904 4.17021413 4.13040219 1.18 0 4.22081907 0 4.12201119  
4.8151104 0 4.170400 1.18 0 2.1405 0 4.12001907 4.4120019 4.8021831 0 4.8111120 4.18191700  
4.19081306 0 3.190704 0 4.15171405 4.14201303 0 4.3041519 1.7 0 3.1303 0 4.1170400 3.31907 0 2.1405

0 4.12001907 4.4120019 4.8020011 0 4.17041804 4.170207 0 3.1303 0 4.3042104 4.11141512 4.4131963  
1.27 1.27 4.68132012 4.4170019 3.82104 0 4.76001907 4.4120019 4.8020011 0 4.69170012 4.4221417  
1.10 0 2.1405 0 4.83071420 4.6071937 1.27 1.27 1.91 0 1.42 0 4.67040508 4.13081908 3.1413180  
3.641303 0 4.78150417 4.191417 1.18 0 1.41 1.27 1.27 1.91 0 4.79171414 1.5 0 1.62 0 3.830704 0 4.11804  
3.130204 0 2.1405 0 4.3142001 2.1963 1.27 1.27 1.92 0 4.82141120 4.19081413 0 1.62 0 2.8314 0  
4.18141121 1.4 0 1.0 0 4.15171401 3.110412 0 4.22081907 3.142019 0 3.190704 0 4.2002018 1.4 0 2.1405  
0 4.131419 4.7041763 1.27 1.27 1.93 0 4.79171401 3.110412 0 1.62 0 3.830704 0 4.15170418 4.4130204 0  
2.1405 0 4.3142001 2.1963 1.27 1.27 1.94 0 4.66141319 3.42319 0 1.62 0 4.78131124 0 3.141304 0  
4.2141319 3.42319 0 2.818 0 4.4210417 0 4.15170418 4.4131963 0 4.83070019 0 3.81831 0 4.14130204 0  
4.19172004 1.31 0 4.112200 2.2418 0 4.19172004 1.63 1.27 1.27 1.95 0 4.77201201 2.417 0 1.62 0 1.64 0  
4.15072418 4.8020011 0 4.18191720 4.2192017 2.431 0 4.15170418 3.41319 0 3.32004 0 2.1914 0 1.0 0  
4.18190019 1.4 0 4.19170013 4.18081908 3.141363 1.27 1.27 1.96 0 4.84130819 0 4.19170013 4.18081908  
2.1413 0 1.62 0 3.830704 0 4.11040018 1.19 0 4.4050504 4.2190821 1.4 0 4.2070013 2.604 0 2.1405 0  
4.18190019 2.463 1.27 1.27 1.97 0 4.66141215 4.14201303 0 4.19170013 4.18081908 2.1413 0 1.62 0 1.64  
0 4.18041620 4.4130204 0 2.1405 0 4.20130819 0 4.19170013 4.18081908 4.14131863 1.27 1.27 1.98 0  
3.820419 0 1.62 0 2.6413 0 4.171700 1.24 0 2.1405 0 4.13201201 4.4171863 1.27 1.27 1.99 0 3.760015 0  
1.62 0 3.781304 0 4.77201201 2.417 0 4.17041820 3.111918 0 2.813 0 4.131419 3.70417 0 4.77201201  
3.41731 0 2.124 0 1.0 0 4.3040508 3.130403 0 4.84130819 0 2.1417 0 4.66141215 4.14201303 0  
4.19170013 4.18081908 3.141363 1.27 1.27 2.9190 0 4.66110008 1.12 0 1.62 0 1.64 0 4.12001563 1.27  
1.27 2.9191 0 4.66141215 4.11041908 2.1413 0 1.62 0 4.79171414 1.5 0 2.1405 0 4.66110008 2.1263 1.27  
1.27 2.9192 0 3.641303 1.27 1.27 2.9193 0 3.871417 0 4.57682302 4.11201808 4.21041124 0 3.781758  
1.27 1.27 2.9194 0 3.771419 1.27 1.27 1.92 0 1.42 0 4.82190019 4.4120413 1.19 0 1.41 1.27 1.27 1.91 0  
3.916396 0 1.61 0 3.916396 0 1.62 0 3.916397 1.27 1.27 1.92 0 2.124 0 4.3040508 4.13081908 2.1413 1.27  
1.27 1.93 0 1.92 0 2.1405 0 3.916396 0 3.1303 0 3.916397 0 1.91 0 3.70018 0 4.20708 4.4210403 0  
4.91639191 1.27 1.27 1.93 0 1.42 0 4.82190019 4.4120413 1.19 0 1.41 1.27 1.27 1.91 0 2.5700 0 1.59 0  
2.158 0 2.818 0 4.15170812 1.4 1.27 1.27 1.92 0 1.8 0 1.62 0 1.0 1.27 1.27 1.93 0 1.8 0 1.62 0 1.1 1.27 1.27  
1.94 0 4.82201212 4.190814 1.13 0 2.1405 1.27 1.27 1.95 0 2.5708 0 1.61 0 2.9158 1.27 1.27 1.96 0 1.94 0  
1.95 0 4.5171412 0 1.92 0 2.1914 0 1.93 1.27 1.27 1.97 0 3.830704 0 4.18162000 2.1704 0 4.17141419 0  
2.1405 1.27 1.27 1.98 0 4.96325700 3.329258 1.27 1.27 1.99 0 4.3040208 3.120011 0 4.15001719 0 2.1405  
0 1.98 1.27 1.27 2.9190 0 4.68230211 4.20180821 3.41124 0 2.7881 1.27 1.27 2.9191 0 1.91 0 1.59 0  
4.57030402 4.8120011 0 4.15001719 0 2.1405 0 2.9858 1.27 1.27 2.9192 0 1.99 0 2.9190 0 2.9191 1.27  
1.27 2.9193 0 1.98 0 4.15112018 0 2.1417 0 4.12081320 1.18 0 3.919231 0 2.818 0 4.15170812 1.4 1.27  
1.27 2.9194 0 4.25041714 0 2.818 0 4.11041818 0 4.19070013 0 2.9190 0 4.4162000 1.11 0 2.1914 0  
2.9192 0 4.22070802 1.7 0 2.818 0 4.11041818 0 4.19070013 0 2.9190 0 4.4162000 1.11 0 2.1914 0  
3.141304 1.27 1.27 2.9195 0 4.57599158 0 2.818 0 3.190704 0 4.18190415 0 4.21001120 1.4 0 2.1405 0  
1.96 1.27 1.27 2.9196 0 2.5700 0 1.34 0 2.158 1.27 1.27 2.9197 0 4.25041714 0 2.818 0 4.11041818 0  
4.19070013 0 2.5700 0 1.59 0 2.158 0 4.22070802 1.7 0 2.818 0 4.11041818 0 4.19070013 0 2.9190 0  
4.4162000 1.11 0 2.1914 0 2.9192 1.27 1.27 2.9198 0 1.91 0 1.96 0 1.98 0 2.9192 0 2.9197 0 2.9193 0  
2.9195 0 2.9196 1.27 1.27 2.9199 0 4.68230402 3.201904 0 2.9198 0 2.18 0 3.190704 0 3.916396 0 2.9190  
0 3.916397 1.27 1.27 2.9290 0 2.9199 0 4.14201915 3.201918 0 4.57830704 0 4.13201201 2.417 0 2.9231 0  
1.93 0 3.871417 0 4.95585050 2.5058 1.27 1.27 4.39190418 1.19 1.27 1.39 0 4.67040508 2.1304 0  
4.5201302 4.19081413 1.18 0 3.51417 0 4.21001708 3.142018 0 4.14150417 4.190814 2.1318 1.27 3.30405  
0 4.11081819 4.60050811 4.4185703 4.8170402 4.19141724 4.62456345 2.5837 1.27 0 0 0 0 4.17041920  
2.1713 0 4.14186311 4.8181903 4.8175703 4.8170402 4.19141724 1.58 1.27 1.27 3.30405 0 4.6041960  
4.18241819 4.4126008 4.13051457 2.5837 1.27 0 0 0 0 4.17041920 2.1713 0 1.43 1.27 0 0 0 0 0 0 0  
4.45021520 4.60150417 4.2041319 2.4537 0 4.15182019 4.8116302 4.15206015 4.4170204 4.13195708  
4.13190417 4.21001162 3.915831 1.27 0 0 0 0 0 0 0 4.45210817 4.19200011 4.60120412 4.14172445  
1.37 0 4.15182019 4.8116321 4.8171920 4.116012 4.4121417 4.24575863 4.60001803 4.8021957 2.5831  
1.27 0 0 0 0 0 0 0 0 4.45030818 4.10602018 4.60445 1.37 0 4.15182019 4.8116303 4.8181060 4.20180006  
4.4574532 4.45586360 4.180308 4.2195758 1.27 0 0 0 0 1.44 1.27 1.27 3.30405 0 4.3142213 4.11140003

4.60050811 4.4572017 2.1131 0 4.18002104 4.60150019 3.75837 1.27 0 0 0 0 4.17041815 4.14131804 0  
1.62 0 4.17041620 4.4181918 4.63060419 4.57201711 1.58 1.27 0 0 0 0 4.22081907 0 4.14150413  
4.57180021 4.4601500 3.190731 0 4.45220145 1.58 0 2.18 0 2.537 1.27 0 0 0 0 0 0 0 4.5632217  
4.8190457 4.17041815 4.14131804 4.63021413 4.19041319 1.58 1.27 0 0 0 0 4.17041920 2.1713 0  
4.5506714 4.22131114 4.30403 0 4.43201711 1.44 0 2.1914 0 4.43180021 4.4601500 4.19074450 1.27 1.27  
3.30405 0 4.12142104 4.60121420 4.18045723 1.31 0 2.2431 0 4.3201700 4.19081413 4.62915837 1.27 0 0  
0 0 4.19172437 1.27 0 0 0 0 0 0 0 1.23 0 1.62 0 4.8131957 2.2358 1.27 0 0 0 0 0 0 0 1.24 0 1.62 0  
4.8131957 2.2458 1.27 0 0 0 0 0 0 0 4.3201700 4.19081413 0 1.62 0 4.5111400 4.19570320 4.17001908  
3.141358 1.27 0 0 0 0 0 0 0 0 4.15240020 4.19140620 4.8631214 4.21048314 3.572331 0 2.2431 0  
4.3201700 4.19081413 4.62032017 4.190814 2.1358 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.5507614  
3.210403 0 4.12142018 1.4 0 2.1914 0 4.57432344 1.31 0 4.43244458 0 2.813 0 4.43032017 4.190814  
2.1344 0 4.18040214 4.13031850 1.27 0 0 0 0 4.4230204 2.1519 0 4.85001120 4.4681717 3.141737 1.27 0  
0 0 0 0 0 0 4.17041920 2.1713 0 4.50681717 3.141737 0 1.23 0 3.1303 0 1.24 0 4.12201819 0 2.104 0  
4.8131904 4.6041718 1.31 0 3.1303 0 4.3201700 4.19081413 0 4.12201819 0 2.104 0 1.0 0 0 4.5111400 1.19  
0 2.1417 0 2.13 0 4.8131904 4.6041763 1.50 1.27 0 0 0 0 4.4230204 2.1519 0 4.68230204 4.15190814 1.13  
0 2.18 0 2.437 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.5506817 3.171417 0 4.4230402 4.20190813 1.6 0  
4.12142104 4.60121420 3.180437 0 4.43044450 1.27 1.27 3.30405 0 4.19001004 4.60180217 4.4041318  
4.7141957 4.18002104 4.60150019 3.75837 1.27 0 0 0 0 4.18021704 4.4131807 2.1419 0 1.62 0  
4.15240020 4.19140620 4.8631802 4.17040413 4.18071419 2.5758 1.27 0 0 0 0 4.18021704 4.4131807  
4.14196318 4.210457 4.18002104 4.60150019 2.758 1.27 0 0 0 0 4.17041920 2.1713 0 4.5508202  
4.17040413 4.18071419 0 4.18002104 1.3 0 2.1914 0 4.43180021 4.4601500 4.19074450 1.27 1.27 1.39 0  
4.67040508 2.1304 0 1.0 0 0 4.12001515 3.81306 0 2.1405 0 4.2141212 3.1303 0 4.13001204 1.18 0 2.1914 0  
4.5201302 4.19081413 1.18 1.27 4.66787676 4.64776782 0 1.62 0 1.43 1.27 0 0 0 0 4.45110818  
4.19600508 4.11041845 1.37 0 4.11081819 4.60050811 3.41831 1.27 0 0 0 0 4.45060419 4.60182418  
4.19041260 4.8130514 2.4537 0 4.6041960 4.18241819 4.4126008 4.13051431 1.27 0 0 0 0 4.45031422  
4.13111400 4.3600508 4.11044537 0 4.3142213 4.11140003 4.60050811 2.431 1.27 0 0 0 0 4.45121421  
4.4601214 4.20180445 1.37 0 4.12142104 4.60121420 3.180431 1.27 0 0 0 0 4.45190010 4.4601802  
4.17040413 4.18071419 2.4537 0 4.19001004 4.60180217 4.4041318 4.7141931 1.27 0 0 0 0 1.39 0  
3.640303 0 4.12141704 0 4.2141212 4.130318 0 2.18 0 4.13040403 2.403 1.27 1.44 1.27 1.27 1.39 0  
4.66170400 2.1904 0 1.0 0 0 4.6111401 2.11 0 4.3080219 4.8141300 2.1724 0 2.1914 0 4.7141103 0  
3.190704 0 4.20180417 4.59030405 4.8130403 0 4.21001708 4.11104 1.18 0 3.1303 0 4.5201302  
4.19081413 1.18 1.27 4.6111401 4.116002 4.14131904 2.2319 0 1.62 0 2.4344 1.27 1.27 1.39 0 4.69201302  
4.19081413 0 2.1914 0 4.4230402 3.201904 0 4.15170403 4.4050813 2.403 0 4.2141212 4.130318 1.27  
3.30405 0 4.4230402 4.20190460 4.20180417 4.60021412 4.12001303 4.57021412 4.12001303  
4.60130012 2.431 0 4.56001706 3.185837 1.27 0 0 0 0 2.805 0 4.2141212 4.130360 4.13001204 0 2.813 0  
4.66787676 4.64776782 1.37 1.27 0 0 0 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 0 0 0 4.17041820 2.1119  
0 1.62 0 4.66787676 4.64776782 4.42021412 4.12001303 4.60130012 4.4415756 4.170618 1.58 1.27 0 0 0  
0 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.17041820 2.1119 1.27 0 0 0 0 0 0 0 4.4230204 2.1519 0  
4.68230204 4.15190814 1.13 0 2.18 0 2.437 1.27 0 0 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.5506817  
3.171417 0 4.4230402 4.20190813 1.6 0 4.2141212 4.130337 0 4.43044450 1.27 0 0 0 0 4.4111804 1.37  
1.27 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.5508413 4.10131422 1.13 0 4.2141212 4.130337 0 4.43021412  
4.12001303 4.60130012 3.44450 1.27 1.27 1.39 0 4.69201302 4.19081413 0 2.1914 0 4.4230402 3.201904  
0 4.3241300 3.120802 0 4.2140304 1.27 3.30405 0 4.4230402 4.20190460 4.3241300 4.12080260  
4.2140304 4.57201804 4.17600214 4.3045837 1.27 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 1.39 0  
4.68230402 3.201904 0 3.190704 0 4.2140304 0 4.22081907 2.813 0 3.190704 0 4.6111401 2.11 0  
4.2141319 3.42319 1.27 0 0 0 0 0 0 4.4230402 4.57201804 4.17600214 3.30431 0 4.6111401 4.116002  
4.14131904 3.231958 1.27 0 0 0 0 4.4230204 2.1519 0 4.68230204 4.15190814 1.13 0 2.18 0 2.437 1.27 0 0  
0 0 0 0 0 4.15170813 4.19570550 4.68171714 1.17 0 4.4230402 4.20190813 1.6 0 4.2140304 1.37 0  
4.43044450 1.58 1.27 1.27 3.30405 0 4.4210011 4.60032413 4.120802 4.60021403 4.4572018 4.4176002  
4.14030458 1.37 1.27 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 1.39 0 4.68210011 4.20001904 0 3.190704 0

4.2140304 0 4.22081907 2.813 0 3.190704 0 4.6111401 2.11 0 4.2141319 3.42319 1.27 0 0 0 0 0 0 0  
4.17041820 2.1119 0 1.62 0 4.4210011 4.57201804 4.17600214 3.30431 0 4.6111401 4.116002 4.14131904  
3.231958 1.27 0 0 0 0 0 0 0 0 4.15170813 4.19570550 4.81041820 3.111937 0 4.43170418 4.20111944  
2.5058 1.27 0 0 0 0 4.4230204 2.1519 0 4.68230204 4.15190814 1.13 0 2.18 0 2.437 1.27 0 0 0 0 0 0 0  
4.15170813 4.19570550 4.68171714 1.17 0 4.4210011 4.20001908 2.1306 0 4.4231517 4.4181808  
3.141337 0 4.43044450 1.58 1.27 1.27 1.39 0 4.69201302 4.19081413 0 2.1914 0 4.7001303 2.1104 0  
4.12201119 4.8110813 1.4 0 4.8131520 1.19 1.27 3.30405 0 4.12201119 4.8110813 4.4600813 4.15201957  
2.5837 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 4.24142017 0 4.79241907 2.1413 0 4.2140304 0  
4.57192415 1.4 0 4.45041303 1.45 0 2.1413 0 1.0 0 3.130422 0 4.11081304 0 2.1914 0 4.5081308  
4.18075837 2.5058 1.27 0 0 0 0 4.11081304 1.18 0 1.62 0 2.4241 1.27 0 0 0 0 4.22070811 1.4 0 4.83172004  
1.37 1.27 0 0 0 0 0 0 0 0 4.11081304 0 1.62 0 4.8131520 3.195758 1.27 0 0 0 0 0 0 0 0 2.805 0 4.11081304  
4.63181917 4.8155758 4.63111422 4.4175758 0 2.6262 0 4.45041303 2.4537 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
4.1170400 1.10 1.27 0 0 0 0 0 0 0 4.11081304 4.18630015 4.15041303 4.57110813 2.458 1.27 0 0 0 0  
4.17041920 2.1713 0 4.45291345 4.63091408 4.13571108 4.13041858 1.27 1.27 1.39 0 4.72131904  
4.17000219 3.82104 0 3.667572 1.27 3.30405 0 4.8131904 4.17000219 4.8210460 4.2110857 2.5837 1.27 0  
0 0 0 4.15170813 4.19575086 4.4110214 2.1204 0 2.1914 0 3.190704 0 4.79241907 2.1413 0 3.667572 0  
3.51417 0 4.86081303 3.142218 0 4.66141319 4.17141150 1.58 1.27 0 0 0 0 4.15170813 4.19575083  
3.241504 0 4.45070411 2.1545 0 2.1914 0 3.180404 0 4.210008 4.11000111 1.4 0 4.2141212 4.130318 1.31  
0 4.45042308 2.1945 0 2.1914 0 4.16200819 1.31 0 2.1417 0 4.15170405 2.823 0 4.22081907 0 3.454945 0  
2.1914 0 4.4230402 3.201904 0 4.79241907 2.1413 0 4.2140304 3.635058 1.27 0 0 0 0 4.15170813  
4.19575083 3.241504 0 4.45122011 3.190845 0 2.1914 0 4.4131904 1.17 0 4.12201119 4.8110813 1.4 0  
4.79241907 2.1413 0 4.2140304 3.635058 1.27 0 0 0 0 1.27 0 0 0 0 4.22070811 1.4 0 4.83172004 1.37 1.27  
0 0 0 0 0 0 0 0 0 4.20180417 4.60081315 2.2019 0 1.62 0 4.8131520 4.19575034 0 2.5058 1.27 0 0 0 0 0 0 0  
1.27 0 0 0 0 0 0 0 0 2.805 0 4.20180417 4.60081315 2.2019 0 2.813 0 4.42450423 4.8194531 0 4.45162008  
4.19454137 1.27 0 0 0 0 0 0 0 0 0 0 4.1170400 1.10 1.27 0 0 0 0 0 0 0 0 4.4110805 0 4.20180417  
4.60081315 2.2019 0 2.6262 0 4.45070411 3.154537 1.27 0 0 0 0 0 0 0 0 0 0 4.15170813 4.19575064  
4.21000811 4.11104 0 4.2141212 4.130318 3.375058 1.27 0 0 0 0 0 0 0 0 0 0 3.51417 0 3.21203 0 2.813  
0 4.66787676 4.64776782 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.15170813 4.19570550 0 0 4.43021203  
3.445058 1.27 0 0 0 0 0 0 0 0 0 0 4.15170813 4.19575088 2.1420 0 3.20013 0 4.111814 0 4.4131904  
1.17 0 4.79241907 2.1413 0 4.2140304 0 4.15170405 4.8230403 0 4.22081907 0 3.454945 0 2.1914 0  
4.4230402 4.20190463 2.5058 1.27 0 0 0 0 0 0 0 0 0 4.15170813 4.19575083 3.241504 0 4.45122011  
3.190845 0 2.1914 0 4.4131904 1.17 0 4.12201119 4.8110813 1.4 0 4.79241907 2.1413 0 4.2140304  
3.635058 1.27 0 0 0 0 0 0 0 0 4.4110805 0 4.20180417 4.60081315 2.2019 0 2.6262 0 4.45122011  
4.19084537 1.27 0 0 0 0 0 0 0 0 0 0 4.2140304 0 1.62 0 4.12201119 4.8110813 4.4600813 4.15201957  
1.58 1.27 0 0 0 0 0 0 0 0 0 0 4.4230402 4.20190460 4.3241300 4.12080260 4.2140304 4.57021403  
2.458 1.27 0 0 0 0 0 0 0 0 4.4110805 0 4.20180417 4.60081315 4.20196318 4.19001719 4.18220819  
4.7574549 3.455837 1.27 0 0 0 0 0 0 0 0 0 0 2.805 0 3.506250 2.813 0 4.20180417 4.60081315 2.2019  
0 2.1417 0 4.45081215 4.14171945 0 2.813 0 4.20180417 4.60081315 3.201937 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 1.39 0 4.71001303 2.1104 0 4.181808 4.6131204 2.1319 0 3.1303 0 4.4230402 4.20190814 1.13 0  
4.18190019 4.4120413 2.1918 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.4230402 4.20190460 4.3241300  
4.12080260 4.2140304 4.57201804 4.17600813 4.15201942 4.91374158 1.27 0 0 0 0 0 0 0 0 0 0  
4.4111804 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 1.39 0 4.71001303 2.1104 0 4.4210011 4.20001908  
2.1413 0 4.18190019 4.4120413 2.1918 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.4210011 4.60032413 4.120802  
4.60021403 4.4572018 4.4176008 4.13152019 4.42913741 1.58 1.27 0 0 0 0 0 0 0 4.4111804 1.37 1.27  
0 0 0 0 0 0 0 0 0 1.39 0 4.68230402 3.201904 0 4.15170403 4.4050813 2.403 0 4.2141212 4.130318  
1.27 0 0 0 0 0 0 0 0 0 0 4.2141212 4.130360 4.15001719 1.18 0 1.62 0 4.20180417 4.60081315  
4.20196318 4.15110819 2.5758 1.27 0 0 0 0 0 0 0 0 0 0 4.2141212 4.130360 4.13001204 0 1.62 0  
4.2141212 4.130360 4.15001719 4.18429041 1.27 0 0 0 0 0 0 0 0 0 0 4.2141212 4.130360 4.170618 0  
1.62 0 4.2141212 4.130360 4.15001719 4.18429137 1.41 1.27 0 0 0 0 0 0 0 0 0 0 4.14201915 2.2019 0  
1.62 0 4.4230402 4.20190460 4.20180417 4.60021412 4.12001303 4.57021412 4.12001303 4.60130012

2.431 0 4.56021412 4.12001303 4.60001706 2.1858 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.15170813 4.19571420  
4.19152019 1.58 1.27 4.39190418 1.19 1.27 1.27 4.2110018 1.18 0 4.83201708 4.13066614 4.12151104  
4.19046420 4.19141200 4.19141392 2.9337 1.27 0 0 0 3.30405 0 4.60600813 4.8196060 4.57180411  
3.55837 1.27 0 0 0 0 0 0 0 4.18041105 4.63190015 1.4 0 1.62 0 2.4344 1.27 0 0 0 0 0 0 0 4.18041105  
4.63070400 4.3601514 4.18081908 2.1413 0 1.62 0 1.90 1.27 1.27 0 0 0 3.30405 0 4.22170819 4.4601900  
4.15049293 4.57180411 2.531 0 4.21001120 3.45837 1.27 0 0 0 0 0 0 0 4.18041105 4.63190015  
4.4421804 4.11056307 4.4000360 4.15141808 4.19081413 1.41 0 1.62 0 4.21001120 1.4 1.27 0 0 0 0 0 0 0  
0 1.39 0 4.70041304 4.17001904 0 3.190704 0 4.15140813 1.19 0 4.1040514 2.1704 0 4.12142108 2.1306 0  
3.190704 0 4.7040003 1.27 0 0 0 0 0 0 0 4.18041105 4.63060413 4.4170019 4.4601514 4.8131992  
3.935758 1.27 0 0 0 0 0 0 0 4.15170813 4.19570550 4.86170819 3.190413 0 4.43210011 3.200444 0 2.19  
0 4.15141808 4.19081413 0 4.43180411 4.5630704 4.36015 4.14180819 4.8141344 2.5058 1.27 0 0 0 0 0 0  
0 0 4.18041105 4.63121421 4.4600704 4.39293 4.57451708 4.6071945 1.31 0 2.9158 0 0 1.39 0  
4.76142104 0 4.7040003 0 4.17080607 1.19 0 4.51904 1.17 0 4.22170819 3.81306 0 3.1303 0 4.6041304  
4.17001908 2.1306 0 4.15140813 1.19 1.27 1.27 0 0 0 0 3.30405 0 4.12142104 4.60070400 4.3929357  
4.18041105 1.31 0 4.3081704 4.2190814 2.1331 0 4.18190415 4.18629158 1.37 1.27 0 0 0 0 0 0 0 0  
4.14110360 4.15141808 4.19081413 0 1.62 0 4.18041105 4.63070400 4.3601514 4.18081908 2.1413 1.27 0  
0 0 0 0 0 0 2.805 0 4.3081704 4.2190814 1.13 0 2.6262 0 4.45110405 3.194537 1.27 0 0 0 0 0 0 0 0 0 0  
4.18041105 4.63070400 4.3601514 4.18081908 2.1413 0 2.5962 0 4.18190415 1.18 1.27 0 0 0 0 0 0 0 0  
4.4110805 0 4.3081704 4.2190814 1.13 0 2.6262 0 4.45170806 4.7194537 1.27 0 0 0 0 0 0 0 0 0 0  
4.18041105 4.63070400 4.3601514 4.18081908 2.1413 0 2.6162 0 4.18190415 1.18 1.27 0 0 0 0 0 0 0  
4.15170813 4.19570550 4.76142104 1.3 0 4.5171412 0 4.43141103 4.60151418 4.8190814 2.1344 0 2.1914  
0 4.43180411 4.5630704 4.36015 4.14180819 4.8141344 2.5058 1.27 1.27 0 0 0 0 3.30405 0 4.6041304  
4.17001904 4.60151408 4.13199293 4.57180411 3.55837 1.27 0 0 0 0 0 0 0 1.23 0 1.62 0 4.18041105  
4.63070400 4.3601514 4.18081908 2.1413 1.27 0 0 0 0 0 0 0 1.24 0 1.62 0 4.18041105 4.63190015  
4.4630604 4.19571804 4.11056307 4.4000360 4.15141808 4.19081413 1.31 0 2.9058 1.27 0 0 0 0 0 0 0  
4.15170813 4.19570550 4.70041304 4.17001904 1.3 0 4.15140813 1.19 0 4.57432344 1.31 0 4.43244458  
2.5058 1.27 1.27 3.30405 0 4.17040003 4.60050811 4.4929357 4.5081104 4.13001204 2.5837 1.27 0 0 0 0  
4.22081907 0 4.14150413 4.57050811 4.4130012 2.431 0 4.45174531 0 4.4130214 4.3081306 4.62452019  
4.5599845 1.58 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.5081104 4.63170400  
3.35758 1.27 1.27 3.30405 0 4.15171402 4.4181860 4.5081104 4.60021413 4.19041319 4.92935702  
4.14131904 3.131931 0 4.19020058 1.37 1.27 0 0 0 0 4.2070017 4.21904 4.17601200 1.15 0 1.62 0 1.43  
1.27 0 0 0 0 0 0 0 4.45004537 0 2.9031 0 4.45014537 0 2.9131 0 4.45024537 0 2.9231 0 4.45034537 0  
2.9331 0 4.45044537 0 2.9431 0 4.45054537 0 2.9531 0 4.45064537 0 2.9631 0 4.45074537 0 2.9731 0  
4.45084537 0 2.9831 0 4.45094537 0 2.9931 0 4.45104537 0 3.919031 0 4.45114537 0 3.919131 0  
4.45124537 0 3.919231 0 4.45134537 0 3.919331 1.27 0 0 0 0 0 0 0 4.45144537 0 3.919431 0 4.45154537  
0 3.919531 0 4.45164537 0 3.919631 0 4.45174537 0 3.919731 0 4.45184537 0 3.919831 0 4.45194537 0  
3.919931 0 4.45204537 0 3.929031 0 4.45214537 0 3.929131 0 4.45224537 0 3.929231 0 4.45234537 0  
3.929331 0 4.45244537 0 3.929431 0 4.45254537 0 3.929531 1.27 0 0 0 0 0 0 0 1.45 0 2.4537 0 3.929631  
0 4.45291345 1.37 0 3.929731 0 4.45291945 1.37 0 3.929831 0 4.45292945 1.37 0 3.929931 0 4.45294545  
1.37 0 3.939031 0 4.45314537 0 3.939131 0 4.45324537 0 3.939231 0 4.45334537 0 3.939331 0  
4.45344537 0 3.939431 0 4.45354537 0 3.939531 0 4.45364537 0 3.939631 0 4.45374537 0 3.939731 1.27  
0 0 0 0 0 0 0 4.45384537 0 3.939831 0 4.45394537 0 3.939931 0 4.45404537 0 3.949031 0 4.45414537 0  
3.949131 0 4.45424537 0 3.949231 0 4.45434537 0 3.949331 0 4.45444537 0 3.949431 0 4.45504537 0  
3.949531 0 1.45 0 3.464537 0 3.949631 0 4.45474537 0 3.949731 0 1.45 0 3.484537 0 3.949831 0  
4.45494537 0 3.949931 1.27 0 0 0 0 0 0 1.45 0 3.514537 0 3.959031 0 4.45524537 0 3.959131 0  
4.45534537 0 3.959231 0 4.45544537 0 3.959331 0 4.45554537 0 3.959431 0 4.45564537 0 3.959531 0  
4.45574537 0 3.959631 0 4.45584537 0 3.959731 0 4.45594537 0 3.959831 0 4.45604537 0 3.959931 0  
4.45614537 0 3.969031 0 4.45624537 0 3.969131 1.27 0 0 0 0 0 0 0 4.45634537 0 3.969231 0 4.45644537  
0 3.969331 0 4.45654537 0 3.969431 0 4.45664537 0 3.969531 0 4.45674537 0 3.969631 0 4.45684537 0  
3.969731 0 4.45694537 0 3.969831 0 4.45704537 0 3.969931 0 4.45714537 0 3.979031 0 4.45724537 0



2.5058 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 0 3.572331 0 2.2458 1.27 1.27 3.30405 0 4.12000813  
 4.92925758 1.37 1.27 0 0 0 0 3.190200 0 1.62 0 4.83201708 4.13066614 4.12151104 4.19046420  
 4.19141200 4.19141357 1.58 1.27 0 0 0 0 4.15140813 2.1918 0 1.62 0 2.4241 1.27 1.27 0 0 0 0 4.15170813  
 4.19575086 4.4110214 2.1204 0 2.1914 0 3.190704 0 4.83201708 2.1306 0 4.66141215 4.11041904 0  
 4.66141417 4.3081300 2.1904 0 4.79140813 1.19 0 4.82081220 4.11001914 3.175058 1.27 0 0 0 0  
 4.15170813 4.19575072 4.13181917 4.20021908 4.14131837 2.5058 1.27 0 0 0 0 4.15170813 3.195750 0 0  
 4.22170819 1.4 0 4.42210011 3.200441 0 1.59 0 4.86170819 1.4 0 1.0 0 4.21001120 1.4 0 2.19 0 3.190704  
 0 4.2201717 3.41319 0 4.19001504 0 4.15141808 4.19081413 3.635058 1.27 0 0 0 0 4.15170813 3.195750  
 0 0 4.12142104 0 4.11040519 0 4.42181904 3.151841 0 1.59 0 4.76142104 0 3.190704 0 4.7040003 0  
 4.11040519 0 2.124 0 1.0 0 4.18150402 4.8050804 1.3 0 4.13201201 2.417 0 2.1405 0 4.18190415  
 4.18635058 1.27 0 0 0 0 4.15170813 3.195750 0 0 4.12142104 0 4.17080607 1.19 0 4.42181904 3.151841 0  
 1.59 0 4.76142104 0 3.190704 0 4.7040003 0 4.17080607 1.19 0 2.124 0 1.0 0 4.18150402 4.8050804 1.3 0  
 4.13201201 2.417 0 2.1405 0 4.18190415 4.18635058 1.27 0 0 0 0 4.15170813 3.195750 0 0 4.6041304  
 4.17001904 0 1.59 0 4.70041304 4.17001904 0 1.0 0 4.15140813 1.19 0 4.1001804 1.3 0 2.1413 0 3.190704  
 0 4.2201717 3.41319 0 4.19001504 0 3.1303 0 4.7040003 0 4.15141808 4.19081413 3.635058 1.27 0 0 0 0  
 4.15170813 3.195750 0 0 4.4230819 0 1.59 0 4.68230819 0 3.190704 0 4.15171406 4.17001263 2.5058  
 1.27 0 0 0 0 4.15170813 3.195758 1.27 1.27 0 0 0 0 4.22070811 1.4 0 4.83172004 1.37 1.27 0 0 0 0 0 0 0  
 4.2141212 3.1303 0 1.62 0 4.8131520 4.19575068 4.13190417 0 4.2141212 4.130337 0 4.50586318  
 4.19170815 4.57586311 4.14220417 2.5758 1.27 0 0 0 0 0 0 0 4.15001719 1.18 0 1.62 0 4.2141212  
 4.130363 4.18151108 3.195758 1.27 0 0 0 0 0 0 0 4.21908 2.1413 0 1.62 0 4.15001719 4.18429041 1.27  
 1.27 0 0 0 0 0 0 0 2.805 0 4.21908 2.1413 0 2.6262 0 4.50221708 3.190450 0 3.1303 0 4.11041357  
 4.15001719 2.1858 0 1.34 0 2.9137 1.27 0 0 0 0 0 0 0 0 0 0 4.21001120 1.4 0 1.62 0 4.5111400  
 4.19571500 4.17191842 3.914158 1.27 0 0 0 0 0 0 0 0 0 0 4.19020063 4.22170819 4.4601900  
 4.15045721 4.112004 1.58 1.27 0 0 0 0 0 0 0 4.4110805 0 4.21908 2.1413 0 2.813 0 4.42501214  
 4.21045031 0 4.50121421 3.45041 0 3.1303 0 4.11041357 4.15001719 2.1858 0 2.6262 0 2.9337 1.27 0 0 0  
 0 0 0 0 0 0 0 0 4.3081704 4.2190814 1.13 0 1.62 0 4.15001719 4.18429141 1.27 0 0 0 0 0 0 0 0 0 0  
 4.18190415 1.18 0 1.62 0 4.8131957 4.15001719 4.18429241 1.58 1.27 0 0 0 0 0 0 0 0 0 0 4.19020063  
 4.12142104 4.60070400 4.3570308 4.17040219 4.8141331 0 4.18190415 2.1858 1.27 0 0 0 0 0 0 0 0  
 4.4110805 0 4.21908 2.1413 0 2.6262 0 4.50060413 4.4170019 3.45037 1.27 0 0 0 0 0 0 0 0 0 0 0  
 4.15140813 4.19186300 4.15150413 4.3571902 4.630604 4.13041700 4.19046015 4.14081319 3.575858  
 1.27 0 0 0 0 0 0 0 4.4110805 0 4.21908 2.1413 0 2.6262 0 4.50042308 3.195037 1.27 0 0 0 0 0 0 0 0 0 0  
 0 4.15170813 4.19575068 4.23081908 4.13066363 3.635058 1.27 0 0 0 0 0 0 0 0 0 0 4.1170400 1.10  
 1.27 0 0 0 0 0 0 0 4.4111804 1.37 1.27 0 0 0 0 0 0 0 0 0 4.15170813 4.19575084 4.13101314 2.2213  
 0 4.2141212 4.130363 0 4.79110400 2.1804 0 3.191724 0 4.60008 4.13635058 1.27 1.27 0 0 0 0  
 4.15170813 4.19575070 4.4130417 4.190403 0 4.66141417 4.3081300 4.19041837 2.5031 0 4.15140813  
 3.191858 1.27 0 0 0 0 4.15111419 4.60021414 4.17030813 4.190418 4.92925715 4.14081319 2.1858 1.27  
 1.27 3.30405 0 4.15111419 4.60021414 4.17030813 4.190418 4.92925715 4.14081319 3.185837 1.27 0 0 0  
 0 4.23602100 3.111831 0 4.24602100 2.1118 0 1.62 0 4.25081557 4.56151408 4.13191858 0 2.805 0  
 4.15140813 2.1918 0 4.4111804 0 4.57424131 0 3.424158 1.27 0 0 0 0 3.50806 0 1.62 0 4.6146369  
 4.806217 4.4570300 4.19006206 4.14638202 4.191904 4.17572362 4.23602100 3.111831 0 4.24622460  
 4.21001118 1.31 0 4.12140304 4.62451200 4.17100417 4.18611108 4.13041845 1.31 0 4.13001204  
 4.62456614 4.14170308 4.13001904 4.18455858 1.27 0 0 0 0 4.5080663 4.20150300 4.19046011 4.241420  
 4.19571908 4.19110462 4.45721319 4.4170002 4.19082104 0 4.79111419 0 2.1405 0 4.70041304  
 4.17001904 1.3 0 4.66141417 4.3081300 4.19041845 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
 4.23002308 4.18601908 4.19110462 2.4587 0 4.64230818 2.4531 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
 0 0 4.24002308 4.18601908 4.19110462 2.4588 0 4.64230818 2.4558 1.27 0 0 0 0 4.5080663 4.18071422  
 2.5758 1.27 0 0 0 0 1.39 0 4.78151908 4.14130011 3.112431 0 3.241420 0 3.20013 0 4.18002104 0  
 3.190704 0 4.15111419 0 2.18 0 2.13 0 4.71837675 0 4.5081104 1.27 0 0 0 0 4.5080663 4.22170819  
 4.4600719 4.12115745 4.15111419 4.63071912 3.114558 1.27 0 0 0 0 4.15170813 4.19575079 3.111419 0  
 4.18002104 1.3 0 2.18 0 4.45151114 4.19630719 4.12114563 2.5058 1.27 1.27 3.30405 0 4.8186015



4.63151708 4.12041863 4.151504 4.13035725 4.60210011 1.58 1.27 1.27 1.39 0 4.79170415 3.1704 0  
2.9367 0 4.2141417 4.3081300 3.190418 1.27 3.30405 0 4.15170415 4.170460 4.2141417 4.3081300  
4.19041892 4.91571517 4.8120418 2.5837 1.27 0 0 0 0 4.2141417 2.318 0 1.62 0 2.4241 1.27 0 0 0  
3.51417 0 1.8 0 2.813 0 4.17001306 4.4579031 0 4.11041357 4.15170812 4.4185831 0 3.935837 1.27 0 0 0  
0 0 0 0 0 4.2072013 1.10 0 1.62 0 4.15170812 4.4184208 4.37086193 1.41 1.27 0 0 0 0 0 0 0 2.805 0  
4.11041357 4.2072013 2.1058 0 1.33 0 2.9337 1.27 0 0 0 0 0 0 0 0 0 0 4.2072013 1.10 0 2.6162 0  
3.429141 0 1.56 0 2.5793 0 1.59 0 4.11041357 4.2072013 3.105858 1.27 0 0 0 0 0 0 0 0 4.2141417  
4.3186300 4.15150413 4.3570207 4.20131058 1.27 0 0 0 0 4.17041920 2.1713 0 4.2141417 2.318 1.27 1.27  
1.39 0 4.64130812 4.190814 1.13 0 4.18041920 1.15 1.27 3.30405 0 4.130812 4.190492 4.91570858 1.37  
1.27 0 0 0 0 4.236302 4.11040017 2.5758 1.27 0 0 0 0 4.236318 4.2001919 4.4175756 4.25081557  
4.56021414 4.17031842 4.37086191 3.415831 0 4.2624517 2.4531 0 4.12001710 4.4176245 3.144558 1.27  
0 0 0 0 4.236318 4.4196023 4.11081257 2.9031 0 4.12002357 4.2141417 3.31831 0 4.10042462  
4.11001201 2.300 0 2.2337 0 4.23429041 4.58429041 0 1.61 0 3.919058 1.27 0 0 0 0 4.236318 4.4196024  
4.11081257 2.9031 0 4.12002357 4.2141417 3.31831 0 4.10042462 4.11001201 2.300 0 2.2337 0  
4.23429141 4.58429141 0 1.61 0 3.919058 1.27 0 0 0 0 4.236318 4.4196025 4.11081257 2.9031 0  
4.12002357 4.2141417 3.31831 0 4.10042462 4.11001201 2.300 0 2.2337 0 4.23429241 4.58429241 0 1.61  
0 3.919058 1.27 1.27 4.39393939 4.39393939 4.39393939 4.39393939 3.393939 1.27 1.27 1.27 3.30405 0  
4.8130304 4.23601914 4.60170601 4.91995702 4.14111417 4.60081303 4.4235837 1.27 0 0 0  
4.50505066 4.14132104 2.1719 0 1.0 0 4.2141114 1.17 0 4.8130304 1.23 0 2.1914 0 2.13 0 3.817065 0  
4.19201511 4.4635050 1.50 1.27 0 0 0 0 3.170403 0 1.62 0 4.57021411 4.14176008 4.13030423 0 2.3434 0  
3.919658 0 1.55 0 3.929595 0 0 1.39 0 4.82070805 1.19 0 4.17080607 1.19 0 2.124 0 2.9196 0 4.1081918 0  
3.1303 0 4.12001810 0 4.22081907 0 3.929595 0 2.1914 0 3.60419 0 3.190704 0 3.170403 0 4.2141215  
4.14130413 1.19 1.27 0 0 0 0 4.6170404 1.13 0 1.62 0 4.57021411 4.14176008 4.13030423 0 2.3434 0  
2.9858 0 1.55 0 3.929595 0 0 1.39 0 4.82070805 1.19 0 4.17080607 1.19 0 2.124 0 1.98 0 4.1081918 0  
3.1303 0 4.12001810 0 4.22081907 0 3.929595 0 2.1914 0 3.60419 0 3.190704 0 4.6170404 1.13 0  
4.2141215 4.14130413 1.19 1.27 0 0 0 0 4.1112004 0 1.62 0 4.2141114 4.17600813 3.30423 0 1.55 0  
3.929595 0 0 1.39 0 4.76001810 0 4.22081907 0 3.929595 0 2.1914 0 3.60419 0 3.190704 0 4.1112004 0  
4.2141215 4.14130413 1.19 1.27 0 0 0 0 4.17041920 2.1713 0 4.57170403 1.31 0 4.6170404 2.1331 0  
4.1112004 1.58 1.27 1.27 3.30405 0 4.2001102 4.20110019 4.4600214 4.13050806 4.20170019 4.8141360  
4.8130304 4.23919957 4.18162000 4.17046002 4.14111417 4.60081303 3.42331 0 4.18241201  
4.14116008 4.13030423 1.31 0 4.5141319 4.60021411 4.14176008 4.13030423 2.5837 1.27 0 0 0  
4.77847660 4.82887665 3.787582 0 1.62 0 2.9997 1.27 0 0 0 0 4.77847660 4.69787783 4.60667875  
3.788182 0 1.62 0 4.92959656 2.5693 1.27 0 0 0 0 4.8130304 1.23 0 1.62 0 4.57181620 4.170460  
4.2141114 4.17600813 3.30423 0 1.56 0 4.77847660 4.82887665 3.787582 0 1.56 0 4.77847660  
4.69787783 4.60667875 4.78818258 0 1.61 0 4.57182412 4.1141160 4.8130304 1.23 0 1.56 0 4.77847660  
4.69787783 4.60667875 4.78818258 0 1.61 0 4.5141319 4.60021411 4.14176008 4.13030423 1.27 0 0 0  
4.17041920 2.1713 0 4.8130304 1.23 1.27 1.27 3.30405 0 4.18002104 4.60021413 4.5080620 4.17001908  
4.14136019 4.14600508 4.11049199 4.57050811 4.4601300 3.120431 0 4.6170803 4.60180825 2.431 0  
4.18162000 4.17046018 4.8250431 0 4.5141319 4.60180825 2.431 0 4.2141305 4.8062017 4.190814  
4.13185837 1.27 0 0 0 0 4.22081907 0 4.14150413 4.57050811 4.4601300 3.120431 0 4.45224558 0 2.18 0  
4.5081104 1.37 1.27 0 0 0 0 0 4.5081104 4.63221708 4.19045705 4.50701708 1.3 0 4.82082504 1.37  
0 4.43061708 4.3601808 4.25044423 4.43061708 4.3601808 4.25044429 3.135058 1.27 0 0 0 0 0 0 0  
4.5081104 4.63221708 4.19045705 4.50822001 4.59181620 3.1704 0 4.79082304 1.11 0 4.75041306  
3.190737 0 4.43181620 4.170460 4.18082504 4.44291350 1.58 1.27 0 0 0 0 0 0 0 4.5081104 4.63221708  
4.19045705 4.50691413 1.19 0 4.82082504 1.37 0 4.43051413 4.19601808 4.25044429 3.135058 1.27 0 0 0  
0 0 0 0 0 4.5081104 4.63221708 4.19045750 4.66141305 4.8062017 4.190814 4.13183729 3.135058 1.27 0  
0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 3.51417 0 4.2141305 2.806 0 2.813 0 4.2141305 4.8062017 4.190814  
3.131837 1.27 0 0 0 0 0 0 0 0 0 0 4.18162000 4.17046002 4.14111417 4.60081303 3.42331 0  
4.18241201 4.14116008 4.13030423 1.31 0 4.5141319 4.60021411 4.14176008 4.13030423 0 1.62 0  
4.2141305 2.806 1.27 0 0 0 0 0 0 0 0 0 4.8130304 1.23 0 1.62 0 4.2001102 4.20110019 4.4600214

4.13050806 4.20170019 4.8141360 4.8130304 4.23919957 4.18162000 4.17046002 4.14111417  
4.60081303 3.42331 0 4.18241201 4.14116008 4.13030423 1.31 0 4.5141319 4.60021411 4.14176008  
4.13030423 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 0 1.39 0 4.66141321 3.41719 0 4.8130308 3.20418 0 2.1914 0  
3.817065 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.18162000 4.17046002 4.14111417 4.60170601 0 1.62 0 4.8130304  
4.23601914 4.60170601 4.91995718 4.16200017 4.4600214 4.11141760 4.8130304 2.2358 1.27 0 0 0 0 0 0  
0 0 0 0 0 4.5141319 4.60021411 4.14176017 2.601 0 1.62 0 4.8130304 4.23601914 4.60170601  
4.91995705 4.14131960 4.2141114 4.17600813 4.3042358 1.27 0 0 0 0 0 0 0 0 0 0 0 4.5081104  
4.63221708 4.19045705 4.50721303 3.42337 0 4.43081303 4.4234431 0 4.82162000 2.1704 0 4.66141114  
1.17 0 4.72130304 2.2337 0 4.43181620 4.170460 4.2141114 4.17600813 4.3042344 0 4.57817065 1.37 0  
4.43181620 4.170460 4.2141114 4.17601706 4.1445831 0 4.82241201 3.141137 0 4.43182412 4.1141160  
4.8130304 3.234431 0 4.69141319 0 4.66141114 1.17 0 4.72130304 2.2337 0 4.43051413 4.19600214  
4.11141760 4.8130304 2.2344 0 4.57817065 1.37 0 4.43051413 4.19600214 4.11141760 4.17060144  
4.58291350 1.58 1.27 1.27 3.30405 0 4.8186015 4.17081204 4.91985713 4.20125837 1.27 0 0 0 0  
4.50505066 4.7040210 0 2.805 0 1.00 0 4.13201201 2.417 0 2.818 0 4.15170812 2.463 0 4.64181820  
3.120418 0 3.132012 0 2.818 0 1.00 0 4.15141808 4.19082104 0 4.8131904 4.6041763 3.505050 1.27 0 0 0 0  
2.805 0 3.132012 0 1.33 0 2.9237 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.69001118 1.4 1.27 0 0 0 0  
3.51417 0 1.8 0 2.813 0 4.17001306 4.4579231 0 4.8131957 4.12001907 4.63181617 4.19571320 3.125858  
0 1.61 0 3.915837 1.27 0 0 0 0 0 0 2.805 0 3.132012 0 1.53 0 1.8 0 2.6262 0 2.9037 1.27 0 0 0 0 0 0 0  
0 0 0 4.17041920 2.1713 0 4.69001118 1.4 1.27 0 0 0 0 4.17041920 2.1713 0 4.83172004 1.27 1.27 3.30405  
0 4.12000813 4.91985758 1.37 1.27 0 0 0 0 1.39 0 4.72131520 1.19 0 1.00 0 3.1303 0 1.1 1.27 0 0 0 0 1.00  
1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 1.00 0 2.5700 0 1.34 0 3.15837 0 3.505858 1.27 0 0 0  
0 1.1 0 1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 1.1 0 2.5701 0 1.33 0 3.5837 0 3.505858  
1.27 1.27 0 0 0 0 1.39 0 4.85001108 4.3001904 0 4.2141303 4.8190814 2.1318 0 3.51417 0 1.00 0 3.1303 0  
1.1 1.27 0 0 0 0 2.805 0 3.131419 0 2.5700 0 1.34 0 1.1 0 3.1303 0 1.90 0 1.33 0 2.5700 0 1.59 0 2.158 0  
2.3362 0 4.91925837 1.27 0 0 0 0 0 0 0 4.15170813 4.19575083 2.704 0 4.2141303 4.8190814 2.1318 0  
1.90 0 1.33 0 2.5700 0 1.59 0 2.158 0 2.3362 0 2.9192 0 3.1303 0 1.00 0 1.34 0 1.1 0 3.1704 0 3.131419 0  
4.12041963 2.5058 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 1.27 0 0 0 0 1.27 0 0 0 0 1.39 0 4.66001102  
4.20110019 1.4 0 2.1531 0 3.190704 0 3.182012 0 4.5171412 0 1.8 0 1.62 0 1.00 0 2.1914 0 1.8 0 1.62 0 1.1 0  
2.1405 0 2.5708 0 1.61 0 2.9158 1.27 0 0 0 0 1.15 0 1.62 0 4.18201257 1.8 0 1.61 0 1.91 0 3.51417 0 1.8 0  
2.813 0 4.17001306 4.4570131 0 1.00 0 1.61 0 3.915858 1.27 0 0 0 0 1.27 0 0 0 0 1.39 0 4.66001102  
4.20110019 1.4 0 1.25 1.27 0 0 0 0 1.25 0 1.62 0 4.12001907 4.63181617 3.195715 0 1.32 0 2.5700 0 1.32 0  
3.925858 1.27 0 0 0 0 1.27 0 0 0 0 1.39 0 4.66001102 4.20110019 1.4 0 4.4151808 3.111413 0 4.57030402  
4.8120011 0 4.15001719 0 2.1405 0 2.2558 1.27 0 0 0 0 4.4151808 4.11141391 0 1.62 0 1.25 0 1.59 0  
4.8131957 2.2558 1.27 0 0 0 0 4.4151808 4.11141392 0 1.62 0 1.91 0 1.59 0 4.4151808 4.11141391 1.27 0  
0 0 0 4.4151808 4.11141393 0 1.62 0 1.91 1.27 0 0 0 0 1.27 0 0 0 0 1.39 0 4.79170415 3.1704 0 3.190704 0  
4.15141818 4.8011104 0 1.25 0 4.21001120 2.418 0 2.1914 0 4.2070402 1.10 0 3.51417 0 4.15170812  
4.110819 1.24 1.27 0 0 0 0 4.25602100 4.11200418 0 1.62 0 1.43 1.27 0 0 0 0 0 0 2.5025 0 1.61 0  
4.4151808 4.11141392 2.5037 0 1.25 0 1.61 0 4.4151808 4.11141392 1.31 1.27 0 0 0 0 0 0 2.5025 0  
1.59 0 4.4151808 4.11141391 2.5037 0 1.25 0 1.59 0 4.4151808 4.11141391 1.31 1.27 0 0 0 0 0 0  
2.5025 0 1.59 0 4.4151808 4.11141392 2.5037 0 1.25 0 1.59 0 4.4151808 4.11141392 1.31 1.27 0 0 0 0 0  
0 0 0 0 2.5025 0 1.61 0 4.4151808 4.11141391 2.5037 0 1.25 0 1.61 0 4.4151808 4.11141391 1.31 1.27 0 0 0  
0 0 0 0 0 2.5025 0 1.59 0 4.4151808 4.11141393 2.5037 0 1.25 0 1.61 0 4.4151808 4.11141393 1.31 1.27 0 0 0  
0 1.44 1.27 0 0 0 0 1.27 0 0 0 0 1.39 0 3.700419 0 4.5081104 0 4.13001204 0 4.5171412 0 4.20180417 0  
3.1303 0 4.22170819 1.4 0 3.190704 0 4.17041820 3.111918 1.27 0 0 0 0 4.5081104 4.60130012 1.4 0 1.62 0  
0 4.8131520 4.19575068 4.13190417 0 3.190704 0 4.13001204 0 2.1405 0 3.190704 0 4.5081104 0 2.1914  
0 4.18002104 0 4.17041820 4.11191837 0 2.5058 1.27 0 0 0 0 4.22081907 0 4.14150413 4.57050811  
4.4601300 3.120431 0 4.50225058 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 4.5081104 4.63221708  
4.19045705 4.50721315 2.2019 0 4.21001120 3.41837 0 1.00 0 1.62 0 4.43004431 0 1.1 0 1.62 0 4.43014429  
3.135058 1.27 0 0 0 0 0 0 0 0 4.5081104 4.63221708 4.19045705 4.50660011 4.2201100 3.190403 0 2.1537

0 4.43154429 3.135058 1.27 0 0 0 0 0 0 0 4.5081104 4.63221708 4.19045705 4.50660011 4.2201100  
3.190403 0 2.2537 0 4.43254429 3.135058 1.27 0 0 0 0 0 0 4.5081104 4.63221708 4.19045705  
4.50660011 4.2201100 3.190403 0 4.4151808 4.11141391 1.37 0 4.43041518 4.8111413 4.91442913  
2.5058 1.27 0 0 0 0 0 0 4.5081104 4.63221708 4.19045705 4.50660011 4.2201100 3.190403 0  
4.4151808 4.11141392 1.37 0 4.43041518 4.8111413 4.92442913 2.5058 1.27 0 0 0 0 0 0 4.5081104  
4.63221708 4.19045705 4.50041518 4.8111413 2.9337 0 4.43041518 4.8111413 4.93442913 2.5058 1.27 0  
0 0 0 0 0 0 3.51417 0 4.11000104 2.1131 0 4.21001120 1.4 0 2.813 0 4.25602100 4.11200418 4.63081904  
4.12185758 1.37 1.27 0 0 0 0 0 0 0 0 0 0 2.805 0 4.8131957 4.21001120 2.458 0 2.6262 0 4.21001120  
2.437 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.15170812 4.4601819 4.192018 0 1.62 0 4.8186015 4.17081204  
4.91985708 4.13195721 4.112004 2.5858 1.27 0 0 0 0 0 0 0 0 0 0 4.4111804 1.37 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.15170812 4.4601819 4.192018 0 1.62 0 4.50773264 0 4.57131419 0 2.13 0 4.8131904  
4.6041758 1.50 1.27 0 0 0 0 0 0 0 0 0 0 0 4.5081104 4.63221708 4.19045705 3.507218 0 4.43110001  
3.411144 0 1.62 0 4.43210011 3.200444 0 4.15170812 2.435 0 4.43151708 4.12046018 4.19001920  
4.18442913 2.5058 1.27 1.27 3.30405 0 4.151504 4.13036018 4.19170813 4.6601914 4.60050811  
4.4925703 4.8170402 4.19141724 1.31 0 4.5081104 4.13001204 1.31 0 4.20180417 4.60181917 4.8130658  
1.37 1.27 0 0 0 0 4.50505064 4.15150413 2.318 0 3.190704 0 4.20180417 0 4.18191708 2.1306 0 2.1914 0  
1.0 0 4.5081104 0 2.805 0 2.819 0 4.3140418 0 3.131419 0 4.111704 3.324 0 4.4230818 1.19 0 2.813 0  
3.190704 0 4.5081104 4.63505050 1.27 0 0 0 0 1.39 0 4.66170400 2.1904 0 4.5201111 0 4.5081104 0  
4.15001907 1.27 0 0 0 0 4.5081104 4.60150019 1.7 0 1.62 0 4.14186315 4.190763 4.9140813 4.57030817  
4.4021914 3.172431 0 4.5081104 4.13001204 1.58 1.27 0 0 0 0 1.27 0 0 0 0 1.39 0 4.66070402 1.10 0 2.805  
0 4.5081104 0 4.4230818 2.1918 0 3.1303 0 4.17040003 0 3.81918 0 4.2141319 4.4131918 0 2.805 0 2.819  
0 4.3140418 1.27 0 0 0 0 4.8055714 4.18631500 4.19076304 4.23081819 4.18570508 4.11046015 4.190758  
0 3.1303 0 4.11041357 4.20180417 4.60181917 4.8130658 0 1.34 0 3.945837 1.27 0 0 0 0 0 0 0  
4.22081907 0 4.14150413 4.57050811 4.4601500 3.190731 0 4.45174531 0 4.4130214 4.3081306  
4.62452019 4.5599845 1.58 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 0 2.805 0 4.20180417  
4.60181917 3.81306 0 2.813 0 4.5081104 4.63170400 4.3575837 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
4.15170813 4.19575083 2.704 0 4.18191708 2.1306 0 2.818 0 4.111704 3.324 0 2.813 0 3.190704 0  
4.5081104 3.635058 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.17041920 2.1713 1.27 0 0 0 0 4.4111804 1.37 1.27  
0 0 0 0 0 0 0 4.15170813 4.19570550 4.29132913 4.43050811 4.4130012 2.444 0 4.3140418 0 4.4230818  
4.19635058 1.27 0 0 0 0 0 0 0 4.15170813 4.19575029 4.13291977 4.14190437 0 4.88142017 0  
4.18191708 2.1306 0 4.12201819 0 2.104 0 4.6170400 3.190417 0 4.19070013 0 1.94 0 4.2070017 4.21904  
2.1718 0 2.813 0 4.11041306 3.190763 0 4.70041304 4.17001904 0 4.91631923 2.1931 0 4.92631923  
2.1931 0 4.93631923 2.1931 0 4.94631923 1.19 0 2.813 0 4.11110013 4.6200006 4.4761403 4.32120015  
4.15081306 1.18 0 4.20180813 1.6 0 4.12140304 0 2.9192 0 4.63636329 4.13291350 1.58 1.27 0 0 0 0 1.27  
0 0 0 2.805 0 4.11041357 4.20180417 4.60181917 4.8130658 0 1.34 0 2.9437 1.27 0 0 0 0 0 0 1.39 0  
4.69081718 2.1931 0 4.3041904 4.17120813 1.4 0 3.190704 0 4.2201717 3.41319 0 4.13201201 2.417 0  
2.1405 0 4.11081304 1.18 0 2.813 0 3.190704 0 4.5081104 1.27 0 0 0 0 0 0 0 2.805 0 4.14186315  
4.190763 4.4230818 4.19185705 4.8110460 4.15001907 2.5837 1.27 0 0 0 0 0 0 0 0 0 4.22081907 0  
4.14150413 4.57050811 4.4601500 3.190731 0 4.45174531 0 4.4130214 4.3081306 4.62452019 4.5599845  
1.58 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 4.2201717 4.4131960 4.11081304  
4.60132012 3.10417 0 1.62 0 4.18201257 1.91 0 3.51417 0 1.60 0 2.813 0 4.5081104 1.58 0 1.61 0 1.91  
1.27 0 0 0 0 0 0 0 4.4111804 1.37 1.27 0 0 0 0 0 0 0 0 0 4.2201717 4.4131960 4.11081304  
4.60132012 3.10417 0 1.62 0 1.91 0 0 1.39 0 4.69081104 0 4.3140418 3.134519 0 4.4230818 1.19 0  
4.24041931 0 2.1814 0 4.18190017 1.19 0 4.5171412 0 4.11081304 0 1.91 1.27 1.27 0 0 0 0 0 0 1.39 0  
4.64151504 2.1303 0 3.190704 0 4.18191708 2.1306 0 2.1914 0 3.190704 0 4.5081104 1.31 0 4.30308  
2.1306 0 3.190704 0 4.11081304 0 4.13201201 2.417 0 4.1040514 2.1704 0 3.190704 0 4.18191708 2.1306  
1.27 0 0 0 0 0 0 4.22081907 0 4.14150413 4.57050811 4.4601500 3.190731 0 4.45004531 0 4.4130214  
4.3081306 4.62452019 4.5599845 1.58 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 4.5081104  
4.63221708 4.19045705 4.50430220 4.17170413 4.19601108 4.13046013 4.20120104 2.1744 0  
4.43201804 4.17601819 4.17081306 4.44291350 1.58 1.27 0 0 0 0 0 0 0 0 4.15170813 4.19570550

4.82191708 2.1306 0 4.30304 1.3 0 2.1914 0 4.43050811 4.4130012 4.4445058 1.27 1.27 1.27 3.30405 0  
4.12000813 4.92575837 1.27 0 0 0 1.39 0 4.81040003 0 4.20180417 0 4.8131520 1.19 1.27 0 0 0 0  
4.20180417 4.60181917 3.81306 0 1.62 0 4.8131520 4.19575079 4.11040018 1.4 0 4.4131904 1.17 0 1.0 0  
4.18191708 3.130637 0 2.5058 1.27 0 0 0 1.27 0 0 0 0 1.39 0 4.66001102 4.20110019 1.4 0 3.190704 0  
4.11041306 2.1907 0 2.1405 0 3.190704 0 4.18191708 2.1306 1.27 0 0 0 1.13 0 1.62 0 4.11041357  
4.20180417 4.60181917 4.8130658 1.27 0 0 0 1.27 0 0 0 0 1.39 0 4.67040508 2.1304 0 4.3081704  
4.2191417 1.24 0 3.1303 0 4.5081104 4.13001204 1.27 0 0 0 0 4.3081704 4.2191417 1.24 0 1.62 0  
4.50111100 4.13062000 4.6047614 4.3321200 4.15150813 3.61850 1.27 0 0 0 0 4.5081104 4.13001204 0  
1.62 0 4.5504313 4.44631923 2.1950 1.27 0 0 0 0 1.27 0 0 0 0 1.39 0 4.68131820 2.1704 0 3.190704 0  
4.3081704 4.2191417 1.24 0 4.4230818 2.1918 1.27 0 0 0 0 2.805 0 3.131419 0 4.14186315 4.190763  
4.4230818 4.19185703 4.8170402 4.19141724 2.5837 1.27 0 0 0 0 0 0 0 4.14186312 4.100403 4.8171857  
4.3081704 4.2191417 2.2458 1.27 0 0 0 0 0 0 0 4.15170813 4.19570550 4.66170400 3.190403 0  
4.3081704 4.2191417 1.24 0 4.43030817 4.4021914 4.17244450 1.58 1.27 0 0 0 0 1.27 0 0 0 0 1.39 0  
4.64151504 2.1303 0 4.18191708 2.1306 0 2.1914 0 3.190704 0 4.151517 4.14151708 3.1904 0 4.5081104  
1.27 0 0 0 0 4.151504 4.13036018 4.19170813 4.6601914 4.60050811 4.4925703 4.8170402 4.19141724  
1.31 0 4.5081104 4.13001204 1.31 0 4.20180417 4.60181917 4.8130658 1.27 1.27 3.30405 0 4.1001804  
4.60236019 4.14600108 4.13001724 4.57132012 1.31 0 4.1001804 2.5837 1.27 0 0 4.50505066 4.14132104  
3.171918 0 1.0 0 4.1001804 0 1.23 0 4.8131904 3.60417 0 2.1914 0 1.0 0 4.1081300 2.1724 0 4.18191708  
4.13066350 2.5050 1.27 0 0 4.3080608 2.1918 0 1.62 0 4.50909192 4.93949596 4.97989964 4.65666768  
2.6950 0 1.39 0 2.8415 0 2.1914 0 4.1001804 0 2.9196 1.27 0 0 2.805 0 4.1001804 0 1.33 0 1.92 0 2.1417 0  
4.1001804 0 1.34 0 3.919637 1.27 0 0 0 0 4.17000818 1.4 0 4.85001120 4.4681717 4.14175750 4.65001804  
0 4.12201819 0 2.104 0 4.1041922 3.40413 0 1.92 0 3.1303 0 4.91965058 1.27 1.27 0 0 4.1081300 2.1724 0  
1.62 0 2.5050 1.27 0 0 4.22070811 1.4 0 3.132012 0 1.34 0 2.9037 1.27 0 0 0 0 4.1081300 2.1724 0 1.62 0  
4.3080608 4.19184213 2.2012 0 1.53 0 2.9241 0 1.61 0 4.1081300 2.1724 1.27 0 0 0 0 3.132012 0 3.323262  
0 1.92 1.27 0 0 4.17041920 2.1713 0 4.1081300 2.1724 1.27 1.27 1.27 3.30405 0 4.21413213 4.4171960  
4.5081104 4.57081315 4.20196005 4.8110431 0 4.14201915 4.20196005 4.8110431 0 4.1001804 2.5837  
1.27 0 0 4.50505066 4.14132104 3.171918 0 1.0 0 4.5081104 0 2.1405 0 4.1001804 0 1.23 0 4.8131904  
4.6041718 0 2.1914 0 1.0 0 4.5081104 0 2.1405 0 4.1081300 2.1724 0 4.18191708 4.13061863 1.27 1.27 0  
0 4.64170618 1.37 1.27 0 0 0 0 0 4.8131520 4.19600508 3.110437 0 4.79001907 0 2.1914 0 3.190704 0  
4.8131520 1.19 0 4.5081104 1.63 1.27 0 0 0 0 0 4.14201915 4.20196005 4.8110437 0 4.79001907 0  
2.1914 0 3.190704 0 4.14201915 2.2019 0 4.5081104 1.63 1.27 0 0 0 0 0 4.1001804 1.37 0 3.830704 0  
4.1001804 0 2.1405 0 3.190704 0 4.8131904 4.6041718 0 2.813 0 3.190704 0 4.8131520 1.19 0 4.5081104  
1.63 1.27 0 0 3.505050 1.27 0 0 4.22081907 0 4.14150413 4.57081315 4.20196005 4.8110431 0  
4.45174558 0 2.18 0 4.5600813 1.31 0 4.14150413 4.57142019 4.15201960 4.5081104 1.31 0 4.45224558 0  
2.18 0 4.5601420 2.1937 1.27 0 0 0 0 3.51417 0 4.11081304 0 2.813 0 4.5600813 1.37 1.27 0 0 0 0 0 0  
4.13201201 3.41718 0 1.62 0 4.42081319 3.572358 0 3.51417 0 1.23 0 2.813 0 4.11081304 4.63181917  
4.8155758 4.63181511 4.8195758 1.41 1.27 0 0 0 0 0 4.1081300 4.17246018 4.19170813 2.618 0 1.62 0  
4.42010018 4.4602360 4.19146001 4.8130017 4.24571320 2.1231 0 4.1001804 1.58 0 3.51417 0 3.132012  
0 2.813 0 4.13201201 4.4171841 1.27 0 0 0 0 0 4.5601420 4.19632217 4.8190457 1.50 0 4.50630914  
4.8135701 4.8130017 4.24601819 4.17081306 2.1858 0 1.61 0 4.50291350 1.58 1.27 1.27 3.30405 0  
4.6041304 4.17001904 4.60001303 4.60180021 4.4601360 4.1081960 4.2141201 4.8130019 4.8141318  
4.57050811 4.4601500 3.190731 0 3.135837 1.27 0 0 0 0 4.22081907 0 4.14150413 4.57050811 4.4601500  
3.190731 0 4.50225058 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 3.51417 0 1.8 0 2.813 0 4.17001306  
4.4579256 4.56135837 0 0 1.39 0 3.925413 0 4.2141201 4.8130019 4.8141318 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.2141201 4.8130019 3.81413 0 1.62 0 4.1081357 4.8584292 4.37416325 4.5081111 3.571358 1.27 0 0 0  
0 0 0 0 0 0 0 4.5081104 4.63221708 4.19045702 4.14120108 4.13001908 2.1413 0 1.61 0 4.50291350  
1.58 1.27 1.27 3.30405 0 4.3040214 4.12150811 4.4600300 4.19005758 1.37 1.27 0 0 0 0 4.50505067  
4.4021412 4.15081104 1.18 0 4.3001900 0 4.20180813 1.6 0 1.0 0 4.3241300 4.12080200 3.111124 0  
4.11140003 2.403 0 4.3040214 4.12150811 1.4 0 4.5201302 4.19081413 4.63505050 1.27 0 0 0 0  
4.13001204 0 1.62 0 4.8131520 4.19575068 4.13190417 0 3.190704 0 4.13001204 0 2.1405 0 3.190704 0

4.3001900 0 4.18191720 4.2192017 1.4 0 2.1914 0 4.3040214 4.12150811 2.437 0 2.5058 1.27 0 0 0 0  
4.5081104 4.13001204 0 1.62 0 4.5504313 4.120444 4.60111406 4.8026315 2.2450 1.27 0 0 0 0 2.805 0  
3.131419 0 4.14186315 4.190763 4.4230818 4.19185705 4.8110413 4.120458 1.37 1.27 0 0 0 0 0 0 0  
4.15170813 4.19570550 4.75140608 1.2 0 4.18021708 2.1519 0 3.51417 0 4.43130012 2.444 0 3.131419 0  
4.5142013 2.363 0 4.79110400 2.1804 0 4.4131820 2.1704 0 3.190704 0 4.3001900 0 4.18191720  
4.2192017 1.4 0 2.818 0 4.3040508 3.130403 0 3.1303 0 4.2141215 4.8110403 0 4.5081718 4.19635058  
1.27 0 0 0 0 0 0 0 4.17041920 2.1713 1.27 0 0 0 0 1.27 0 0 0 0 4.2141215 4.8110403 4.60081315 2.2019 0  
1.62 0 4.8131520 4.19575068 4.13190417 0 4.2141215 4.8110403 0 4.3001900 0 4.18191708 3.130637 0  
2.5058 1.27 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 4.3040214 4.12150811 4.4036003 3.1900 0 1.62 0  
4.4230402 4.20190460 4.3241300 4.12080260 4.18021708 4.15195713 4.120431 0 4.5500304 4.2141215  
4.8110460 4.43130012 4.4445031 0 4.2141215 4.8110403 4.60081315 3.201958 1.27 0 0 0 0 0 0 0  
4.15170813 4.19575067 4.4021412 4.15081104 1.3 0 4.67001900 3.375031 0 4.9181413 4.63032012  
4.15185703 4.4021412 4.15081104 4.3600300 3.190031 0 4.8130304 4.13196294 2.5858 1.27 0 0 0 0  
4.4230204 2.1519 0 4.68230204 4.15190814 1.13 0 2.18 0 2.437 1.27 0 0 0 0 0 0 0 4.15170813  
4.19575064 1.13 0 4.41717114 1.17 0 4.14020220 4.17170403 3.375031 0 2.458 1.27 1.27 3.30405 0  
4.4230402 4.20190460 4.3241300 4.12080260 4.18021708 4.15195713 4.120431 0 4.5201302 4.19081413  
4.60130012 2.431 0 4.3001900 2.5837 1.27 0 0 0 0 4.50505067 4.24130012 4.8020011 2.1124 0 4.8121514  
2.1719 0 3.1303 0 4.4230402 3.201904 0 1.0 0 4.5201302 4.19081413 0 4.5171412 0 1.0 0 4.6041304  
4.17001904 1.3 0 4.18021708 4.15196350 2.5050 1.27 0 0 0 0 4.5081104 4.13001204 0 1.62 0 4.5504313  
4.120444 4.60111406 4.8026315 2.2450 1.27 0 0 0 0 4.18150402 0 1.62 0 4.8121514 4.17191108  
4.1632019 4.8116318 4.15040260 4.5171412 4.60050811 4.4601114 4.2001908 4.14135713 4.120431 0  
4.5081104 4.13001204 1.58 1.27 0 0 0 0 4.12140320 2.1104 0 1.62 0 4.8121514 4.17191108 4.1632019  
4.8116312 4.14032011 4.4600517 4.14126018 4.15040257 4.18150402 1.58 1.27 0 0 0 0 4.18150402  
4.63111400 4.3041763 4.4230402 4.60121403 4.20110457 4.12140320 3.110458 1.27 0 0 0 0 4.5201302 0  
1.62 0 4.6041900 4.19191757 4.12140320 3.110431 0 4.5201302 4.19081413 4.60130012 2.458 1.27 0 0 0  
0 4.17041920 2.1713 0 4.5201302 4.57030019 2.58 1.27 0 0 0 0 1.27 3.30405 0 4.15141520 4.11001904  
4.60030019 4.571300 4.12045837 1.27 0 0 0 0 4.50505079 4.17141215 1.19 0 3.190704 0 4.20180417 0  
2.1914 0 4.4131904 1.17 0 4.3001900 0 3.51417 0 3.190704 0 4.3040508 3.130403 0 4.18191720  
4.2192017 1.4 0 3.1303 0 4.18002104 0 2.1914 0 1.0 0 4.19042319 0 4.5081104 4.63505050 1.27 0 0 0  
4.5081104 4.13001204 0 1.62 0 4.5504313 4.120444 4.63091814 2.1350 1.27 0 0 0 0 2.805 0 3.131419 0  
4.14186315 4.190763 4.4230818 4.19185705 4.8110413 4.120458 1.37 1.27 0 0 0 0 0 0 4.15170813  
4.19570550 4.67001900 0 4.18191720 4.2192017 1.4 0 4.3040508 4.13081908 2.1413 0 3.51417 0  
4.43130012 2.444 0 3.131419 0 4.5142013 2.363 0 4.79110400 2.1804 0 4.3040508 2.1304 0 3.190704 0  
4.3001900 0 4.18191720 4.2192017 1.4 0 4.5081718 4.19635058 1.27 0 0 0 0 0 0 0 4.17041920 2.1713  
1.27 1.27 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 4.22081907 0 4.14150413 4.57050811 4.4130012 2.431 0  
4.45174558 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 4.3001900 4.60181917 4.20021920 2.1704  
0 1.62 0 4.9181413 4.63111400 4.3570508 3.110458 1.27 0 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 4.3001900 0  
1.62 0 2.4344 1.27 0 0 0 0 0 0 3.51417 0 4.5080411 1.3 0 2.813 0 4.3001900 4.60181917 4.20021920  
4.17044245 4.5080411 4.3184541 1.37 1.27 0 0 0 0 0 0 0 0 4.3001900 4.42050804 3.110341 0 1.62  
0 4.8131520 4.19570550 4.68131904 1.17 0 4.21001120 1.4 0 3.51417 0 4.43050804 4.11034437 0 2.5058  
1.27 1.27 0 0 0 0 0 0 0 4.3001900 4.60050811 4.4130012 1.4 0 1.62 0 4.5504313 4.120444 4.6003019  
4.631923 2.1950 1.27 0 0 0 0 0 0 0 4.22081907 0 4.14150413 4.57030019 4.600508 4.11041300 3.120431  
0 4.45004558 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 4.5081104 4.63221708 4.19045745 0 1.47  
0 4.45630914 4.8135718 4.19175703 4.190042 4.5080411 3.34158 0 3.51417 0 4.5080411 1.3 0 2.813 0  
4.3001900 4.60181917 4.20021920 4.17044245 4.5080411 4.3184541 1.58 0 1.61 0 4.45291345 1.58 1.27 0  
0 0 0 0 0 0 4.15170813 4.19570550 4.67001900 0 4.18002104 1.3 0 2.1914 0 4.43030019 4.600508  
4.11041300 4.12044463 2.5058 1.27 0 0 0 1.27 0 0 0 0 4.4230204 2.1519 0 4.69081104 4.77141969  
4.14201303 4.68171714 2.1737 1.27 0 0 0 0 0 0 0 4.15170813 4.19575068 4.17171417 0 4.11140003  
3.81306 0 4.3001900 0 4.18191720 4.2192017 1.4 0 4.5081104 1.63 0 4.79110400 2.1804 0 4.2070402 1.10  
0 3.190704 0 4.5081104 0 4.18241819 4.4126350 1.58 1.27 1.27 1.27 3.30405 0 4.2142013 4.19000111

4.4185758 1.37 1.27 0 0 0 0 4.22070811 1.4 0 4.83172004 1.37 1.27 0 0 0 0 0 0 0 0 4.15170813 4.19575029  
 4.1376008 1.13 0 4.76041320 3.375058 1.27 0 0 0 0 0 0 0 4.15170813 4.19575091 1.63 0 4.67040508  
 2.1304 0 4.67001900 0 4.82191720 4.2192017 3.45058 1.27 0 0 0 0 0 0 0 4.15170813 4.19575092 1.63 0  
 4.79141520 4.11001904 0 4.67001900 2.5058 1.27 0 0 0 0 0 0 0 4.15170813 4.19575093 1.63 0  
 4.66141215 3.81104 0 4.67001900 2.5058 1.27 0 0 0 0 0 0 0 4.15170813 4.19575094 1.63 0 4.67040214  
 4.12150811 1.4 0 4.67001900 2.5058 1.27 0 0 0 0 0 0 0 4.15170813 4.19575095 1.63 0 4.68230819  
 2.5058 1.27 0 0 0 0 0 0 0 4.2071408 2.204 0 1.62 0 4.8131520 4.19575068 4.13190417 0 4.2071408  
 3.20437 0 2.5058 1.27 1.27 0 0 0 0 0 0 0 2.805 0 4.2071408 2.204 0 2.6262 0 4.50915037 1.27 0 0 0 0 0 0  
 0 0 0 0 0 4.3040508 4.13046003 4.190060 4.18191720 4.2192017 3.45758 1.27 0 0 0 0 0 0 0 0 0 4.4110805  
 0 4.2071408 2.204 0 2.6262 0 4.50925037 1.27 0 0 0 0 0 0 0 0 0 4.13001204 0 1.62 0 4.8131520  
 4.19575068 4.13190417 0 3.190704 0 4.13001204 0 2.1405 0 3.190704 0 4.3001900 0 4.18191720  
 4.2192017 1.4 0 2.1914 0 4.15141520 4.11001904 1.37 0 2.5058 1.27 0 0 0 0 0 0 0 0 0 0 0 4.15141520  
 4.11001904 4.60030019 4.571300 3.120458 1.27 0 0 0 0 0 0 0 4.4110805 0 4.2071408 2.204 0 2.6262 0  
 4.50935037 1.27 0 0 0 0 0 0 0 0 0 0 0 4.13001204 0 1.62 0 4.8131520 4.19575068 4.13190417 0 3.190704  
 0 4.13001204 0 2.1405 0 3.190704 0 4.3001900 0 4.18191720 4.2192017 1.4 0 2.1914 0 4.2141215  
 4.8110437 0 2.5058 1.27 0 0 0 0 0 0 0 0 0 4.2141215 4.8110460 4.3001900 4.57130012 2.458 1.27 0  
 0 0 0 0 0 4.4110805 0 4.2071408 2.204 0 2.6262 0 4.50945037 1.27 0 0 0 0 0 0 0 0 0 0 4.3040214  
 4.12150811 4.4600300 4.19005758 1.27 0 0 0 0 0 0 0 4.4110805 0 4.2071408 2.204 0 2.6262 0  
 4.50955037 1.27 0 0 0 0 0 0 0 0 0 4.1170400 1.10 1.27 0 0 0 0 0 0 0 4.4111804 1.37 1.27 0 0 0 0 0  
 0 0 0 0 0 4.15170813 4.19575072 4.13210011 2.803 0 4.2071408 3.20463 0 4.79110400 2.1804 0  
 4.18041104 2.219 0 4.60008 4.13635058 1.27 1.27 1.39 0 4.64181820 4.12081306 0 3.190704 0 4.4230402  
 4.20190460 4.3241300 4.12080260 4.18021708 2.1519 0 4.5201302 4.19081413 0 2.818 0 4.111704 3.324  
 0 4.3040508 3.130403 0 3.1303 0 4.8121514 4.17190403 0 2.18 0 4.3081802 4.20181804 2.363 1.27 1.27  
 3.30405 0 4.3040508 4.13046003 4.190060 4.18191720 4.2192017 4.4575837 1.27 0 0 0 0 4.19041215  
 4.11001904 4.60020714 3.80204 0 1.62 0 4.8131520 4.19575082 4.4110402 1.19 0 4.19041215 4.11001904  
 0 4.57010018 3.80231 0 4.3041900 4.8110403 2.5837 0 2.5058 1.27 0 0 0 0 2.805 0 4.19041215  
 4.11001904 4.60020714 3.80204 0 3.131419 0 2.813 0 4.19041215 4.11001904 2.1837 1.27 0 0 0 0 0 0  
 4.15170813 4.19575072 4.13210011 2.803 0 4.19041215 4.11001904 1.63 0 4.81041920 4.17130813 1.6 0  
 2.1914 0 4.12000813 0 4.12041320 3.635058 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 1.27 1.27 0 0 0  
 4.13001204 0 1.62 0 4.8131520 4.19575068 4.13190417 0 4.13001204 0 3.51417 0 3.190704 0 4.3001900 0  
 4.18191720 4.2192017 2.437 0 2.5058 1.27 0 0 0 4.3001900 4.60181917 4.20021920 2.1704 0 1.62 0  
 4.43451300 4.12044537 0 4.13001204 1.31 0 4.45050804 4.11031845 1.37 0 4.19041215 4.11001904  
 4.18421904 4.12151100 4.19046002 4.7140802 3.44144 1.27 0 0 0 4.5081104 4.13001204 0 1.62 0  
 4.5504313 4.120444 4.63091814 2.1350 1.27 0 0 0 0 4.22081907 0 4.14150413 4.57050811 4.4130012  
 2.431 0 4.45224558 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 4.9181413 4.63032012 4.15570300  
 4.19006018 4.19172002 4.19201704 1.31 0 4.5081104 1.58 1.27 0 0 0 0 4.18021708 4.15196002  
 4.14131904 2.1319 0 1.62 0 4.6041304 4.17001904 4.60180217 4.8151918 4.57130012 2.431 0 4.19041215  
 4.11001904 4.18421904 4.12151100 4.19046002 4.7140802 3.44158 1.27 0 0 0 0 4.18002104 4.60180217  
 4.8151957 4.13001204 1.31 0 4.18021708 4.15196002 4.14131904 3.131958 1.27 0 0 0 0 4.15170813  
 4.19570550 4.67001900 0 4.18191720 4.2192017 1.4 0 4.43130012 2.444 0 4.3040508 3.130403 0  
 4.22081907 0 4.19041215 4.11001904 0 4.43190412 4.15110019 4.4600207 4.14080204 1.44 0 3.1303 0  
 4.18002104 1.3 0 2.18 0 4.43050811 4.4130012 4.4446350 1.58 1.27 1.27 3.30405 0 4.6041304 4.17001904  
 4.60180217 4.8151918 4.57130012 2.431 0 4.5080411 4.3185837 1.27 0 0 0 0 4.2141215 4.8110460  
 4.5201302 0 1.62 0 4.5500304 1.5 0 4.2141215 4.8110460 4.43130012 4.4445703 4.190058 4.37291350  
 1.27 0 0 0 0 4.2141215 4.8110460 4.5201302 0 2.6162 0 1.50 0 0 0 0 4.17041920 2.1713 0 4.45474563  
 4.9140813 4.57181917 4.57030019 4.420508 4.4110341 1.58 0 3.51417 0 4.5080411 1.3 0 2.813 0  
 4.3001900 4.58291350 1.27 0 0 0 0 1.27 0 0 0 0 4.3040214 4.12150811 4.4600520 2.1302 0 1.62 0  
 4.5500304 1.5 0 4.3040214 4.12150811 4.4604313 4.120444 4.57030019 4.583729 2.1350 1.27 0 0 0 0  
 4.3040214 4.12150811 4.4600520 2.1302 0 2.6162 0 1.50 0 0 0 0 4.5080411 2.318 0 1.62 0 1.50 0 1.61 0  
 4.18191757 4.5080411 3.31858 0 1.61 0 4.50291350 1.27 0 0 0 0 4.3040214 4.12150811 4.4600520 2.1302

0 2.6162 0 1.50 0 0 0 0 4.17041920 2.1713 0 4.3080219 4.57250815 4.57050804 4.11031831 0 4.3001900  
4.63181511 4.8195745 4.47455858 4.58291350 1.27 0 0 0 0 1.27 0 0 0 0 4.17041920 2.1713 0 4.2141215  
4.8110460 4.5201302 0 1.61 0 4.50291350 0 1.61 0 4.3040214 4.12150811 4.4600520 2.1302 1.27 1.27  
3.30405 0 4.18002104 4.60180217 4.8151957 4.13001204 1.31 0 4.2141319 4.4131958 1.37 1.27 0 0 0 0  
4.5081104 4.13001204 0 1.62 0 4.5504313 4.120444 4.60111406 4.8026315 2.2450 1.27 0 0 0 0  
4.22081907 0 4.14150413 4.57050811 4.4130012 2.431 0 4.45224558 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0  
0 0 0 0 4.5081104 4.63221708 4.19045702 4.14131904 3.131958 1.27 0 0 0 0 4.15170813 4.19570550  
4.66141215 4.8110432 4.67040214 4.12150811 1.4 0 4.11140608 1.2 0 4.18002104 1.3 0 2.1914 0  
4.43050811 4.4130012 4.4446350 1.58 1.27 0 0 0 0 1.27 3.30405 0 4.2141215 4.8110460 4.3001900  
4.57130012 3.45837 0 0 0 0 1.27 0 0 0 0 4.50505066 4.14121508 3.1104180 4.3001900 0 4.20180813 1.6 0  
1.0 0 4.3241300 4.12080200 3.111124 0 4.11140003 2.403 0 4.2141215 3.81104 0 4.5201302 4.19081413  
4.63505050 1.27 0 0 0 0 4.5081104 4.13001204 0 1.62 0 4.5504313 4.120444 4.60111406 4.8026315  
2.2450 1.27 0 0 0 0 2.805 0 3.131419 0 4.14186315 4.190763 4.4230818 4.19185705 4.8110413 4.120458  
1.37 1.27 0 0 0 0 0 0 0 4.15170813 4.19570550 4.75140608 1.2 0 4.18021708 2.1519 0 3.51417 0  
4.43130012 2.444 0 3.131419 0 4.5142013 2.363 0 4.79110400 2.1804 0 4.4131820 2.1704 0 3.190704 0  
4.3001900 0 4.18191720 4.2192017 1.4 0 2.818 0 4.3040508 3.130403 0 3.1303 0 4.2141215 4.8110403 0  
4.5081718 4.19635058 1.27 0 0 0 0 0 0 0 0 4.17041920 2.1713 1.27 1.27 0 0 0 0 4.15170813 4.19575068  
4.13190417 0 4.3001900 0 2.1914 0 4.2141215 3.81104 0 4.57051417 3.120019 0 2.18 0 4.73827877 1.31 0  
4.4630663 1.31 0 4.43450803 4.45374591 4.92934531 0 4.45130012 4.4453745 4.73140713 4.45445837  
2.5058 1.27 0 0 0 0 4.15170813 4.19575077 4.14190437 0 4.68131820 2.1704 0 4.24142017 0 4.73827877  
0 2.818 0 4.2141717 4.4021911 1.24 0 4.5141712 4.191904 1.3 0 3.51417 0 4.24142017 0 4.2141212  
3.1303 0 4.11081304 0 4.4132108 4.17141312 4.4131963 2.5058 1.27 0 0 0 0 4.3001900 4.60081315  
2.2019 0 1.62 0 4.8131520 3.195758 1.27 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 1.39 0 4.82191708 1.15 0  
3.142019 0 4.20132200 4.13190403 0 4.2070017 4.21904 2.1718 0 2.805 0 4.13040204 4.18180017 1.24 0  
4.57181504 4.2080508 1.2 0 2.1914 0 4.24142017 0 4.4132108 4.17141312 4.4131958 1.27 0 0 0 0 0 0 0  
4.3001900 4.60081315 2.2019 0 1.62 0 4.3001900 4.60081315 4.20196318 4.19170815 2.5758 1.27 0 0 0 0  
0 0 0 0 2.805 0 4.3001900 4.60081315 4.20196318 4.19001719 4.18220819 4.7575045 2.5058 0 3.1303 0  
4.3001900 4.60081315 4.20196304 4.13031822 4.8190757 4.50455058 1.37 1.27 0 0 0 0 0 0 0 0 0 0  
4.3001900 4.60081315 2.2019 0 1.62 0 4.3001900 4.60081315 4.20194291 4.37599141 0 0 1.39 0  
4.81041214 2.2104 0 4.18201717 4.14201303 3.81306 0 4.18081306 2.1104 0 4.16201419 2.418 0 3.51417  
0 4.84130823 4.32750813 2.2023 0 4.18070411 2.1118 1.27 0 0 0 0 0 0 0 4.3001900 0 1.62 0 4.9181413  
4.63111400 4.3185703 4.190060 4.8131520 2.1958 0 0 1.39 0 4.82000504 2.1124 0 4.15001718 1.4 0  
3.190704 0 4.73827877 0 4.8131520 1.19 1.27 0 0 0 0 0 0 0 4.2141215 4.8110403 4.60030019 1.0 0 1.62  
0 4.4230402 4.20190460 4.3241300 4.12080260 4.18021708 4.15195713 4.120431 0 4.5500214  
4.12150811 4.4604313 4.120444 2.5031 0 4.3001900 1.58 1.27 0 0 0 0 0 0 4.15170813 4.19575066  
4.14121508 3.110403 0 4.67001900 3.375031 0 4.2141215 4.8110403 4.60030019 2.58 1.27 0 0 0 0  
4.4230204 2.1519 0 4.9181413 4.63738278 4.77670402 4.14030468 4.17171417 1.37 1.27 0 0 0 0 0 0 0  
4.15170813 4.19575072 4.13210011 2.803 0 4.73827877 0 4.8131520 2.1963 0 4.79110400 2.1804 0  
4.2070402 1.10 0 3.190704 0 4.5141712 2.19 0 3.1303 0 3.191724 0 4.60008 4.13635058 1.27 0 0 0 0  
4.4230204 2.1519 0 4.68230204 4.15190814 1.13 0 2.18 0 2.437 1.27 0 0 0 0 0 0 0 4.15170813  
4.19575064 1.13 0 4.41717114 1.17 0 4.14020220 4.17170403 3.375031 0 2.458 1.27 1.27 1.39 0  
4.67081704 4.2191124 0 4.8132114 4.10081306 0 3.190704 0 4.12000813 0 4.12041320 0 4.5201302  
4.19081413 0 4.22070413 0 3.190704 0 4.18021708 2.1519 0 4.17201318 1.27 4.39021420 4.13190001  
4.11041857 1.58 1.27 4.39393939 4.39393939 4.39393939 4.39393939 4.39393939 1.39 1.27 1.27 3.30405  
0 4.12000813 4.90575837 1.27 0 0 0 0 4.15170813 4.19575029 4.13291386 4.4110214 3.120463 0  
4.66071414 2.1804 0 4.24142017 0 4.14150417 4.190814 2.1337 0 2.5058 1.27 0 0 0 0 1.39 0 4.84180417 0  
4.14151908 2.1413 0 2.1914 0 4.6041304 4.17001904 0 4.12001515 4.8130618 0 2.1417 0 3.131419 1.27 0  
0 0 0 4.6041304 4.17001904 4.60120015 4.15081306 1.18 0 1.62 0 4.8131520 4.19575067 1.14 0 3.241420  
0 4.22001319 0 2.1914 0 4.6041304 4.17001904 0 4.12001515 4.8130618 1.35 0 1.91 0 4.42051417 0  
4.24041841 1.31 0 1.90 0 4.42051417 0 4.13144137 0 2.5058 1.27 0 0 0 2.805 0 4.6041304 4.17001904

4.60120015 4.15081306 1.18 0 2.6262 0 4.50915037 1.27 0 0 0 0 0 0 0 0 4.75110013 4.6200006 4.4601214  
4.3041157 1.58 1.27 1.27 0 0 0 1.39 0 4.67040508 2.1304 0 4.8131520 1.19 0 3.1303 0 4.14201915  
2.2019 0 4.3081704 4.2191417 3.80418 1.27 0 0 0 0 3.81391 0 1.62 0 4.50111100 4.13062000 4.6047614  
4.3320813 4.15201918 1.50 1.27 0 0 0 0 4.14201991 0 1.62 0 4.50111100 4.13062000 4.6047614  
4.3321420 4.19050811 2.450 1.27 0 0 0 0 4.14201992 0 1.62 0 4.50111100 4.13062000 4.6047614  
4.3321420 3.190150 1.27 1.27 1.27 0 0 0 0 1.27 0 0 0 0 4.15171402 2.418 0 1.62 0 4.8131520 4.19575067  
1.14 0 3.241420 0 4.22001319 0 2.1914 0 4.15171402 3.41818 0 4.5081104 2.1835 0 1.91 0 4.42051417 0  
4.24041841 1.31 0 1.90 0 4.42051417 0 4.13144137 0 2.5058 1.27 0 0 0 0 4.8055715 4.17140204 1.18 0  
2.6262 0 4.45914558 1.37 1.27 0 0 0 0 0 0 1.39 0 4.81040003 0 4.12001515 4.8130618 0 4.14131124 0  
2.805 0 4.19070424 0 3.1704 0 4.13040204 4.18180017 1.24 1.27 0 0 0 0 0 0 0 4.12001515 4.8130618 0  
1.62 0 4.17040003 4.60120015 4.15081306 3.185758 1.27 0 0 0 0 0 0 0 1.39 0 3.700419 0 4.20180417 0  
4.8131520 1.19 0 3.51417 0 3.190704 0 4.17001306 1.4 0 2.1405 0 4.5081104 1.18 0 2.1914 0 4.15171402  
3.41818 1.27 0 0 0 0 0 0 0 0 4.18190017 1.19 0 1.62 0 4.8131597 4.8131520 4.19575068 4.13190417 0  
3.190704 0 4.18190017 1.19 0 2.1405 0 3.190704 0 4.17001306 1.4 0 2.1405 0 4.5081104 1.18 0 2.1914 0  
4.17040003 0 4.57046306 2.6331 0 1.95 0 3.51417 0 4.95631923 3.195837 0 3.505858 1.27 0 0 0 0 0 0 0  
3.41303 0 1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 3.190704 0 3.41303 0 2.1405 0 3.190704  
0 4.17001306 1.4 0 2.1405 0 4.5081104 1.18 0 2.1914 0 4.17040003 0 4.57046306 2.6331 0 2.9195 0  
3.51417 0 4.91956319 4.23195837 0 3.505858 1.27 1.27 0 0 0 0 0 0 1.39 0 4.79171402 3.41818 0  
4.5081104 1.18 0 4.22081907 2.813 0 3.190704 0 4.18150402 4.8050804 1.3 0 4.17001306 1.4 1.27 0 0 0 0  
0 0 0 0 4.15171402 4.4181860 4.5081104 4.18570813 2.9131 0 4.14201991 1.31 0 4.18190017 2.1931 0  
4.4130331 0 4.12001515 4.8130618 1.31 0 4.14201992 1.58 1.27 1.27 0 0 0 0 1.39 0 4.78151908  
4.14130011 3.112431 0 4.6041304 4.17001904 0 2.13 0 4.8120006 1.4 0 4.5171412 0 4.19042319 0  
4.8131520 1.19 1.27 0 0 0 0 4.16200418 4.19081413 1.4 0 1.62 0 4.8131520 4.19575070 4.41304173 3.1904  
0 4.19042319 0 4.8131520 1.19 0 2.1914 0 4.8120006 1.4 0 4.5081104 1.35 0 1.91 0 4.42051417 0  
4.24041841 1.31 0 1.90 0 4.42051417 0 4.13144137 0 2.5058 1.27 0 0 0 0 2.805 0 4.16200418 4.19081413  
1.4 0 2.6262 0 4.50915037 1.27 0 0 0 0 0 0 0 4.3001900 1.62 0 4.14150413 4.60050811 4.4905758 1.27  
0 0 0 0 0 0 0 2.805 0 4.3001900 1.37 1.27 0 0 0 0 0 0 0 0 0 4.15171402 4.4181860 4.8120006  
4.4600604 4.13041700 4.19081413 4.90570300 3.190058 1.27 1.27 3.30405 0 4.15171402 4.4181860  
4.8120006 4.4600604 4.13041700 4.19081413 4.90570300 4.19005837 1.27 0 0 0 4.50505070 4.4130417  
4.190418 0 2.13 0 4.8120006 1.4 0 4.5171412 0 4.15171421 4.8030403 0 4.3001900 4.63505050 1.27 0 0 0  
0 2.805 0 3.131419 0 4.3001900 1.37 1.27 0 0 0 0 0 0 0 4.15170813 4.19575077 1.14 0 4.3001900  
2.1914 0 4.15171402 3.41818 0 3.51417 0 4.8120006 1.4 0 4.6041304 4.17001908 4.14136350 1.58 1.27 0  
0 0 0 0 0 0 4.17041920 2.1713 1.27 0 0 0 0 4.15170813 4.19575067 3.1900 0 4.17040204 4.8210403 0  
3.51417 0 4.8120006 1.4 0 4.6041304 4.17001908 4.14133750 1.31 0 4.3001900 1.58 1.27 0 0 0  
4.2141114 4.17600300 2.1900 0 1.62 0 4.3001900 4.60191460 4.2141114 4.17905703 4.190058 1.27 0 0 0  
0 2.805 0 3.131419 0 4.2141114 4.17600300 3.190037 1.27 0 0 0 0 0 0 0 4.15170813 4.19575077 1.14 0  
4.2141114 1.17 0 4.3001900 4.6041304 4.17001904 4.3635058 1.27 0 0 0 0 0 0 0 4.17041920 2.1713  
1.27 0 0 0 0 4.8120006 1.4 0 1.62 0 4.2170400 4.19046008 4.12000604 4.90570214 4.11141760 4.3001900  
1.58 1.27 0 0 0 0 4.8120006 4.4631807 4.14225758 0 1.39 0 4.78151908 4.14130011 2.1124 0 4.3081815  
3.110024 0 3.190704 0 4.8120006 1.4 1.27 0 0 0 0 4.18002104 4.60141519 3.81413 0 1.62 0 4.8131520  
4.19575067 1.14 0 3.241420 0 4.22001319 0 2.1914 0 4.18002104 0 4.19070818 0 4.8120006 1.4 0 2.18 0  
1.0 0 4.79676935 0 4.57240418 4.32131458 1.37 0 2.5058 1.27 0 0 0 0 2.805 0 4.18002104 4.60141519  
4.8141363 4.11142204 3.175758 0 2.6262 0 4.45240418 2.4537 1.27 0 0 0 0 0 0 1.39 0 4.64181820  
4.12081306 0 4.45081200 3.60445 0 2.818 0 1.0 0 3.797275 0 4.72120006 1.4 0 4.14010904 2.219 0  
4.24142045 2.2104 0 4.2170400 3.190403 0 2.1417 0 4.12001308 4.15201100 3.190403 1.27 0 0 0 0 0 0  
4.5081104 4.60150019 1.7 0 1.62 0 4.8131520 4.19575068 4.13190417 0 3.190704 0 4.15001907 0 2.1914  
0 4.18002104 0 3.190704 0 3.796769 0 4.5081104 0 4.57081302 4.11200308 2.1306 0 4.5081104  
4.13001204 2.5837 0 2.5058 1.27 0 0 0 0 0 0 0 4.18002104 4.60081200 4.6046000 4.18601503 4.5905708  
4.12000604 1.31 0 4.5081104 4.60150019 2.758 1.27 0 0 0 0 0 0 0 1.27 3.30405 0 4.2170400 4.19046008  
4.12000604 4.90570214 4.11141760 4.3001900 2.5837 1.27 0 0 0 0 3.505050 0 4.66170400 2.1904 0 2.13 0



4.15170813 4.19570550 4.68171714 1.17 0 4.14150413 3.81306 0 4.5081104 1.37 0 4.43044450 1.58 1.27  
0 0 0 0 0 0 0 4.17041920 2.1713 0 2.4241 1.27 1.27 1.39 0 4.84130214 4.12120413 1.19 0 3.190704 0  
4.5141111 4.14220813 1.6 0 4.11081304 0 2.1914 0 3.172013 0 3.190704 0 4.12000813 0 4.5201302  
4.19081413 0 2.813 0 1.0 0 4.18021708 2.1519 0 4.4132108 4.17141312 3.41319 1.27 1.39 0 4.12000813  
2.5758 1.27 1.27 1.27 3.30405 0 4.75110013 4.6200006 4.4601214 4.3041157 2.5837 1.27 0 0 0 0  
4.2070017 4.21904 2.1718 0 1.62 0 4.45000102 4.3040506 4.7080910 4.11121314 4.15161718 4.19202122  
3.232425 0 4.29132919 4.29292945 4.31323334 4.35363738 4.39404142 4.43442945 0 2.4647 0 3.484950  
0 4.51525354 4.55565758 4.59606162 4.63646566 4.67686970 4.71727374 4.75767778 4.79808182  
4.83848586 4.87888990 4.91929394 4.95969798 2.9945 1.27 0 0 0 0 1.27 0 0 0 1.39 0 4.64181008 2.1306  
0 4.20180417 0 3.51417 0 4.8131520 1.19 1.27 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 4.15081313 2.403 0  
1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 3.190704 0 4.13201201 2.417 0 2.1405 0 4.2070017  
4.21904 2.1718 0 2.1914 0 3.201804 0 3.572015 0 2.1914 0 4.91909058 1.31 0 2.1417 0 3.919090 0  
3.51417 0 4.111137 0 3.505858 1.27 0 0 0 0 0 0 0 4.2070017 4.21904 2.1718 0 1.62 0 4.2070017 4.21904  
4.17184237 4.15081313 3.40341 1.27 0 0 0 0 4.4230204 2.1519 0 4.85001120 4.4681717 3.141737 1.27 0 0  
0 0 0 0 0 4.15170813 4.19575072 4.13210011 2.803 0 4.13201201 3.41731 0 4.20180813 1.6 0 3.1111 0  
4.2070017 4.21904 4.17186350 1.58 1.27 0 0 0 0 0 0 0 4.15081313 2.403 0 1.62 0 3.919090 1.27 0 0 0 0  
1.27 0 0 0 0 4.18190017 4.19601104 4.13061907 0 1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0  
3.190704 0 4.18190017 1.19 0 4.11041306 2.1907 0 3.51417 0 4.2141201 4.8130019 4.8141318 0  
4.57046306 2.6331 0 3.915837 0 3.505858 1.27 0 0 0 0 4.4130360 4.11041306 2.1907 0 1.62 0 4.8131957  
4.8131520 4.19575068 4.13190417 0 3.190704 0 3.41303 0 4.11041306 2.1907 0 3.51417 0 4.2141201  
4.8130019 4.8141318 0 4.57046306 2.6331 0 3.945837 0 3.505858 1.27 0 0 0 0 4.2170400 4.19046002  
4.20181914 4.12600017 3.170024 0 1.62 0 4.8131520 4.19575067 1.14 0 3.241420 0 4.22001319 0 2.1914  
0 4.2170400 2.1904 0 1.0 0 4.2201819 2.1412 0 4.171700 2.2435 0 4.57240418 4.32131458 1.37 0  
4.50586311 4.14220417 2.5758 0 2.6262 0 4.45240418 1.45 1.27 0 0 0 0 1.27 0 0 0 0 2.805 0 4.2170400  
4.19046002 4.20181914 4.12600017 4.17002437 1.27 0 0 0 0 0 0 0 4.2201819 4.14126000 4.17170024 0  
1.62 0 2.4241 1.27 0 0 0 0 0 0 0 4.15170813 4.19575068 4.13190417 0 3.190704 0 4.8130308 3.20418 0  
3.51417 0 4.2414202017 0 4.2201819 2.1412 0 4.2070017 4.21904 1.17 0 4.171700 4.24375058 1.27 0 0 0 0 0 0  
0 0 0 3.51417 0 1.60 0 2.813 0 4.17001306 4.4571508 4.13130403 2.5837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.8130304 1.23 0 1.62 0 4.8131957 4.8131520 4.19570550 4.68131904 1.17 0 4.8130304 1.23 0 2.4390 0  
1.61 0 4.91443243 4.15081313 3.40344 0 4.57905908 4.13030423 4.4035837 0 3.505858 1.27 0 0 0 0 0 0 0  
0 0 0 0 0 4.2201819 4.14126000 4.17170024 4.63001515 4.4130357 4.2070017 4.21904 4.17184208  
4.13030423 2.4158 1.27 0 0 0 0 0 0 0 4.2070017 4.21904 2.1718 0 1.62 0 4.2201819 4.14126000  
4.17170024 1.27 1.27 0 0 0 0 1.39 0 4.68131820 2.1704 0 3.190704 0 4.14201915 2.2019 0 4.3081704  
4.2191417 1.24 0 4.4230818 2.1918 1.27 0 0 0 0 4.14201915 4.20196003 4.8170402 4.19141724 0 1.62 0  
4.50111100 4.13062000 4.6047614 4.3321200 4.15150813 3.61850 1.27 0 0 0 0 4.14186312 4.100403  
4.8171857 4.14201915 4.20196003 4.8170402 4.19141724 1.31 0 4.4230818 4.19601410 4.62831720  
2.458 1.27 1.27 0 0 0 0 3.51417 0 4.11041306 2.1907 0 2.813 0 4.17001306 4.4571819 4.171960  
4.11041306 3.190731 0 4.4130360 4.11041306 2.1907 0 1.61 0 3.915837 1.27 0 0 0 0 0 0 0 4.22081907 0  
4.14150413 4.57055043 4.14201915 4.20196003 4.8170402 4.19141724 4.44324311 4.4130619 4.7446319  
4.23195031 0 4.50225058 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 0 4.15170813 4.19570550  
4.70041304 4.17001908 2.1306 0 4.43110413 4.6190744 4.59020700 4.17000219 2.417 0 4.2141201  
4.8130019 4.8141318 4.63636350 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 4.13201260 4.2141201 4.8130019  
4.8141318 0 1.62 0 4.12001907 4.63151422 4.57110413 4.57020700 4.17000219 4.4171858 1.31 0  
4.11041306 3.190758 1.27 0 0 0 0 0 0 0 0 0 0 3.51417 0 1.8 0 2.813 0 4.17001306 4.4570813  
4.19571320 4.12600214 4.12010813 4.190814 4.13185858 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.2141201 4.8130019 3.81413 0 1.62 0 2.5050 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.19041215 0 1.62 0 1.8  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 3.51417 0 1.9 0 2.813 0 4.17001306 4.4571104 4.13061907 2.5837 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 4.2141201 4.8130019 3.81413 0 1.62 0 4.2070017 4.21904 4.17184219  
3.41215 0 1.53 0 4.11041357 4.2070017 4.21904 4.17185841 0 1.61 0 4.2141201 4.8130019 3.81413 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 4.19041215 0 3.323262 0 4.11041357 4.2070017 4.21904 3.171858 1.27 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.5081104 4.63221708 4.19045705 4.50430844 0 4.43021412 4.1081300  
4.19081413 4.44291350 1.58 1.27 1.27 0 0 0 0 0 4.15170813 4.19575066 4.14120108 4.13001908 3.1413180  
4.6041304 4.17001904 1.3 0 4.18200202 4.4181805 4.20111124 3.635058 1.27 1.27 1.39 0 4.68230012  
3.151104 0 2.1405 0 3.71422 0 2.1914 0 4.2001111 0 3.190704 0 4.5201302 4.19081413 0 4.57201302  
4.14121204 2.1319 0 3.190704 0 4.5141111 4.14220813 1.6 0 4.11081304 0 2.1914 0 3.201804 0 2.8190  
2.813 0 2.13 0 4.21920 2.11 0 4.18021708 3.151958 1.27 1.39 0 4.75110013 4.6200006 4.4601214  
4.3041157 1.58 1.27 1.27 1.27 3.30405 0 4.17040003 4.60120015 4.15081306 4.18575837 1.27 0 0 0 0  
4.50505081 4.4000318 0 4.12001515 4.8130618 0 4.5171412 0 4.5081104 1.18 0 4.13001204 1.3 0  
4.91631923 1.19 0 2.1914 0 4.94631923 1.19 0 3.1303 0 4.18191417 2.418 0 4.19070412 0 2.813 0 1.0 0  
4.11081819 0 2.1405 0 4.3080219 4.8141300 4.17080418 4.63505050 1.27 0 0 0 0 0 4.12001515 4.8130618 0  
1.62 0 3.424344 0 3.51417 0 1.60 0 2.813 0 4.17001306 4.4579458 1.41 1.27 0 0 0 0 3.51417 0 1.8 0 2.813 0  
4.17001306 4.4579131 0 3.955837 1.27 0 0 0 0 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.22081907 0  
4.14150413 4.57055011 4.11001306 4.20000604 4.76140332 4.12001515 4.8130618 4.32430844  
4.63192319 2.5031 0 4.50175058 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.51417 0  
4.11081304 0 2.813 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.15001719 1.18 0 1.62 0  
4.11081304 4.63181917 4.8155758 4.63181511 4.8195758 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2.805  
0 4.11041357 4.15001719 2.1858 0 2.6262 0 2.9237 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.8036018 3.191731 0 4.21001120 4.4601819 1.17 0 1.62 0 4.15001719 1.18 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 4.12001515 4.8130618 4.42085991 4.41422100 4.11200460 4.18191741 0 1.62 0  
4.8036018 2.1917 1.27 0 0 0 0 0 0 0 4.4230204 2.1519 0 4.69081104 4.77141969 4.14201303 4.68171714  
2.1737 1.27 0 0 0 0 0 0 0 0 0 0 4.15170813 4.19570550 4.68171714 1.17 0 4.14150413 3.81306 0  
4.5081104 0 4.12001515 4.8130618 4.32430844 4.63192319 2.5058 1.27 0 0 0 0 4.17041920 2.1713 0  
4.12001515 4.8130618 1.27 1.27 3.30405 0 4.15171402 4.4181860 4.5081104 4.18570813 4.15201960  
4.3081704 4.2191417 2.2431 0 4.14201915 4.20196003 4.8170402 4.19141724 1.31 0 4.18190017 2.1931 0  
4.4130331 0 4.12001515 4.8130618 1.31 0 4.14201903 4.8179258 1.37 1.27 0 0 0 0 4.50505079  
4.17140204 4.18180418 0 4.5081104 1.18 0 4.22081907 2.813 0 1.0 0 4.6082104 1.13 0 4.17001306 1.4 0  
4.20180813 1.6 0 4.15170403 4.4050813 2.403 0 4.12001515 4.8130618 0 3.1303 0 4.6041304 4.17001904  
1.18 0 4.14201915 2.2019 0 4.5081104 4.18635050 1.50 1.27 0 0 0 0 4.12002360 4.5081104 4.60132012  
3.10417 0 1.62 0 4.12002357 4.42081319 4.57141863 4.15001907 4.63181511 4.8190423 4.19571418  
4.63150019 4.7630100 4.18041300 4.12045705 4.58584290 2.4158 0 3.51417 0 1.5 0 2.813 0 4.6111401  
4.63061114 4.1570550 4.43142019 4.15201960 4.3081704 4.2191417 4.24443256 4.63192319 4.50584131  
0 4.3040500 4.20111962 2.9058 1.27 0 0 0 0 3.51417 0 1.8 0 2.813 0 4.17001306 4.4571819 4.171931 0  
3.41303 0 1.61 0 3.915837 1.27 0 0 0 0 0 0 4.8131520 4.19600508 4.11046015 3.1907 0 1.62 0  
4.5504308 4.13152019 4.60030817 4.4021914 4.17244432 4.43084463 4.19231950 1.27 0 0 0 0 0 0 0 0  
4.19172437 1.27 0 0 0 0 0 0 0 0 0 4.15171402 4.4181860 4.11001706 4.4600508 4.11045708  
4.13152019 4.60050811 4.4601500 3.190731 0 4.14201915 4.20196003 4.8170402 4.19141724 1.31 0  
4.12001515 4.8130618 1.31 0 4.12002360 4.5081104 4.60132012 4.1041758 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.15171402 4.4181860 4.11001706 4.4600508 4.11049257 4.8131520 4.19600508 4.11046015 4.190731 0  
4.14201903 4.8179231 0 4.12001515 4.8130618 1.31 0 4.12002360 4.5081104 4.60132012 4.1041758 1.27  
0 0 0 0 0 0 0 0 0 4.12002360 4.5081104 4.60132012 3.10417 0 2.6162 0 1.91 1.27 0 0 0 0 0 0 0 0  
4.4230204 2.1519 0 4.69081104 4.77141969 4.14201303 4.68171714 2.1737 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.15170813 4.19570550 4.66142011 1.3 0 3.131419 0 4.14150413 0 3.190704 0 4.5081104 0 4.43081315  
4.20196005 4.8110460 4.15001907 3.445058 1.27 1.27 3.30405 0 4.15171402 4.4181860 4.11001706  
4.4600508 4.11045708 4.13152019 4.60050811 4.4601500 3.190731 0 4.14201915 4.20196003 4.8170402  
4.19141724 1.31 0 4.12001515 4.8130618 1.31 0 4.5081104 4.60132012 4.1041758 1.37 1.27 0 0 0 0  
4.50505068 4.5050802 4.8041319 2.1124 0 4.15171402 4.4181804 1.18 0 1.0 0 4.11001706 1.4 0  
4.5081104 0 4.20180813 1.6 0 4.12001515 4.8130618 0 3.1303 0 4.18002104 1.18 0 2.1914 0 2.13 0  
4.14201915 2.2019 0 4.5081104 4.63505050 1.27 0 0 0 0 4.14201915 4.20196005 4.8110460 4.15001907 0  
1.62 0 4.5504314 4.20191520 4.19600308 4.17040219 4.14172444 4.32430508 4.11046013 4.20120104  
1.17 0 1.61 0 4.91446319 3.231950 1.27 0 0 0 0 4.22081907 0 4.14150413 4.57081315 4.20196005

4.8110460 4.15001907 1.31 0 4.50175058 0 2.18 0 4.8131520 4.19600508 3.110431 0 4.14150413  
4.57142019 4.15201960 4.5081104 4.60150019 2.731 0 4.50225058 0 2.18 0 4.14201915 4.20196005  
4.8110437 1.27 0 0 0 0 0 0 0 3.51417 0 4.11081304 0 2.813 0 4.8131520 4.19600508 3.110437 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 4.14201915 4.20196002 4.14131904 2.1319 0 1.62 0 4.2141321 4.4171960 4.2141319  
4.4131957 4.11081304 1.31 0 4.12001515 4.8130618 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.14201915  
4.20196005 4.8110463 4.22170819 4.4571420 4.19152019 4.60021413 4.19041319 1.58 1.27 1.27 3.30405  
0 4.15171402 4.4181860 4.11001706 4.4600508 4.11049257 4.8131520 4.19600508 4.11046015 4.190731  
0 4.14201915 4.20196003 4.8170402 4.19141724 1.31 0 4.12001515 4.8130618 1.31 0 4.5081104  
4.60132012 4.1041758 1.37 1.27 0 0 0 0 4.50505068 4.5050802 4.8041319 2.1124 0 4.15171402 4.4181804  
1.18 0 1.0 0 4.11001706 1.4 0 4.5081104 0 4.20180813 1.6 0 4.12001515 4.8130618 0 3.1303 0 4.18002104  
1.18 0 2.1914 0 2.13 0 4.14201915 2.2019 0 4.5081104 4.63505050 1.27 0 0 0 0 4.14201915 4.20196005  
4.8110460 4.15001907 0 1.62 0 4.5504314 4.20191520 4.19600308 4.17040219 4.14172444 4.32430508  
4.11046013 4.20120104 1.17 0 1.61 0 4.91446319 3.231950 1.27 0 0 0 0 4.22081907 0 4.14150413  
4.57081315 4.20196005 4.8110460 4.15001907 1.31 0 4.50175058 0 2.18 0 4.8131520 4.19600508  
3.110431 0 4.14150413 4.57142019 4.15201960 4.5081104 4.60150019 2.731 0 4.50225058 0 2.18 0  
4.14201915 4.20196005 4.8110437 1.27 0 0 0 0 0 0 0 0 3.51417 0 4.11081304 0 2.813 0 4.8131520  
4.19600508 3.110437 1.27 0 0 0 0 0 0 0 0 0 0 0 4.14201915 4.20196001 0 1.62 0 4.2141321 4.4171960  
4.2141319 4.4131992 4.57110813 2.431 0 4.12001515 4.8130618 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.14201915 4.20196005 4.8110463 4.22170819 4.4571420 4.19152019 3.600158 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 1.27 1.27 3.30405 0 4.2141321 4.4171960 4.2141319 4.4131957 4.2141319 4.4131931 0 4.12001515  
4.8130618 2.5837 1.27 0 0 0 4.50505066 4.14132104 3.171918 0 4.2141319 3.41319 0 4.20180813 1.6 0  
4.12001515 4.8130618 4.63505050 1.27 0 0 0 0 4.14201915 4.20196002 4.14131904 2.1319 0 1.62 0  
2.5050 1.27 0 0 0 4.8130304 1.23 0 1.62 0 1.90 1.27 0 0 0 0 4.22070811 1.4 0 4.8130304 1.23 0 1.33 0  
4.11041357 4.2141319 4.4131958 1.37 1.27 0 0 0 0 0 0 0 0 2.805 0 4.2141319 4.4131942 4.8130304 2.2341  
0 2.6262 0 4.45291345 1.37 1.27 0 0 0 0 0 0 0 0 0 0 4.14201915 4.20196002 4.14131904 2.1319 0  
2.6162 0 4.50916392 1.97 0 1.50 1.27 0 0 0 0 0 0 0 0 0 4.8130304 1.23 0 2.6162 0 1.91 1.27 0 0 0 0 0  
0 0 0 0 0 0 4.2141319 4.8132004 1.27 0 0 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 0 0 4.5142013 1.3 0 1.62 0 4.69001118  
1.4 1.27 0 0 0 0 0 0 0 3.51417 0 4.11041306 2.1907 0 2.813 0 4.17001306 4.4579431 0 2.9031 0  
4.59915837 1.27 0 0 0 0 0 0 0 0 0 0 0 2.805 0 4.8130304 1.23 0 1.61 0 4.11041306 2.1907 0 2.3362 0  
4.11041357 4.2141319 4.4131958 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.18200118 2.1917 0 1.62 0  
4.2141319 4.4131942 4.8130304 4.23370813 3.30423 0 1.61 0 4.11041306 3.190741 1.27 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 2.805 0 4.18200118 2.1917 0 2.813 0 4.12001515 4.8130618 4.42110413 3.61907 0 1.59 0  
3.914137 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.14201915 4.20196002 4.14131904 2.1319 0 2.6162 0  
4.5504311 4.4130619 4.7446343 4.12001515 4.8130618 4.42110413 3.61907 0 1.59 0 4.91414218  
4.20011819 3.174144 0 1.50 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.8130304 1.23 0 2.6162 0  
4.11041306 2.1907 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.5142013 1.3 0 1.62 0 4.83172004 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.1170400 1.10 1.27 0 0 0 0 0 0 0 2.805 0 3.131419 0 4.5142013 2.337  
1.27 0 0 0 0 0 0 0 0 0 0 0 4.14201915 4.20196002 4.14131904 2.1319 0 2.6162 0 2.5090 0 1.50 1.27 0 0 0  
0 0 0 0 0 0 0 0 4.8130304 1.23 0 2.6162 0 1.91 1.27 0 0 0 0 4.17041920 2.1713 0 4.14201915 4.20196002  
4.14131904 2.1319 1.27 1.27 3.30405 0 4.2141321 4.4171960 4.2141319 4.4131992 4.57021413  
4.19041319 1.31 0 4.12001515 4.8130618 2.5837 1.27 0 0 0 0 4.50505066 4.14132104 3.171918 0  
4.2141319 3.41319 0 4.20180813 1.6 0 4.12001515 4.8130618 4.63505050 1.27 0 0 0 0 4.14201915  
4.20196001 0 1.62 0 2.5050 1.27 0 0 0 0 4.8130304 1.23 0 1.62 0 1.90 1.27 0 0 0 0 4.22070811 1.4 0  
4.8130304 1.23 0 1.33 0 4.11041357 4.2141319 4.4131958 1.37 1.27 0 0 0 0 0 0 2.805 0 4.2141319  
4.4131942 4.8130304 2.2341 0 2.6262 0 4.45291345 1.37 1.27 0 0 0 0 0 0 0 0 0 4.14201915  
4.20196001 0 2.6162 0 3.509297 0 1.50 1.27 0 0 0 0 0 0 0 0 0 4.8130304 1.23 0 2.6162 0 1.91 1.27 0 0  
0 0 0 0 0 0 0 0 4.2141319 4.8132004 1.27 0 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 4.5142013 1.3 0 1.62 0  
4.69001118 1.4 1.27 0 0 0 0 0 0 0 3.51417 0 4.11041306 2.1907 0 2.813 0 4.17001306 4.4579431 0  
2.9031 0 4.59915837 1.27 0 0 0 0 0 0 0 0 0 0 2.805 0 4.8130304 1.23 0 1.61 0 4.11041306 2.1907 0  
2.3362 0 4.11041357 4.2141319 4.4131958 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.18200118 2.1917 0



4.6041763 2.5058 1.27 0 0 0 0 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.17041620 4.4181960 4.20180417  
4.60081315 4.20195715 4.17141215 2.1931 0 4.8131520 4.19601924 3.150458 1.27 0 0 0 0 0 0 0  
4.17041920 2.1713 0 4.21001120 1.4 1.27 0 0 0 0 4.4230204 2.1519 0 4.85001120 4.4681717 3.141737  
1.27 0 0 0 0 0 0 0 4.15170813 4.19570550 4.72132100 3.110803 0 4.8131520 2.1963 0 4.79110400  
2.1804 0 4.4131904 1.17 0 1.0 0 4.21001108 1.3 0 4.43081315 4.20196019 4.24150463 4.60601300  
4.12046060 4.44635058 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.17041620 4.4181960 4.20180417  
4.60081315 4.20195715 4.17141215 2.1931 0 4.8131520 4.19601924 3.150458 1.27 1.27 3.30405 0  
4.6041304 4.17001904 4.60021100 4.18180802 4.116002 4.8170220 4.8196005 4.17141260 4.11081304  
4.57110813 2.431 0 4.5081104 4.13001204 2.5837 1.27 0 0 0 0 1.39 0 4.78131124 0 4.19001004 0  
3.190704 0 4.15001719 0 2.1405 0 3.190704 0 4.11081304 0 4.51904 1.17 0 3.190704 0 4.2141114 1.13 0  
2.805 0 2.819 0 4.4230818 2.1918 1.27 0 0 0 0 4.6001904 1.18 0 1.62 0 4.11081304 4.63181511 4.8195745  
4.37455842 4.59914163 4.18191708 4.15575863 4.18151108 4.19574531 0 2.4558 1.27 0 0 0 0 1.3 0 1.62 0  
4.18020704 4.12031700 4.22636717 4.220813 3.65758 1.27 0 0 0 0 4.6001904 1.18 0 1.62 0 4.11081304  
4.63181917 4.8155758 4.63181511 4.8195745 1.31 0 2.4558 1.27 0 0 0 0 4.11001819 4.60041104  
4.12041319 0 1.62 0 4.77141304 1.27 0 0 0 0 4.11001819 4.60142019 3.152019 0 1.62 0 4.77141304 1.27  
1.27 0 0 0 0 3.514170 2.831 0 4.6001904 0 2.813 0 4.4132012 4.4170019 4.4570600 4.19041858 1.37 1.27  
0 0 0 0 0 0 0 4.6001904 0 1.62 0 4.6001904 4.63181917 4.8155758 1.27 0 0 0 0 0 0 0 0 2.805 0 4.6001904  
0 2.6262 0 4.45647767 2.4537 1.27 0 0 0 0 0 0 0 0 0 0 4.4110412 3.41319 0 1.62 0 4.11140608  
4.2636413 4.3570813 4.15201918 3.629231 0 4.11000104 4.11620545 4.64776743 4.8444558 1.27 0 0 0 0  
0 0 0 0 4.4110805 0 4.6001904 0 2.6262 0 4.45788145 1.37 1.27 0 0 0 0 0 0 0 0 0 0 4.4110412 3.41319 0  
1.62 0 4.11140608 4.2637817 4.57081315 4.20191862 2.9231 0 4.11000104 4.11620545 4.78814308  
3.444558 1.27 0 0 0 0 0 0 0 0 0 4.4110805 0 4.6001904 0 2.6262 0 4.45777883 2.4537 1.27 0 0 0 0 0 0 0 0 0  
0 0 4.4110412 3.41319 0 1.62 0 4.11140608 4.2637714 4.19571100 4.1041162 4.5457778 4.83430844  
2.4558 1.27 0 0 0 0 0 0 0 0 4.4110805 0 4.6001904 0 2.6262 0 4.45776477 3.674537 1.27 0 0 0 0 0 0 0 0  
0 0 4.4110412 3.41319 0 1.62 0 4.11140608 4.2637700 4.13035708 4.13152019 4.18629231 0 4.11000104  
4.11620545 4.77647767 4.43084445 1.58 1.27 0 0 0 0 0 0 0 0 4.4110405 0 4.6001904 0 2.6262 0  
4.45777881 2.4537 1.27 0 0 0 0 0 0 0 0 0 4.4110412 3.41319 0 1.62 0 4.11140608 4.2637714  
4.17570813 4.15201918 3.629231 0 4.11000104 4.11620545 4.77788143 4.8444558 1.27 0 0 0 0 0 0 0  
4.4110805 0 4.6001904 0 2.6262 0 4.45877881 2.4537 1.27 0 0 0 0 0 0 0 0 0 0 4.4110412 3.41319 0 1.62  
0 4.11140608 4.2638714 4.17570813 4.15201918 3.629231 0 4.11000104 4.11620545 4.87788143  
4.8444558 1.27 0 0 0 0 0 0 0 4.4110805 0 4.6001904 0 2.6262 0 4.45877778 3.814537 1.27 0 0 0 0 0 0 0  
0 0 0 0 4.4110412 3.41319 0 1.62 0 4.11140608 4.2638713 4.14175708 4.13152019 4.18629231 0  
4.11000104 4.11620545 4.87777881 4.43084445 1.58 1.27 0 0 0 0 0 0 0 0 4.4111804 1.37 1.27 0 0 0 0 0 0  
0 0 0 0 4.15170813 4.19570550 4.84131704 4.2140613 4.8250403 0 4.6001904 1.37 0 4.43060019  
4.4445058 1.27 0 0 0 0 0 0 0 0 0 4.2141319 4.8132004 1.27 1.27 0 0 0 0 0 0 0 1.39 0 3.640303 0  
3.190704 0 4.4110412 3.41319 0 2.1914 0 3.190704 0 4.3170022 3.81306 0 4.22081907 0 3.190704 0  
4.2141717 3.40219 0 4.15141808 4.19081413 1.27 0 0 0 0 0 0 0 2.805 0 4.11001819 4.60041104  
4.12041319 0 2.818 0 3.131419 0 4.77141304 1.37 1.27 0 0 0 0 0 0 0 0 0 0 4.4110412 4.4131963  
4.195711 4.181960 4.14201915 3.201958 1.27 0 0 0 0 0 0 0 0 0 0 2.3903 0 2.6162 0 4.4110412  
4.4131963 4.195711 4.181960 4.4110412 4.4131963 4.130207 4.14171842 4.45142019 3.454158 1.27 0 0 0  
0 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 0 4.4111804 1.37 1.27 0 0 0 0 0 0 0 0 0 1.3 0 2.6162 0 4.4110412  
3.41319 1.27 1.27 0 0 0 0 0 0 0 0 4.39110018 4.19601420 4.19152019 0 1.62 0 4.4110412 4.4131963  
4.130207 4.14171842 4.45142019 2.4541 1.27 1.27 0 0 0 0 0 0 0 1.39 0 4.84150300 2.1904 0 4.11001819  
4.60041104 4.12041319 0 2.1914 0 3.190704 0 4.2201717 3.41319 0 3.141304 1.27 0 0 0 0 0 0 0  
4.39393911 4.181960 4.4110412 3.41319 0 1.62 0 4.3630411 4.4120413 4.19184259 2.9141 0 0 1.39 0  
3.700419 0 3.190704 0 4.17040504 4.17041302 1.4 0 2.1914 0 3.190704 0 4.21920 2.11 0 4.30304 1.3 0  
4.4110412 3.41319 1.27 0 0 0 0 0 0 0 4.11001819 4.60142019 3.152019 0 1.62 0 4.3630411 4.4120413  
4.19184259 4.91416300 4.13020714 4.17184245 4.14201945 1.41 1.27 1.27 0 0 0 4.3631800 4.21045705  
4.8110413 4.120458 0 0 0 1.27 0 0 0 0 4.17041920 2.1713 0 1.3 1.27 1.27 1.27 1.27 1.39 0 4.69201302  
4.19081413 0 2.1914 0 4.15171412 2.1519 0 3.190704 0 4.20180417 0 3.51417 0 3.190704 0 4.13201201

2.417 0 2.1405 0 4.2041111 1.18 0 4.57060019 1.4 0 4.11002404 3.171858 1.27 3.30405 0 4.6041960  
 4.20180417 4.60081315 4.20195715 4.17141215 2.1931 0 4.8131520 4.19601924 4.15046208 4.13195837  
 1.27 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 4.21001120 1.4 0 1.62 0 4.8131520 4.19601924 4.15045708  
 4.13152019 4.57151714 4.12151958 1.58 1.27 0 0 0 0 0 0 0 2.805 0 4.8131520 4.19601924 2.1504 0  
 2.818 0 3.81319 0 3.1303 0 4.21001120 1.4 0 2.3362 0 2.9037 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.15170813  
 4.19575079 4.11040018 1.4 0 4.4131904 1.17 0 1.0 0 4.15141808 4.19082104 0 4.8131904 4.6041763  
 2.5058 1.27 0 0 0 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.6041960 4.20180417 4.60081315 4.20195715  
 4.17141215 2.1931 0 4.8131520 4.19601924 3.150458 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 0  
 4.21001120 1.4 1.27 0 0 0 0 4.4230204 2.1519 0 4.85001120 4.4681717 3.141737 1.27 0 0 0 0 0 0 0  
 4.15170813 4.19570550 4.72132100 3.110803 0 4.8131520 2.1963 0 4.79110400 2.1804 0 4.4131904 1.17  
 0 1.0 0 4.21001108 1.3 0 4.43081315 4.20196019 4.24150463 4.60601300 4.12046060 4.44635058 1.27 0 0  
 0 0 0 0 0 0 4.17041920 2.1713 0 4.6041960 4.20180417 4.60081315 4.20195715 4.17141215 2.1931 0  
 4.8131520 4.19601924 3.150458 1.27 1.27 3.30405 0 4.6041304 4.17001904 4.60020817 4.2200819  
 4.60051714 4.12601108 4.13045711 4.8130431 0 4.13201260 4.16200108 4.19185837 1.27 0 0 0 0 2.1602  
 0 1.62 0 4.80200013 4.19201266 4.8170220 4.8195713 4.20126016 4.20010819 2.1858 1.27 0 0 0 0  
 4.2141212 4.130318 0 1.62 0 4.11081304 4.63181917 4.8155758 4.63181511 4.8195745 0 4.45584292  
 2.3741 0 0 1.39 0 4.82100815 0 3.190704 0 4.45660817 4.2200819 0 3.873745 0 4.15001719 1.27 0 0 0 0  
 3.51417 0 3.21203 0 2.813 0 4.2141212 4.130318 1.37 1.27 0 0 0 0 0 0 0 4.6001904 1.31 0 4.170618 0  
 1.62 0 4.2120363 4.18151108 4.19574557 2.4558 1.27 0 0 0 0 0 0 0 4.170618 0 1.62 0 4.170618  
 4.63181917 4.8155745 4.58455863 4.18151108 4.19574531 2.4558 1.27 1.27 0 0 0 0 0 0 0 2.805 0  
 4.6001904 0 2.813 0 4.42452345 1.31 0 4.45244531 0 4.45254531 0 4.45074531 0 4.45184531 0  
 4.45180306 2.4531 0 4.45194531 0 4.45190306 3.454137 1.27 0 0 0 0 0 0 0 0 0 0 4.6041900  
 4.19191757 3.160231 0 4.6001904 4.58570813 4.19570017 4.6184290 3.415858 1.27 0 0 0 0 0 0 0  
 4.4110805 0 4.6001904 0 2.813 0 4.42451723 2.4531 0 4.45172445 1.31 0 4.45172545 2.4137 1.27 0 0 0  
 0 0 0 0 0 0 0 2.805 0 4.45150832 2.9245 0 2.813 0 4.170618 4.42904137 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
 0 4.130611 1.4 0 1.62 0 4.13156315 1.8 0 1.32 0 1.92 1.27 0 0 0 0 0 0 0 0 0 4.4111804 1.37 1.27 0 0 0  
 0 0 0 0 0 0 0 0 0 0 0 0 4.130611 1.4 0 1.62 0 4.5111400 4.19570017 4.6184290 2.4158 0 0 1.39 0  
 4.64181820 4.12081306 0 4.14190704 1.17 0 4.130611 2.418 0 3.1704 0 4.3081704 4.2191124 0  
 4.18150402 4.8050804 1.3 1.27 0 0 0 0 0 0 0 0 0 0 4.6041900 4.19191757 3.160231 0 4.6001904  
 4.58570013 4.6110431 0 4.8131957 4.170618 4.42914158 1.58 1.27 0 0 0 0 0 0 0 4.4110805 0 4.6001904  
 0 2.6262 0 4.45022345 1.37 1.27 0 0 0 0 0 0 0 0 0 4.6041900 4.19191757 3.160231 0 4.6001904  
 4.58570813 4.19570017 4.6184290 3.415831 0 4.8131957 4.170618 4.42914158 1.58 1.27 0 0 0  
 4.17041920 2.1713 0 2.1602 1.27 1.27 3.30405 0 4.6041304 4.17001904 4.60021215 3.575837 1.27 0 0 0  
 1.39 0 4.66170400 2.1904 0 3.190704 0 4.45081318 4.19172002 4.19081413 4.82041963 4.19231945 0  
 4.5081104 1.27 0 0 0 0 4.13001204 2.6012 0 1.62 0 4.8131520 4.19575068 4.13190417 0 3.190704 0  
 4.13001204 0 3.51417 0 4.24142017 0 4.2141212 3.1303 0 4.12001917 3.82337 0 2.5058 1.27 0 0 0 0  
 4.22081907 0 4.14150413 4.57130012 4.4601231 0 4.45224558 0 2.18 0 2.537 1.27 0 0 0 0 0 0 1.39 0  
 4.75081304 1.18 0 1.91 0 2.1914 0 4.91939190 2.9737 0 4.18191708 4.13066021 3.1739 0 1.62 0  
 4.8131520 4.19574568 4.13190417 0 4.18191708 4.13066021 3.1737 0 2.4558 1.27 0 0 0 0 0 0 0 3.51417  
 0 1.8 0 2.813 0 4.17001306 4.4579193 4.91909758 1.37 1.27 0 0 0 0 0 0 0 0 0 4.5632217 4.8190457  
 4.5501819 4.17081306 4.60210017 3.430844 0 1.62 0 4.8131520 4.19574568 4.13190417 0 4.18191708  
 4.13066021 3.1737 0 4.45582913 2.5058 1.27 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 1.39 0 4.75081304 1.18 0  
 4.91939190 1.98 0 2.1914 0 4.92969291 2.9437 0 4.21001760 4.6082104 2.1339 0 1.62 0 4.8131520  
 4.19574568 4.13190417 0 3.190704 0 4.21001708 4.11104 0 4.13001204 0 2.1914 0 2.104 0 4.20180403  
 1.37 0 2.4558 1.27 0 0 0 0 0 0 3.51417 0 1.8 0 2.813 0 4.17001306 4.4579193 4.91909758 1.37 1.27 0 0  
 0 0 0 0 0 0 0 0 4.5632217 4.8190457 4.5502100 4.17600608 4.21041343 2.844 0 1.62 0 4.8131520  
 4.19574568 4.13190417 0 3.190704 0 4.21001708 4.11104 0 4.13001204 0 2.1914 0 2.104 0 4.20180403  
 1.37 0 4.45582913 2.5058 1.27 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 1.39 0 4.75081304 1.18 0 4.92969291  
 1.95 0 2.1914 0 4.93999392 2.9137 0 4.21001760 4.6111401 1.39 0 1.62 0 4.6111401 4.111857 4.58422100  
 4.17600608 4.21041339 1.41 1.27 0 0 0 0 0 0 0 3.51417 0 1.8 0 2.813 0 4.17001306 4.4579193

4.91909758 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.5632217 4.8190457 4.5502100 4.17600611 4.14014308 1.44  
0 1.62 0 4.6111401 4.111857 4.58630604 4.19572100 4.17600608 4.21041343 3.84431 0 4.45850017  
4.8000111 1.4 0 3.131419 0 4.5142013 3.34558 0 0 1.39 0 4.68171714 1.17 0 4.7001303 4.11081306 1.37 0  
4.45850017 4.8000111 1.4 0 3.131419 0 4.5142013 4.3452913 2.5058 1.27 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0  
0 1.39 0 4.75081304 1.18 0 4.93999392 1.92 0 2.1914 0 4.95929492 2.9837 0 4.4230402 4.57210017  
4.60061114 3.13958 1.27 0 0 0 0 0 0 0 3.51417 0 1.8 0 2.813 0 4.17001306 4.4579193 4.91909758 1.37  
1.27 0 0 0 0 0 0 0 0 0 0 4.5632217 4.8190457 4.5500805 0 4.8180813 4.18190013 4.2045721 4.176006  
4.11140143 3.84431 0 4.18191758 1.37 0 4.4230402 4.57210017 4.60061114 4.1430844 1.58 0 0 1.39 0  
4.68171714 1.17 0 4.7001303 4.11081306 1.37 0 4.68230402 3.201904 0 4.14131124 0 2.805 0 4.8194518  
0 1.0 0 4.18191708 4.13062913 2.5058 1.27 0 0 0 0 0 0 1.27 0 0 0 0 0 0 1.39 0 4.75081304 1.18 0  
4.95929492 1.99 0 2.1914 0 4.96959593 2.9537 0 4.4210011 4.57210017 4.60061114 3.13958 1.27 0 0 0 0  
0 0 0 3.51417 0 1.8 0 2.813 0 4.17001306 4.4579193 4.91909758 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
4.5632217 4.8190457 4.5500805 0 4.8180813 4.18190013 4.2045721 4.176006 4.11140143 3.84431 0  
4.18191758 1.37 0 4.17041820 2.1119 0 1.62 0 4.4210011 4.57210017 4.60061114 4.1430844 1.58 0 0 1.39 0  
0 4.68171714 1.17 0 4.7001303 4.11081306 1.37 0 4.68210011 4.20001904 0 4.14131124 0 2.805 0  
4.8194518 0 1.0 0 4.18191708 4.13062913 2.5058 1.27 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 1.39 0  
4.75081304 0 4.96959593 2.9637 0 4.71241504 4.17110813 1.10 1.27 0 0 0 0 0 0 0 4.5632217 4.8190457  
4.50220401 4.1171422 4.18041763 4.14150413 4.57450719 4.19151837 4.3232222 4.22631415  
4.4130008 4.63021412 4.45585058 1.27 1.27 0 0 0 0 1.39 0 4.77141904 1.37 0 3.691417 0 4.11140606  
4.8130631 0 3.241420 0 3.20013 0 3.303 0 1.0 0 4.11081304 0 2.1914 0 4.22170819 1.4 0 3.190704 0  
4.4230402 4.20190403 0 2.1417 0 4.4210011 4.20001904 1.3 0 4.2141212 3.1303 0 2.1914 0 1.0 0  
3.111406 0 4.5081104 1.63 1.27 0 0 0 0 1.39 0 4.77141904 1.37 0 3.691417 0 4.20180417 0 4.201907  
4.4131908 4.2001908 3.141331 0 3.241420 0 3.20013 0 3.303 0 1.0 0 4.11081304 0 2.1914 0 4.2070402  
1.10 0 4.20180417 0 4.2170403 4.4131908 3.1118 0 4.1040514 2.1704 0 4.4230402 4.20190813 1.6 0  
2.1417 0 4.4210011 4.20001908 2.1306 0 1.0 0 4.2141212 4.130363 1.27 0 0 0 0 1.27 3.30405 0  
4.19042319 4.60040308 4.19141757 2.5837 1.27 0 0 0 0 4.15170813 4.19575082 4.8121511 1.4 0  
4.83042319 0 4.68030819 4.14175058 1.27 0 0 0 0 4.5081104 4.60150019 1.7 0 1.62 0 4.8131520  
4.19575068 4.13190417 0 3.190704 0 4.15001907 0 2.1405 0 3.190704 0 4.19042319 0 4.5081104 0 2.1914  
0 4.4030819 0 3.571417 0 1.0 0 3.130422 0 4.5081104 0 4.13001204 0 2.1914 0 4.2170400 4.19045837 0  
2.5058 1.27 1.27 0 0 0 0 1.39 0 3.831724 0 2.1914 0 4.14150413 0 3.190704 0 4.5081104 0 3.1303 0  
4.17040003 0 3.81918 0 4.2141319 4.4131918 0 4.8131914 0 4.45110813 3.41845 1.27 0 0 0 0 4.19172437  
1.27 0 0 0 0 0 0 0 4.22081907 0 4.14150413 4.57050811 4.4601500 3.190731 0 4.45174558 0 2.18 0  
4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 4.11081304 1.18 0 1.62 0 4.5081104 4.63170400 4.3110813  
4.4185758 1.27 0 0 0 0 0 0 0 4.11081304 1.18 0 1.62 0 4.42110813 4.4631718 4.19170815 2.5758 0  
3.51417 0 4.11081304 0 2.813 0 4.11081304 2.1841 0 0 1.39 0 4.81041214 2.2104 0 4.13042211 3.81304 0  
4.2070017 4.21904 2.1718 1.27 0 0 0 0 4.4230204 2.1519 0 4.69081104 4.77141969 4.14201303  
4.68171714 2.1737 1.27 0 0 0 0 0 0 4.15170813 4.19575069 3.81104 0 3.131419 0 4.5142013 2.363 0  
4.82190017 4.19081306 0 4.22081907 0 2.13 0 4.4121519 1.24 0 4.5081104 3.635058 1.27 0 0 0 0 0 0 0  
4.11081304 1.18 0 1.62 0 2.4241 1.27 1.27 0 0 0 4.22070811 1.4 0 4.83172004 1.37 1.27 0 0 0 0 0 0  
4.15170813 4.19575029 3.139163 0 3.640303 0 4.19042319 2.5058 1.27 0 0 0 0 0 0 4.15170813  
4.19575092 1.63 0 4.85080422 0 4.19042319 2.5058 1.27 0 0 0 0 0 0 4.15170813 4.19575093 1.63 0  
4.84150300 2.1904 0 1.0 0 4.11081304 2.5058 1.27 0 0 0 0 0 0 4.15170813 4.19575094 1.63 0  
4.67041104 2.1904 0 1.0 0 4.11081304 2.5058 1.27 0 0 0 0 0 0 4.15170813 4.19575095 1.63 0  
4.82002104 0 4.2070013 4.6041850 1.58 1.27 0 0 0 0 0 0 4.15170813 4.19575096 1.63 0 4.68230819 0  
4.22081907 3.142019 0 4.18002108 4.13065058 1.27 0 0 0 0 0 0 4.2071408 2.204 0 1.62 0 4.8131520  
4.19575066 4.7141418 1.4 0 2.13 0 4.14151908 3.141337 0 2.5058 1.27 1.27 0 0 0 0 0 0 2.805 0  
4.2071408 2.204 0 2.6262 0 4.45914537 1.27 0 0 0 0 0 0 0 0 0 4.19042319 0 1.62 0 4.8131520  
4.19575068 4.13190417 0 4.24142017 0 4.19042319 1.37 0 2.5058 1.27 0 0 0 0 0 0 0 0 0 4.11081304  
4.18630015 4.15041303 4.57190423 2.1958 1.27 0 0 0 0 0 0 0 0 0 4.15170813 4.19575083 3.42319 0  
4.30304 1.3 0 4.18200202 4.4181805 4.20111124 3.635058 1.27 0 0 0 0 0 0 0 4.4110805 0 4.2071408



3.9158 1.27 0 0 0 0 2.191 0 1.62 0 4.8131520 1.19 0 4.57506813 3.190417 0 4.7040806 2.719 0 2.1405 0  
4.8120006 2.437 0 0 2.5058 1.27 0 0 0 0 1.1 0 1.62 0 4.8131957 3.19158 1.27 0 0 0 0 4.15081360 2.1591 0  
1.62 0 4.320207 4.130604 1.27 0 0 0 0 4.15081360 2.1592 0 1.62 0 4.1320207 4.130604 2.6007 1.27 0 0 0 0  
1.22 0 1.62 0 4.66001321 4.185712 4.181904 2.1731 0 4.22080319 2.762 0 2.31 0 4.7040806 3.71962 0  
2.158 1.27 1.27 0 0 0 0 2.3902 0 1.62 0 4.42501520 4.17151104 2.5031 0 4.50061704 4.4135031 0  
4.500614111 3.35031 0 4.50170403 2.5031 0 4.50240411 4.11142250 1.31 0 4.50141700 4.13060450 1.31 0  
4.50150813 3.105031 0 4.50011714 4.22135031 0 4.50022400 3.135031 0 4.50110812 3.45031 0  
4.50190400 3.115031 0 4.50120006 4.4131900 2.5041 1.27 0 0 0 0 1.2 0 1.62 0 2.4241 1.27 0 0 0 0  
4.22070811 1.4 0 4.83172004 1.37 1.27 0 0 0 0 0 0 0 4.151560 3.41303 0 1.62 0 4.8131520 4.19575068  
4.13190417 0 4.2141114 2.2017 0 3.710423 0 4.22081907 0 1.39 0 2.1417 0 4.2141114 2.2017 0  
4.13001204 1.37 0 2.5058 1.27 0 0 0 0 0 0 0 4.2630015 4.15041303 4.57001515 4.60041303 1.58 1.27 0 0  
0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 0 4.2141319 2.860 0 1.62 0 4.8131520 4.19575068 4.13190417 0 1.91 0 2.1914  
0 3.303 0 4.131419 3.70417 0 4.2141114 3.201731 0 1.90 0 2.1914 0 4.12142104 0 3.141337 0 2.5058 1.27  
0 0 0 0 0 0 0 2.805 0 4.2141319 2.860 0 2.4962 0 4.45914537 0 0 1.39 0 4.65170400 2.1018 0 3.190704 0  
4.11141415 0 2.805 0 4.8131520 1.19 0 2.818 0 3.131419 0 3.459145 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.1170400  
1.10 1.27 0 0 0 0 2.2302 0 1.62 0 1.90 1.27 0 0 0 0 4.25041714 1.23 0 1.62 0 1.90 1.27 0 0 0 0 4.25041714  
1.24 0 1.62 0 1.90 1.27 0 0 0 0 1.15 0 1.62 0 1.91 1.27 0 0 0 0 4.17001306 4.4600514 1.17 0 1.62 0  
4.8131957 4.57003202 4.7001306 4.4585657 4.1320207 4.130604 4.60075858 1.27 0 0 0 0 4.13001204 0  
1.62 0 1.91 1.27 0 0 0 0 4.2041111 1.18 0 1.62 0 4.57570032 4.32020700 4.13060458 4.56570132  
4.32020700 4.13060460 3.75858 1.27 0 0 0 0 4.20151504 1.17 0 1.62 0 4.2041111 1.18 0 1.59 0 1.91 1.27 0  
0 0 0 4.13011760 4.2141201 0 1.62 0 4.12001907 4.63151422 4.57110413 4.57025831 4.2041111 2.1858  
1.27 0 0 0 0 4.5081104 1.18 0 1.62 0 4.8131957 4.13011760 4.2141201 1.58 1.27 0 0 0 0 4.39151708  
4.13195711 4.4135702 2.5858 1.27 0 0 0 0 4.17142213 0 1.62 0 1.90 1.27 0 0 0 0 4.8120660 4.5136015  
4.17040508 1.23 0 1.62 0 4.8131520 4.19575068 4.13190417 0 4.72120006 1.4 0 4.5081104 4.13001204 0  
4.15170405 2.823 0 4.57141208 1.19 0 3.190704 0 4.5081104 0 4.4231904 4.13180814 1.13 0 4.13001204  
2.5837 0 2.5058 1.27 0 0 0 0 4.5081104 4.60021420 2.1319 0 1.62 0 1.90 0 0 0 0 1.27 0 0 0 0 3.51417 0 1.19  
0 2.813 0 4.17001306 4.4579031 4.5081104 3.185837 1.27 0 0 0 0 0 0 0 4.5081104 4.60021420 2.1319 0  
1.62 0 4.5081104 4.60021420 2.1319 0 1.61 0 1.91 1.27 0 0 0 0 0 0 0 4.39080557 4.5081104 4.60021420  
2.1319 0 2.6262 0 4.91929158 1.37 1.27 0 0 0 0 0 0 0 1.39 0 0 0 0 4.1170400 1.10 1.27 0 0 0 0 0 0 0  
4.18220819 2.207 0 1.62 0 1.91 1.27 0 0 0 0 0 0 0 2.1822 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 4.39051417 0  
1.23 0 2.813 0 4.17001306 4.4571700 4.13060460 4.5141758 1.37 1.27 0 0 0 0 0 0 0 4.39051417 0 1.23 0  
2.813 0 4.17001306 1.4 0 4.57903102 4.4111118 2.5837 1.27 0 0 0 0 0 0 0 1.23 0 1.62 0 1.90 1.27 0 0 0 0  
0 0 0 0 3.21411 0 1.62 0 4.2041111 1.18 0 1.59 0 1.91 1.27 0 0 0 0 0 0 0 4.39151708 4.13195717  
4.14221358 1.27 0 0 0 0 0 0 0 0 4.22070811 3.45723 0 1.33 0 4.2041111 3.185837 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 1.27 0 0 0 0 0 0 0 0 0 0 2.3905 0 1.62 0 4.14150413 4.57455318 4.63151845 0 1.53 0 4.13001204 1.31  
0 4.45220145 1.58 1.27 0 0 0 0 0 0 0 0 0 4.39056302 4.11141804 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.8055718 4.22081902 1.7 0 2.6262 0 3.915837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.171422 0 1.62 0 1.91  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.13231104 2.519 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.13231708 3.60719 0 1.62 0 4.2070013 2.604 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.13241104 2.519 0 1.62 0  
1.90 1.27 0 0 0 0 0 0 0 0 0 0 0 4.13241708 3.60719 0 1.62 0 4.2070013 4.6046007 1.27 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 4.25041714 1.23 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.25041714 1.24 0 1.62  
0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 4.2601104 4.13061907 0 1.62 0 4.11041357 2.258 0 0 0 1.27 0 0 0 0 0 0  
0 0 0 0 0 4.18220819 2.207 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 4.39170013 0 1.62 0 4.17001303  
4.14126317 4.130308 4.13195790 4.31026011 4.4130619 1.7 0 1.59 0 2.9158 1.27 1.27 0 0 0 0 0 0 0 0 0 0  
0 4.17030821 0 1.62 0 4.12001907 4.63151422 4.57110413 4.57025831 4.2141158 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 4.2041111 0 1.62 0 4.57171422 4.13321703 3.82158 0 1.53 0 4.57110413 4.57025858 1.27 1.27 0 0 0 0  
0 0 0 0 0 0 4.2041111 2.403 0 1.62 0 4.8131957 4.2041111 1.58 1.27 0 0 0 0 0 0 0 0 0 0 4.15170813  
4.19570204 4.11110403 1.58 1.27 1.27 0 0 0 0 0 0 0 0 0 4.39226302 4.17040019 4.4601704  
4.2190013 4.6110457 4.25041714 2.2331 0 4.13241104 3.51931 0 4.13231708 4.6071931 4.13241708  
4.6071931 0 4.5081111 0 1.62 0 4.2420204 4.11110403 2.4131 0 4.14201911 3.81304 0 1.62 0 4.50011100

4.2105031 0 4.22080319 1.7 0 1.62 0 2.9058 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.22630217 4.4001904 4.60170402  
4.19001306 4.11045725 4.4171423 1.31 0 4.13241104 3.51931 0 4.13231708 4.6071931 4.13241708  
4.6071931 0 4.5081111 0 1.62 0 4.2420204 4.11110403 2.4131 0 4.14201911 3.81304 0 1.62 0 4.50011100  
4.2105031 0 4.22080319 1.7 0 1.62 0 2.9058 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.8055723 0 2.3362 0 4.2041111  
3.185837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 3.21411 0 1.62 0 3.21411 0 1.59 0 1.91 1.27 1.27 0 0 0 0 0 0 0  
0 0 0 0 4.22630617 4.8035717 2.1422 0 1.62 0 4.25041714 2.2431 0 4.2141120 2.1213 0 1.62 0 4.25041714  
1.23 0 1.61 0 4.2070013 3.60458 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.8055715 0 2.3462 0 4.15081360 2.1591 0  
3.1303 0 4.15531508 4.13601591 0 2.6262 0 3.905837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.25041714 1.24 0  
1.62 0 4.25041714 1.24 0 1.61 0 4.2070013 4.6046007 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.25041714 1.23 0  
1.62 0 4.59020700 3.130604 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.39250417 2.1423 0 1.62 0 1.90 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 4.13231104 2.519 0 1.62 0 4.2070013 2.604 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.13231708 3.60719 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.13241104 2.519 0 1.62 0 4.13241104  
2.519 0 1.61 0 4.2070013 4.6046007 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.13241708 3.60719 0 1.62 0  
4.13241708 3.60719 0 1.61 0 4.2070013 4.6046007 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.25041714 1.23 0 1.62 0  
4.25041714 1.23 0 1.61 0 4.2070013 2.604 1.27 0 0 0 0 0 0 0 0 0 0 0 0 1.15 0 1.62 0 1.15 0 1.61 0 1.91 1.27  
0 0 0 0 0 0 0 0 0 0 0 0 0 3.392302 0 1.61 0 1.91 1.27 0 0 0 0 0 0 0 0 0 0 0 4.13231708 3.60719 0 1.62 0  
4.13231708 3.60719 0 1.61 0 4.2070013 2.604 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.8055723 1.2 0 2.6262 0  
3.935837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 2.2302 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 0 1.23 0 1.62 0  
1.23 0 1.61 0 1.91 0 0 0 1.27 0 0 0 0 0 0 0 4.17142213 0 1.62 0 4.17142213 0 1.61 0 1.91 1.27 0 0 0 0 0  
0 0 2.204 0 1.62 0 4.18191757 4.13001204 1.58 1.27 0 0 0 0 0 0 0 4.22632015 4.3001904 2.5758 1.27 0 0  
0 0 0 0 0 4.39226315 4.14181918 4.2170815 4.19570508 2.1104 0 1.62 0 2.204 0 1.61 0 4.50631518  
2.5031 0 4.21411114 4.17121403 4.4624502 4.14111417 2.4558 1.27 0 0 0 0 0 0 0 0 4.22631514 4.18191802  
4.17081519 4.57050811 4.4620812 4.6600513 4.60151704 3.50823 0 1.61 0 2.204 0 1.61 0 4.50631518  
2.5031 0 4.21411114 4.17121403 4.4624502 4.14111417 2.4531 0 4.23629031 0 4.24629031 0 4.22080319  
4.7620031 0 4.7040806 4.7196201 1.58 1.27 0 0 0 0 0 0 0 4.13001204 0 1.62 0 4.13001204 0 1.61 0 1.91  
1.27 1.27 3.30405 0 4.2141215 4.8110460 4.8120006 4.4600508 4.11045758 1.37 1.27 0 0 0 0 3.131405 0  
1.62 0 4.8131520 4.19575072 4.13152019 0 4.13201201 2.417 0 2.1405 0 4.5081104 2.1837 0 2.5058 1.27  
0 0 0 0 4.13140560 0 1.62 0 4.8131957 4.13140558 1.27 0 0 0 0 4.8131860 1.23 0 1.62 0 4.8131520  
4.19575068 4.13190417 0 4.5081718 1.19 0 4.5081104 0 3.80337 0 2.5058 1.27 0 0 0 0 1.23 0 1.62 0  
4.8131957 4.8131860 2.2358 1.27 0 0 0 0 4.3141320 0 1.62 0 4.8131520 4.19575067 1.14 0 3.241420 0  
4.7002104 0 4.12141704 0 4.19070013 0 3.141304 0 4.5081104 0 2.5724 0 4.42880418 2.4131 0 1.13 0  
4.42771441 1.35 0 2.5058 1.27 0 0 0 0 4.8055703 3.141320 0 2.6262 0 4.45244558 1.37 1.27 0 0 0 0 0 0 0  
4.8131860 1.24 0 1.62 0 4.8131520 4.19575068 4.13190417 0 4.11001819 0 4.5081104 0 3.80337 0 2.5058  
1.27 0 0 0 0 0 0 0 1.24 0 1.62 0 4.8131957 4.8131860 2.2458 1.27 0 0 0 0 3.81318 0 1.62 0 4.8131520  
4.19575068 4.13190417 0 4.15001907 0 2.1405 0 4.5081104 2.1837 0 2.5058 1.27 1.27 0 0 0 0 4.39030405  
0 4.2141321 4.4171960 4.19146015 4.13065715 4.190758 1.37 1.27 0 0 0 0 3.151704 0 1.62 0 4.8131520  
4.19575068 4.13190417 0 4.5081104 4.13001204 4.59151704 4.5082337 0 2.5058 1.27 0 0 0 0 4.8055713  
3.140560 0 1.34 0 3.915837 1.27 0 0 0 0 0 0 0 3.51417 0 1.8 0 2.813 0 4.17001306 4.4572331 4.24619158  
1.37 1.27 0 0 0 0 0 0 0 0 0 0 3.82324 0 1.62 0 4.18191757 2.858 1.27 0 0 0 0 0 0 0 0 0 0 0 4.39151704  
0 1.62 0 4.8131520 4.19575068 4.13190417 0 4.5081104 4.13001204 4.59151704 4.5082337 0 2.5058 1.27  
0 0 0 0 0 0 0 0 0 0 4.15001907 1.62 0 3.81318 0 1.61 0 4.50292950 0 1.61 0 3.151704 0 1.61 0 3.82324 0  
1.61 0 4.50631518 1.50 1.27 0 0 0 0 0 0 0 0 0 3.81206 0 1.62 0 4.72120006 4.4631415 4.4135715  
4.190758 1.27 0 0 0 0 0 0 0 0 0 0 4.8120663 4.18002104 4.57151704 0 1.61 0 3.82324 0 1.61 0  
4.50631513 3.65058 1.27 0 0 0 0 4.8055713 3.140560 0 2.6262 0 3.915837 1.27 0 0 0 0 0 0 0 0 4.39151704  
0 1.62 0 4.8131520 4.19575068 4.13190417 0 4.5081104 4.13001204 4.59151704 4.5082337 0 2.5058 1.27  
0 0 0 0 0 0 0 4.15001907 0 1.62 0 3.81318 0 1.61 0 4.50292950 0 1.61 0 3.151704 0 1.61 0 4.8131860 1.23  
0 1.61 0 4.50631518 1.50 1.27 0 0 0 0 0 0 0 0 3.81206 0 1.62 0 4.72120006 4.4631415 4.4135715 4.190758  
1.27 0 0 0 0 0 0 0 4.8120663 4.18002104 4.57151704 0 1.61 0 4.8131860 1.23 0 1.61 0 4.50631513  
3.65058 1.27 1.27 0 0 0 0 4.15170813 4.19575067 4.14130450 1.58 1.27 1.27 1.27 0 0 0 0 4.39120018  
4.19041763 4.12000813 4.11141415 2.5758 1.27 1.27 3.30405 0 4.15171406 4.17001212 4.8130660

4.4130608 4.13045758 1.37 1.27 0 0 0 0 4.39393939 4.8121514 2.1719 0 4.15242204 4.1011714  
4.22180417 1.27 0 0 0 0 4.39021704 3.1904 0 1.0 0 4.2201819 2.1412 0 4.4210413 1.19 0 4.19241504 1.27  
0 0 0 0 4.76886066 4.84828378 4.76606885 3.687783 0 1.62 0 4.15240600 4.12046384 4.82688168  
4.85687783 0 1.61 0 1.91 1.27 1.27 0 0 0 1.39 0 4.72130819 4.8001108 2.2504 0 4.15240600 2.1204 1.27  
0 0 0 0 4.15240600 4.12046308 4.13081957 1.58 1.27 1.27 0 0 0 1.39 0 4.72130819 4.8001108 2.2504 0  
4.9142418 4.19080210 0 4.12140320 2.1104 1.27 0 0 0 0 4.15240600 4.12046309 4.14241819 4.8021063  
4.8130819 2.5758 1.27 1.27 0 0 0 1.39 0 4.66070402 1.10 0 2.805 0 3.1324 0 4.9142418 4.19080210 1.18  
0 3.1704 0 4.2141313 4.4021904 1.3 1.27 0 0 0 2.805 0 4.15240600 4.12046309 4.14241819 4.8021063  
4.6041960 4.2142013 3.195758 0 1.34 0 2.9037 1.27 0 0 0 0 0 0 1.39 0 3.700419 0 3.190704 0  
4.5081718 1.19 0 4.9142418 4.19080210 1.27 0 0 0 0 0 0 0 4.9142418 4.19080210 0 1.62 0 4.15240600  
4.12046309 4.14241819 4.8021063 4.73142418 4.19080210 3.579058 1.27 0 0 0 0 0 0 1.39 0  
4.72130819 4.8001108 2.2504 0 3.190704 0 4.9142418 4.19080210 1.27 0 0 0 0 0 0 0 4.9142418  
4.19080210 4.63081308 3.195758 1.27 1.27 0 0 0 1.39 0 3.820419 0 4.18021704 2.413 0 4.18082504 1.27  
0 0 0 0 4.39041319 2.417 0 1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 3.190704 0 4.3081204  
4.13180814 1.13 0 3.595934 0 4.14151908 3.141318 0 4.94929231 0 3.1303 0 4.12201119 4.8151104 1.18  
0 2.1405 0 4.94929250 2.5858 1.27 0 0 0 0 4.4131904 1.17 0 1.62 0 3.949292 1.27 0 0 0 0 4.18021704  
4.4136022 4.8031907 0 1.62 0 4.4131904 1.17 1.27 0 0 0 0 4.18021704 4.4136007 4.4080607 1.19 0 1.62 0  
4.4131904 1.17 1.27 1.27 1.27 0 0 0 3.11118 0 1.62 0 2.9592 1.27 0 0 0 0 4.18021704 2.413 0 1.62 0  
4.15240600 4.12046303 4.8181511 4.246318 4.4196012 4.14030457 4.57180217 4.4041360 4.22080319  
2.731 0 4.18021704 4.4136007 4.4080607 3.195858 1.27 0 0 0 0 4.39180217 3.40413 0 1.62 0 4.15240600  
4.12046303 4.8181511 4.246318 4.4196012 4.14030457 4.57180217 4.4041360 4.22080319 2.731 0  
4.18021704 4.4136007 4.4080607 3.195831 0 4.15240600 4.12046381 4.68827289 4.64657568 1.58 1.27 0  
0 0 0 4.39393939 4.39393939 4.39393939 4.39393939 2.3939 1.27 1.27 0 0 0 0 4.39661704 3.1904 0  
4.66141212 4.130318 1.27 0 0 0 0 4.1201860 0 1.62 0 3.929595 1.27 0 0 0 0 4.121420 3.131960 0 1.62 0  
4.57012018 1.60 0 1.61 0 2.9158 0 1.56 0 4.57012018 1.60 0 1.61 0 2.9158 1.27 0 0 0 0 4.11081304  
4.60132012 4.1041760 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 1.27 0 0 0 0 4.2170400 4.19046005  
0 1.62 0 4.14150413 4.57506614 4.12120013 4.3608304 4.12151100 4.19046319 4.23195031 0  
4.50225058 1.27 0 0 0 0 4.22070811 4.4571108 4.13046013 4.20120104 2.1760 0 1.33 0 4.121420  
4.13196058 1.37 1.27 0 0 0 0 0 0 0 2.3908 0 1.62 0 4.11140691 4.90571108 4.13046013 4.20120104  
2.1758 1.27 0 0 0 0 0 0 0 0 4.2170400 4.19046005 4.63221708 4.19045750 4.15170813 4.19572950  
4.42681215 2.1924 0 4.66141212 3.1303 0 4.82111419 1.41 0 4.57660700 3.130604 0 4.20180813 1.6 0 1.0  
0 4.19042319 4.59040308 3.191417 0 2.1914 0 4.84150300 2.1904 0 4.19070818 0 4.18111419 0 2.813 0  
4.66141212 4.130360 4.83041215 4.11001904 4.63192319 3.295058 0 4.39170412 4.4120104 4.17081306  
0 2.1914 0 4.17041300 2.1204 0 4.66141212 4.130360 4.83041215 4.11001904 4.63192319 4.29135058  
1.27 0 0 0 0 0 0 0 4.11081304 4.60132012 4.1041760 0 1.62 0 4.11081304 4.60132012 4.1041760 0 1.61  
0 1.91 1.27 1.27 0 0 0 0 4.2170400 4.19046005 4.63021114 4.18045758 1.27 1.27 0 0 0 0 4.39751400  
4.36614142 4.12001303 1.18 1.27 0 0 0 0 3.230813 0 1.62 0 1.90 1.27 0 0 0 0 4.5081113 2.12 0 1.62 0  
4.506614142 4.12001303 4.60830412 4.15110019 4.4631923 2.1950 1.27 0 0 0 0 4.22081907 0 4.14150413  
4.57050811 4.13001231 0 4.45174558 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 4.2141212 4.130318 0  
1.62 0 4.5081104 4.63170400 4.3110813 4.4185758 1.27 0 0 0 0 0 0 0 3.230813 0 1.62 0 3.230813 0 1.61  
0 1.91 1.27 1.27 1.27 0 0 0 0 4.18190017 0 1.62 0 3.429041 1.27 1.27 0 0 0 0 3.51417 0 2.831 0 4.2141212  
3.1303 0 2.813 0 4.4132012 4.4170019 4.4570214 4.12120013 4.3185837 1.27 0 0 0 0 0 0 0 4.2141212  
4.130318 3.420841 0 1.62 0 4.2141212 4.130363 4.18191708 3.155758 1.27 0 0 0 0 0 0 0 4.39021203 0  
1.62 0 4.2141212 4.130363 4.18151108 4.19575031 2.5058 1.27 0 0 0 0 0 0 0 4.18190017 4.63001515  
4.4130357 4.2141212 4.130358 1.27 0 0 0 0 3.30411 0 4.18190017 3.429041 1.27 1.27 0 0 0 0 4.39393939  
4.39393939 4.39393939 4.39393939 3.393939 1.27 1.27 0 0 0 0 4.39393939 4.39393939 4.39393939  
4.39393939 2.3939 1.27 1.27 0 0 0 0 4.39751400 4.3661414 4.17030813 4.190418 1.27 0 0 0 0 4.23081392  
0 1.62 0 1.90 1.27 0 0 0 0 4.22081907 0 4.14150413 4.57456614 4.14170308 4.13001904 4.18607924  
4.19071413 4.63192319 2.4531 0 4.45174558 0 2.18 0 4.5081104 2.9237 1.27 0 0 0 0 0 0 0 4.2141212  
4.130318 1.92 0 1.62 0 4.5081104 4.92631704 4.31108 4.13041857 1.58 1.27 0 0 0 0 0 0 0 0 4.23081392 0

1.62 0 4.23081392 0 1.61 0 1.91 1.27 1.27 0 0 0 0 4.18190017 2.1823 0 1.62 0 3.429041 1.27 0 0 0  
 4.18190017 2.1824 0 1.62 0 3.429041 1.27 1.27 0 0 0 0 3.514170 3.89231 0 4.2141212 4.130392 0 2.813 0  
 4.4132012 4.4170019 4.4570214 4.12120013 4.3189258 1.37 1.27 0 0 0 0 0 0 0 4.2141212 4.130318  
 4.92420892 1.41 0 1.62 0 4.2141212 4.130392 4.63181917 4.8155758 1.27 0 0 0 0 0 0 0 4.2120392 0 1.62  
 0 4.2141212 4.130392 4.63181511 4.8195750 3.315058 1.27 0 0 0 0 0 0 0 4.18190017 4.18236300  
 4.15150413 4.3570813 4.19570212 4.3924291 3.415858 1.27 0 0 0 0 0 0 0 4.21001724 1.18 0 1.62 0  
 4.8131957 4.2120392 4.42924163 4.18191708 4.15575858 1.27 0 0 0 0 0 0 0 4.18190017 4.18246300  
 4.15150413 4.3572100 4.17241858 1.27 0 0 0 0 0 0 0 1.27 0 0 0 0 3.30411 0 4.18190017 4.18234290 1.41  
 1.27 0 0 0 0 3.30411 0 4.18190017 4.18244290 1.41 1.27 1.27 0 0 0 0 3.505050 1.27 0 0 0 0 3.514170  
 3.89331 0 4.2141212 4.130324 0 2.813 0 4.18190017 3.182437 1.27 0 0 0 0 0 0 0 3.130422 0 1.62 0  
 4.18190017 4.18244208 4.93416318 4.19170815 2.5758 1.27 0 0 0 0 0 0 0 4.18190017 4.18244208  
 2.9341 0 1.62 0 4.8131957 4.13042258 1.27 0 0 0 0 3.505050 1.27 1.27 0 0 0 0 4.15170813 4.19571819  
 4.171823 1.58 1.27 0 0 0 0 4.15170813 4.19571819 4.171824 1.58 1.27 0 0 0 0 4.39393939 4.39393939  
 4.39393939 4.39393939 3.393939 1.27 1.27 1.27 0 0 0 0 1.39 0 4.75140003 0 4.8120006 2.418 1.27 0 0 0 0  
 4.1111402 4.10600812 4.60418 0 1.62 0 2.4241 1.27 0 0 0 0 3.51417 0 1.8 0 2.813 0 4.17001306 4.4579295  
 3.965837 1.27 0 0 0 0 0 0 0 0 4.1111402 4.10600812 4.60418 4.63001515 4.4130357 4.15240600  
 4.12046308 4.12000604 4.63111400 4.3570550 4.1111402 4.10430861 4.91446315 4.13065058 1.58 1.27  
 1.27 1.27 0 0 0 1.39 0 4.76002504 0 4.11002414 2.2019 1.27 0 0 0 0 4.12002504 4.60110024 3.142019 0  
 1.62 0 1.42 1.27 0 0 0 0 0 0 0 3.429031 0 2.9131 0 2.9231 0 3.934131 1.27 0 0 0 0 0 0 0 3.429431 0  
 2.9531 0 2.9631 0 3.974131 1.27 0 0 0 0 0 0 0 3.429831 0 2.9931 0 3.919031 0 4.91914131 1.27 0 0 0 0 0  
 0 0 0 4.42919231 0 3.919331 0 3.919431 0 3.919541 1.27 0 0 0 0 1.41 1.27 1.27 1.27 0 0 0 0 4.39692013  
 4.2190814 1.13 0 2.1914 0 4.2070402 1.10 0 2.805 0 1.0 0 4.15141808 4.19081413 0 2.818 0 4.8131808  
 2.304 0 3.190704 0 4.12002504 1.27 0 0 0 0 3.30405 0 4.8186008 4.13180803 4.4601200 4.25045723 1.31  
 0 2.2431 0 4.12002504 4.60220803 3.190731 0 4.12002504 4.60070408 4.6071931 0 4.1111858 1.37 1.27  
 0 0 0 0 0 0 4.17041920 2.1713 0 1.90 0 2.3362 0 1.23 0 1.33 0 4.12002504 4.60220803 2.1907 0 1.56 0  
 3.11118 0 3.1303 0 1.90 0 2.3362 0 1.24 0 1.33 0 4.12002504 4.60070408 3.60719 0 1.56 0 3.11118 1.27  
 1.27 0 0 0 0 1.39 0 4.75140003 0 4.15110024 2.417 0 4.8120006 2.418 1.27 0 0 0 0 4.15110024 4.4176008  
 4.12000604 0 1.62 0 4.15240600 4.12046308 4.12000604 4.63111400 4.3575015 4.11002404 4.17631513  
 3.65058 1.27 0 0 0 0 4.15110024 4.4176008 4.12000604 1.92 0 1.62 0 4.15240600 4.12046308 4.12000604  
 4.63111400 4.3575015 4.11002404 4.17926315 4.13065058 1.27 1.27 0 0 0 0 4.39180200 4.11040360  
 4.1111402 4.10600812 4.60418 0 1.62 0 4.42152406 4.120463 4.19170013 4.18051417 4.12631802  
 4.110457 4.8120631 0 4.57081319 4.57081206 4.63060419 4.60220803 4.19075758 0 1.56 0 4.18020011  
 4.4600500 4.2191417 2.5831 0 4.8131957 4.8120663 4.6041960 4.7040806 4.7195758 0 1.56 0 4.18020011  
 4.4600500 4.2191417 3.585858 0 3.51417 0 3.81206 0 2.813 0 4.1111402 4.10600812 4.60418 1.41 1.27  
 0 0 0 4.39180200 4.11040360 4.15110024 4.4176008 4.12000604 0 1.62 0 4.15240600 4.12046319  
 4.17001318 4.5141712 4.63180200 4.11045715 4.11002404 4.17600812 4.60431 0 4.57081319  
 4.57151100 4.24041760 4.8120006 4.4630604 4.19602208 4.3190757 1.58 0 1.56 0 4.18020011 4.4600500  
 4.2191417 2.5831 0 4.8131957 4.15110024 4.4176008 4.12000604 4.63060419 4.60070408 4.6071957  
 1.58 0 1.56 0 4.18020011 4.4600500 4.2191417 3.585858 1.27 1.27 1.27 0 0 0 0 1.39 0 3.820419 0  
 4.15110024 2.417 0 4.15141808 4.19081413 1.27 0 0 0 0 4.15110024 4.4176023 0 1.62 0 1.90 1.27 0 0 0  
 4.15110024 4.4176024 0 1.62 0 1.90 1.27 0 0 0 0 4.2141360 4.15171406 3.170012 0 1.62 0 4.8131957  
 4.8131520 4.19575066 4.14131908 4.13200435 0 1.91 0 4.42240418 2.4131 0 1.90 0 4.42131441 1.37 0  
 3.505858 1.27 0 0 0 0 4.8055702 4.14136015 4.17140617 2.12 0 2.6262 0 3.915837 1.27 0 0 0 0 0 0 0  
 4.15170813 4.19575078 1.10 0 4.63636350 1.58 1.27 0 0 0 0 4.8055702 4.14136015 4.17140617 2.12 0  
 2.6262 0 3.905837 1.27 0 0 0 0 0 0 4.15170813 4.19575068 4.23081908 2.1306 0 4.63636350 1.58 1.27  
 0 0 0 0 0 0 4.4230819 2.5758 1.27 0 0 0 0 4.8055702 4.14136015 4.17140617 2.12 0 2.4962 0 1.90 0  
 3.1303 0 4.2141360 4.15171406 3.170012 0 2.4962 0 3.915837 1.27 0 0 0 0 0 0 0 4.15170813 4.19575068  
 4.17171417 2.5058 1.27 0 0 0 0 0 0 4.4230819 2.5758 1.27 0 0 0 1.39 0 3.820419 0 4.8130819 3.80011  
 0 4.8131520 1.19 0 4.19241504 1.27 0 0 0 0 4.19241504 1.8 0 1.62 0 3.501050 1.27 0 0 0 0 4.15170813  
 4.19575029 4.13291372 1.13 0 4.75140200 1.11 0 4.66141212 3.1303 0 4.76140304 4.29132913 2.5058















0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.15110024 4.4176024 0 1.62 0 4.12142018 3.46024 0 2.3232 0 3.11118 1.27 1.27 0 0  
0 0 0 0 0 1.27 0 0 0 0 0 0 0 0 1.39 0 4.66110400 1.17 0 4.18021704 2.413 1.27 0 0 0 0 0 0 0 4.18021704  
4.4136305 4.8111157 4.57929595 1.31 0 4.92959531 0 4.92959558 1.58 1.27 1.27 0 0 0 0 0 0 0 1.27 0 0 0  
0 0 0 0 1.27 0 0 0 0 0 0 0 1.39 0 4.67170022 0 4.1111402 2.1018 1.27 0 0 0 0 0 0 0 4.8055719  
4.24150408 0 2.6262 0 3.451045 0 2.1417 0 4.19241504 1.8 0 2.6262 0 3.451245 0 2.1417 0 4.19241504  
1.8 0 2.6262 0 4.45034558 1.37 1.27 0 0 0 0 0 0 0 0 0 0 3.51417 0 1.24 0 2.813 0 4.17001306 4.4570813  
4.19571802 4.17040413 4.60070408 4.6071932 4.1111858 2.5837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
3.51417 0 1.23 0 2.813 0 4.17001306 4.4570813 4.19571802 4.17040413 4.60220803 4.19073201  
4.11185858 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.17041100 4.19082104 2.6023 0 1.62 0 1.23 0  
1.59 0 4.15110024 4.4176023 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.17041100 4.19082104 2.6024 0  
1.62 0 1.24 0 1.59 0 4.15110024 4.4176024 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.18021704  
4.4136301 4.11081957 4.1111402 4.10600812 4.60418 4.42170411 4.190821 3.46023 0 1.61 0 4.17041100  
4.19082104 2.6024 0 1.56 0 4.8131957 4.18021704 4.4136022 4.8031907 0 2.3232 0 4.57011118  
4.58584131 0 2.5723 0 1.56 0 4.1111831 0 1.24 0 1.56 0 4.1111858 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.8055719 4.24150408 0 2.6262 0 3.451045 0 2.1417 0 4.19241504 1.8 0 2.6262 0 3.451245 0 2.1417 0  
4.19241504 1.8 0 2.6262 0 4.45034558 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.18021704 4.4136301  
4.11081957 4.15110024 4.4176008 4.12000604 1.31 0 4.57151100 4.24041760 1.23 0 1.56 0 4.1111831 0  
4.15110024 4.4176024 0 1.56 0 4.1111858 1.58 1.27 1.27 0 0 0 0 0 0 0 4.8055719 4.24150408 0  
2.6262 0 3.451045 0 2.1417 0 4.19241504 1.8 0 2.6262 0 3.451245 0 2.1417 0 4.19241504 1.8 0 2.6262 0  
4.45034558 1.37 1.27 0 0 0 0 0 0 0 0 0 0 1.39 0 4.84150300 2.1904 0 4.3081815 3.110024 1.27 0 0 0 0 0  
0 0 0 0 0 0 0 4.15240600 4.12046303 4.8181511 4.246320 4.15030019 3.45758 1.27 0 0 0 0 0 0 0 0 0 1.27  
1.27 0 0 0 0 1.39 0 4.80200819 0 4.15240600 2.1204 1.27 0 0 0 0 4.15240600 4.12046316 4.20081957 1.58  
1.27 1.27 3.30405 0 4.7041115 4.60151708 4.13195758 1.37 1.27 0 0 0 0 4.15170813 4.19575029  
4.13860411 4.2141204 0 2.1914 0 4.64120011 3.40231 0 4.7041704 0 3.1704 0 4.24142017 0 4.14151908  
3.141318 0 4.63636329 4.13291350 1.58 1.27 0 0 0 0 4.15170813 4.19575029 4.13291979 4.17141405 0  
2.818 0 3.190704 0 4.11804 3.130204 0 2.1405 0 4.3142001 2.1963 0 4.70200418 1.18 0 3.131419 0  
4.12142104 1.63 0 2.6524 0 3.220024 0 2.1405 0 3.190704 0 4.20130117 4.4001000 3.11104 0 4.75002263  
0 4.29132919 4.83070818 0 2.818 0 3.190704 0 4.18141120 4.19081413 1.63 0 4.83070019 0 2.818 0  
4.22070802 1.7 0 4.18141121 2.418 0 4.22081907 3.142019 0 3.190704 0 4.2002018 1.4 0 2.1405 0  
3.141304 0 4.15171401 3.110412 0 4.4230211 4.20180821 3.41124 0 2.1417 0 4.15171401 4.11041218  
4.63291329 4.13861708 4.19190413 0 2.124 0 3.761763 0 4.67141208 3.130802 0 4.64110423 4.130304  
1.17 0 4.66141415 4.4175058 1.27 0 0 0 0 4.15170813 4.19575029 4.13291368 4.13190417 0 2.9037 0  
4.72120006 1.4 0 4.6041304 4.17001914 1.17 0 4.57810416 4.20081704 1.3 0 3.51417 0 3.190704 0  
3.201804 0 2.1405 0 4.92585058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 2.9137 0 4.72120006  
1.4 0 4.66141215 4.8110417 2.5058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 2.9237 0  
4.79171406 4.17001212 3.81306 0 4.68130608 4.13045058 1.27 0 0 0 0 4.15170813 4.19575068  
4.13190417 0 2.9337 0 4.66141812 4.14111406 1.24 0 4.57841804 4.11104 0 4.22081907 0 2.9031 0 1.91 0  
3.1303 0 4.92585058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 2.9437 0 4.82041620 4.4131908  
2.11 0 4.75001306 4.20000604 0 4.70041304 4.17001908 2.1413 0 4.57860017 4.13081306 1.31 0  
3.241420 0 4.12201819 0 4.5081119 2.417 0 3.142019 0 3.190704 0 4.15171405 4.130463 0 4.64111814 0  
1.0 0 4.20180417 0 4.63192319 0 4.5081104 0 4.12201819 0 4.4230818 1.19 0 4.22081907 2.813 0  
3.190704 0 4.22141710 3.81306 0 4.3081704 4.2191417 1.24 0 4.19070019 0 4.11081819 1.18 0  
4.57141304 0 4.2070017 4.21904 1.17 0 3.150417 0 4.11081304 1.58 0 4.84130802 3.140304 0 4.2070017  
4.21904 3.171837 0 2.5058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 2.9537 0 4.83042319 0  
4.68030819 4.14175058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 2.9637 0 4.70041304  
4.17001904 0 4.66141212 3.1303 0 4.76001917 2.823 0 4.79112006 2.813 0 3.51417 0 3.925058 1.27 0 0 0  
0 4.15170813 4.19575068 4.13190417 0 2.9737 0 4.68230819 0 4.79171406 4.17001250 1.58 1.27 0 0 0  
4.15170813 4.19575068 4.13190417 0 2.9837 0 2.9367 0 3.666467 0 3.666467 0 4.68130608 4.13045058  
1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 2.9937 0 4.80200013 3.192012 0 4.66081702 3.200819  
0 4.80081810 2.819 0 4.86141710 4.65041302 3.75058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0

3.919037 0 4.66110018 4.18080200 1.11 0 4.66081702 3.200819 0 4.82020704 4.12671700 1.22 0  
4.86141710 4.65041302 3.75058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 3.919137 0  
4.66141812 4.14111406 4.80200110 4.82081220 4.11001908 2.1413 0 2.124 0 4.83042319 2.5058 1.27 0 0  
0 0 4.15170813 4.19575068 4.13190417 0 3.919237 0 4.75001706 1.4 0 4.75001306 4.20000604 0  
4.76140304 3.115058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 3.919337 0 4.67001900 0  
4.82021708 2.1519 0 2.124 0 4.66201819 2.1412 0 4.82191720 4.2192017 4.4185058 1.27 0 0 0 0  
4.15170813 4.19575068 4.13190417 0 3.919437 0 4.70041304 4.17001904 0 4.65081300 2.1724 0  
4.84130816 2.2004 0 4.82191708 3.130618 0 2.1914 0 1.0 0 4.2201819 4.14126319 2.2319 0 4.5081104  
2.5058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 3.919637 0 4.14201901 0 4.5081104 0 2.1914 0  
4.1081300 4.17241420 1.19 0 4.5081104 0 4.57682319 4.4131808 2.1413 0 2.1405 0 4.91925850 1.58 1.27  
0 0 0 0 4.15170813 4.19575068 4.13190417 0 3.919737 0 4.83042319 0 4.67001900 4.1001804 2.5058 1.27  
0 0 0 0 4.15170813 4.19575068 4.13190417 0 3.919837 0 4.79171423 4.8120011 0 4.79170812 1.4 0  
4.77201201 4.4171850 1.58 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 3.919937 0 4.66141305  
4.8062017 4.190814 2.1318 0 3.51417 0 4.76140304 0 3.925058 1.27 0 0 0 0 4.15170813 4.19575068  
4.13190417 0 3.929137 0 4.79171423 4.8120011 0 4.79170812 2.418 0 4.64130812 4.191417 2.5058 1.27 0  
0 0 0 4.15170813 4.19575068 4.13190417 0 3.929237 0 4.76001320 2.11 0 4.83201708 2.1306 0  
4.66141215 4.11041904 0 4.69081308 2.1904 0 4.82190019 1.4 0 4.76000207 4.8130450 1.58 1.27 0 0 0 0  
4.15170813 4.19575068 4.13190417 0 3.929337 0 4.64201914 4.12001908 1.2 0 4.83201708 2.1306 0  
4.66141215 4.11041904 0 4.69081308 2.1904 0 4.82190019 1.4 0 4.76000207 4.8130450 1.58 1.27 0 0 0 0  
4.15170813 4.19575068 4.13190417 0 3.929437 0 4.79241907 2.1413 0 4.68130001 3.110403 0  
4.66141212 3.1303 0 4.75081304 0 4.72131904 4.17050002 3.45058 1.27 0 0 0 0 4.15170813 4.19575068  
4.13190417 0 3.929537 0 4.64201914 4.12001908 1.2 0 4.66141319 3.171411 0 4.68130608 4.13040417  
4.8130650 1.58 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 3.929637 0 4.64201914 4.12001904 1.3  
0 4.67041808 2.613 0 3.1303 0 4.83042027 4.13141114 4.6245058 1.27 0 0 0 0 4.15170813 4.19575068  
4.13190417 0 3.929737 0 4.66141318 4.19172002 1.19 0 4.24142017 0 3.142213 0 4.75081306 4.20081819  
2.802 0 4.75001306 4.20000604 2.5058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0 3.929837 0  
4.66141321 3.41719 0 4.83042319 0 4.68130214 4.3081306 0 4.57691714 1.12 0 3.919258 0 2.1914 0 1.0 0  
4.66141321 4.4170604 2.1319 0 4.72120006 3.45058 1.27 0 0 0 0 4.15170813 4.19575068 4.13190417 0  
3.929937 0 4.76001907 4.4120019 3.80218 0 4.75000150 1.58 1.27 0 0 0 0 4.15170813 4.19575068  
4.13190417 0 3.939037 0 4.76201119 4.8597500 4.13062000 2.604 0 4.66070017 4.21904 1.17 0  
4.76001515 4.4175058 1.27 0 0 0 0 0 0 0 1.27 4.22070811 4.4578317 4.20045837 1.27 0 0 0 0 1.27 0 0 0 0  
4.4131917 4.130204 0 1.62 0 4.8131957 4.8131520 4.19575029 4.13291329 4.19681319 2.417 0  
4.24142017 0 4.12140304 0 2.1405 0 4.14150417 4.190814 1.13 0 3.579195 0 3.51417 0 4.210008  
4.11000111 1.4 0 4.14151908 4.14131858 1.37 0 3.505858 1.27 0 0 0 0 4.8055704 4.13191700 3.130204 0  
2.6262 0 3.905837 1.27 0 0 0 0 0 0 0 4.8120660 4.6041304 4.17001914 3.175758 1.27 0 0 0 0 4.8055704  
4.13191700 3.130204 0 2.6262 0 3.915837 1.27 0 0 0 0 0 0 0 4.2141215 4.8110460 4.8120006 4.4600508  
4.11045758 1.27 0 0 0 0 4.8055704 4.13191700 3.130204 0 2.6262 0 3.925837 1.27 0 0 0 0 0 0 0  
4.15171406 4.17001212 4.8130660 4.4130608 4.13045758 1.27 0 0 0 0 4.8055704 4.13191700 3.130204 0  
2.6262 0 3.935837 1.27 0 0 0 0 0 0 0 4.2201717 4.4131960 3.21411 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0  
4.2201717 4.4131960 3.171422 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 4.17041808 4.25040360 4.8120006 2.418  
0 1.62 0 2.4241 0 0 1.39 0 4.70111401 2.11 0 4.21001708 4.11104 0 2.1914 0 4.18191417 1.4 0 4.17041808  
3.250403 0 4.8120006 2.418 1.27 1.27 0 0 0 0 0 0 0 0 1.39 0 4.81041819 0 2.1405 0 4.24142017 0  
4.2140304 3.636363 1.27 0 0 0 0 0 0 0 0 1.39 0 4.70111401 2.11 0 4.21001708 4.11104 0 3.51417 0  
3.190704 0 4.8120006 1.4 0 4.18041104 4.2190814 1.13 0 4.5170012 1.4 1.27 0 0 0 0 0 0 0 0 4.8120006  
4.4601804 4.11040219 4.8141360 4.5170012 1.4 0 1.62 0 4.77141304 1.27 0 0 0 0 0 0 0 0 4.18041104  
4.2190403 4.60081200 2.604 0 1.62 0 4.77141304 0 0 1.39 0 4.70111401 2.11 0 4.21001708 4.11104 0  
2.1914 0 4.7141103 0 3.190704 0 4.2201717 4.4131911 1.24 0 4.18041104 4.2190403 0 4.8120006 1.4 1.27  
0 0 0 0 0 0 0 1.39 0 4.70111401 2.11 0 4.21001708 4.11104 0 2.1914 0 4.10040415 0 4.19170002 1.10 0  
2.1405 0 3.190704 0 4.2201717 3.41319 0 4.12140304 1.27 0 0 0 0 0 0 0 4.2201717 4.4131960  
4.12140304 0 1.62 0 4.50031700 4.22600207 3.1750 0 0 1.39 0 4.79141818 4.8011104 0 4.21001120

























4.4170413 4.2045031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641818 4.20121519 4.8141350 1.31 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 4.50821900 4.19041204 4.13195031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791418  
4.19201100 4.19045031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791714 2.1405 0 2.1240 4.68230700  
4.20181908 4.14135031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50822411 4.11140608 4.18125031 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 4.50661413 4.18191720 4.2190821 1.4 0 4.79171414 3.55031 1.27 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.50771413 4.59661413 4.18191720 4.2190821 1.4 0 4.79171414 3.55031 1.27 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.50831708 4.21080011 0 4.79171414 3.55031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50850002  
4.20142018 0 4.79171414 3.55031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50650802 4.14130308 4.19081413  
4.115031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661413 4.3081908 4.14135031 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 4.50822005 4.5080208 4.4130224 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50770402 4.4181808  
4.19245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50802000 4.13190805 4.8041750 1.31 1.27 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.50841308 4.21041718 2.110 4.80200013 4.19080508 4.4175031 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 4.50682308 4.18190413 4.19080011 0 4.80200013 4.19080508 4.4175031 1.27 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 4.50651420 2.1303 0 4.85001708 4.11104 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50691704  
1.4 0 4.85001708 4.11104 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791704 4.3080200 4.19045031  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791714 4.15141808 4.19081413 2.110 4.75140608 3.25031 1.27 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 4.50761403 2.2018 0 4.79141304 4.13185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50761403 2.2018 0 4.83141111 4.4131850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50670818 4.2170419  
1.4 0 4.76001907 4.4120019 4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50820419 0 4.83070414  
4.17245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50692013 4.2190814 3.135031 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 4.50650809 4.4021908 4.14135031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50721309 4.4021908  
4.14135031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50822017 4.9040219 4.8141350 1.31 1.27 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.50681620 4.8210011 4.4130204 0 4.81041100 4.19081413 2.5031 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 4.50790017 4.19080011 0 4.78170304 3.175031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.50831419 2.11  
0 4.78170304 3.175031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50860411 4.11597817 4.3041750 1.31 1.27 0 0  
0 0 0 0 0 0 0 0 0 0 0 4.50810405 4.11042308 4.21081924 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50822412 4.12041917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50831700 4.13180819 4.8210819  
3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641319 4.8182412 4.12041917 3.245031 1.27 0 0 0 0 0 0  
0 0 0 0 0 0 0 4.50661412 4.15110419 4.4130418 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50661412 4.15000219 4.13041818 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661413 4.13040219  
4.4031304 4.18185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661413 4.21041706 4.4130204 2.5031 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 4.50670821 4.4170604 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50750812 4.8195031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50820417 4.8041850 1.31 1.27 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 4.50820417 4.8041850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50761413 4.14191413  
4.8020819 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50660020 3.20724 0 4.82041620 4.4130204  
2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50721305 4.8130819 1.4 0 4.82041950 1.31 1.27 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.50690813 3.81904 0 4.82041950 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 4.50660017  
4.3081300 4.11081924 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661420 4.13190001 2.1104 0  
4.82041950 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50841302 4.14201319 4.11104 0 4.82041950 1.31  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.50822001 4.18041950 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50822015 4.4171804 3.195031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50721319 4.4171804 4.2190814  
3.135031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50841308 4.14135031 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
4.50681215 2.1924 0 4.82041950 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791422 2.4170 4.82041950  
1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50660017 4.19041808 2.13 0 4.79171403 4.20021950 1.31 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 4.50681620 4.8210011 4.4130204 0 4.66110018 3.185031 1.27 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.50790017 4.19081908 4.14135031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50690804  
4.11035031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50810813 3.65031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50701714 4.20155031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50640104 4.11080013 0 4.70171420 3.155031  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50771413 4.59000104 4.11080013 0 4.70171420 3.155031 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 4.50760019 4.17082350 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50850402 3.191417







0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50790700 4.17120002 4.14111406 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 4.50650814 4.8130514 4.17120019 4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50650814  
4.19040207 4.13141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791714 4.19041412 4.8021850  
1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50700413 4.14120802 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50821917 4.20021920 3.170011 0 4.1081411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 4.50822418 4.19041218 0 4.1081411  
4.50850817 4.14111406 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50670421 4.41111415 4.12041319 2.11 0 4.1081411  
4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50682114 4.11201908 4.14130017 1.24 0 4.1081411  
4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50822413 4.19070419 2.802 0 4.1081411 4.14062450  
1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50760419 4.11411 4.14120802 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 4.50681508 4.6041304 4.19080218 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50830818  
3.182004 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50770013  
4.14190402 4.7131411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50760019 4.4170800 2.1118 0  
4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50802000 4.13192012 0 4.15072418  
4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661413 4.3041318 2.403 0 4.12001919 2.417 0  
4.15072418 4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50790017 4.19080211 1.4 0 4.15072418  
4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641819 4.17141507 4.24180802 3.185031 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 4.50661418 4.12141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50781519  
4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641914 3.120802 0 3.1303 0 4.12141104 4.2201100  
1.17 0 4.15072418 4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50691120 2.803 0 4.12040207  
4.130802 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50830704 4.17121403 4.24130012 4.8021850 1.31  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50681321 4.8171413 4.12041319 2.11 0 4.18020804 4.13020450 1.31  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661108 4.12001904 0 4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 4.50700414 4.11140624 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50780204 4.131406  
4.17001507 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641912 4.14181507 4.4170802 0 4.18020804  
4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50680214 4.11140624 2.5031 1.27 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 4.50661413 4.18041721 4.190814 1.13 0 4.1081411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 4.50651419 4.132450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50891414 4.11140624 2.5031 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 0 4.50681319 4.14121411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50760017 3.81304 0 4.1081411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50790011  
4.4141319 4.14111406 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641319 4.7171415 4.14111406  
3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641702 4.7000414 4.11140624 2.5031 1.27 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.50791824 4.2071411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661406  
4.13081908 2.2104 0 4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50821402 3.80011 0  
4.15182402 4.7141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50750813 4.6200818 4.19080218  
2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641719 4.8050802 3.80011 0 4.8131904 4.11110806  
4.4130204 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50760002 4.7081304 0 4.11040017 4.13081306  
2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661412 4.15201904 1.17 0 4.21081808 4.14135031 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 4.50770019 4.20170011 0 4.11001306 4.20000604 0 4.15171402 4.4181808  
4.13065031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50712012 4.135902 4.14121520 3.190417 0 4.8131904  
4.17000219 4.8141350 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50811401 4.14190802 3.185031 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 4.50661412 4.15201904 1.17 0 4.6170015 4.7080218 2.5031 1.27 0 0 0 0 0 0 0 0  
0 0 0 0 0 4.50670019 1.0 0 4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50760019  
4.7041200 4.19080200 1.11 0 4.12140304 4.11081306 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50760019 4.7041200 4.19080200 1.11 0 4.15072418 4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50772012 3.10417 0 4.19070414 4.17245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641106 4.4011700  
2.802 0 4.6041412 4.4191724 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50670805 4.5041704 4.13190800  
1.11 0 4.4162000 4.19081413 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661412 4.15201900  
4.19081413 2.11 0 4.15072418 4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50760019 4.7041200  
4.19080200 1.11 0 4.1081411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50781504 4.17001908

3.141318 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50650814 4.18190019  
4.8181908 4.2185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50681508 4.3041208 4.14111406 3.245031 1.27  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50660013 3.20417 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 4.50670800 4.1041904 1.18 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50710400 2.1719 0 4.3081804 3.1804 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50721305 4.4021908 3.142018 0 4.3081804 4.180418 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 4.50721212 4.20131419 4.7041700 4.15245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50821904 1.12 0 4.2041110 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
1.4 0 4.19070417 4.152450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50671720 1.6 0 4.3081802 4.14210417  
3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791704 4.2081808 2.1413 0 4.12040308 4.2081304 2.5031  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50710400 3.111907 0 4.8130514 4.17120019 4.8021850 1.31 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 4.50810413 4.422001 2.1104 0 4.4130417 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50681304 3.170624 0 4.18191417 4.60450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.19000813 4.11104 0 4.12001904 4.17080011 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.8171413 4.12041319 2.11 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50860019 2.4170 0 4.12001300 4.6041204 4.13195031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.13181514 4.17190019 3.81413 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 4.50660821 2.811 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.12080200 1.11 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.14181500 2.204 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.12040308 3.20011 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50681104 4.2191708 3.20011 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50760402 4.7001308 3.20011 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50811401 4.14190802 1.18 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50802000 4.13192012 0 4.2141215 4.20190813 3.65031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.15191406 4.17001507 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50662401 4.4171804 4.2201708  
4.19245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50770419 4.22141710 0 4.4130608 4.13040417 4.8130650  
1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50830411 4.4021412 4.12201308 4.2001908 4.14131850 1.31 1.27  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50712012 2.13 0 4.6041304 4.19080218 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50691417 4.4131808 1.2 0 4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50821500 2.204 0 4.4231511 4.14170019 3.81413 0 3.1303 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 4.50791100 4.13041900 2.1724 0 4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 4.50641819 4.17140108 4.14111406 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.17140207 4.4120818 4.19172450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641819 4.17140604  
4.14111406 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641819 4.17140813 4.5141712 4.190802  
3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50682314 4.15110013 2.419 0 4.17041804 4.170207 2.5031  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50821904 4.11110017 0 4.4211411 4.20190814 3.135031 1.27 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 4.50700011 4.21908 1.2 0 4.181917 4.14131412 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50780118 4.4172100 4.19081413 2.11 0 4.181917 4.14131412 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50661412 4.15201900 4.19081413 2.11 0 4.181917 4.14150724 4.18080218 2.5031 1.27 0 0 0 0  
0 0 0 0 0 0 0 0 4.50802000 4.13192012 0 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50661412 4.15201900 4.19081413 2.11 0 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 4.50781706 4.130802 0 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50721314 4.17060013 2.802 0 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50790724 4.18080200 1.11 0 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50681321 4.8171413 4.12041319 2.11 0 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 4.50641300 4.11241908 3.20011 0 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50640617 4.8022011 4.19201700 1.11 0 4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50691414 1.3 0 4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.14130011 0 4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4.15072418 4.8141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50821514 3.171918 0 4.18020804  
4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50650814 4.12040207 4.130802 3.185031 1.27 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 4.50791100 2.1319 0 4.15072418 4.8141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50791100 2.1319 0 4.6041304 4.19080218 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791100  
2.1319 0 4.15001907 4.14111406 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50821408 1.11 0  
4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50712403 4.17141114 4.6245031 1.27 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 4.50700414 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50700414 4.15072418 4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50700414 4.12141715  
4.7141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50810412 3.141904 0 4.18041318 4.8130650  
1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50700414 4.19040207 4.13080200 1.11 0 4.4130608 4.13040417  
4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50790419 4.17141104 2.2012 0 4.4130608 4.13040417  
4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50640417 4.14181500 2.204 0 4.12001904 4.17080011  
3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50770013 4.14120019 4.4170800 4.11185031 1.27 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 4.50791411 4.24120417 0 4.18020804 4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 4.50661412 4.15201900 4.19081413 2.11 0 4.12001904 4.17080011 1.18 0 4.18020804 4.13020450 1.31  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50790714 4.19141308 4.2185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50790724 4.18080200 1.11 0 4.14151908 4.2185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50802000  
4.13192012 0 4.14151908 4.2185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50770420 4.17140413 4.6081304  
4.4170813 3.65031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50651700 2.813 0 4.8120006 4.8130650 1.31 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661406 4.13081908 2.2104 0 4.13042017 4.14180208 4.4130204 2.5031  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50770420 4.17140813 4.5141712 4.190802 3.185031 1.27 0 0 0 0 0  
0 0 0 0 0 0 0 0 4.50791824 4.2071415 4.7241808 4.2185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50670421 4.41111415 4.12041319 2.11 0 4.15182402 4.7141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 4.50790417 4.18141300 4.11081924 0 4.15182402 4.7141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 4.50661108 4.13080200 1.11 0 4.15182402 4.7141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50721303 4.20181917 4.8001159 4.14170600 4.13082500 4.19081413 2.11 0 4.15182402 4.7141114  
4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50680320 4.2001908 4.14130011 0 4.15182402 4.7141114  
4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791824 4.2071411 4.8130620 4.8181908 4.2185031 1.27  
0 0 0 0 0 0 0 0 0 0 0 0 0 4.50712012 2.13 0 4.6041304 4.19080218 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50682114 4.11201908 4.14130017 1.24 0 4.6041304 4.19080218 2.5031 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50791415 4.20110019 3.81413 0 4.6041304 4.19080218 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50700413 4.4190802 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50700413 4.4190802 0 4.2124013 4.18041108 4.13065031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.6041314 4.12080218 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50660017 4.3081421 4.180220  
3.110017 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50810418 4.15081700  
4.19141724 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.13190417 4.14111406 1.24 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.14021708 4.13141114 2.624 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50770415 4.7171411 3.140624 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50710412 4.1914111 3.140624 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50781507 4.19070011 4.12141114 2.624 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 4.50781719 4.7141504 3.30802 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50670417 4.12001914 4.11140624 0 4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50850419 4.4170813 3.1724 0 4.12040308 4.2081304 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50641308 3.120011 0 4.1040700 4.21081417 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.18041721 4.190814 1.13 0 4.4021411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.31108015 4.1 0 4.1081411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.12041319 2.11 0 4.12080217 4.14010814 4.11140624 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50640617 4.8022011 4.19201700 1.11 0 4.4021413 4.14120802 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 4.50670421 4.41111415 4.12041319 0 4.4021413 4.14120802 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0

4.450650407 4.210814 3.170011 0 4.4021413 4.14120802 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50680214 4.13141204 4.19170802 3.185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50690813 4.130208  
2.11 0 4.12001907 4.4120019 4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50781504 4.17001908  
3.141318 0 4.12001300 4.6041204 4.13195031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50822015 3.151124 0  
4.2070008 1.13 0 4.12001300 4.6041204 4.13195031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50721303  
4.20181917 3.80011 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50712012 4.135902 4.14121520 3.190417 0 4.8131904 4.17000219 4.8141350 1.31 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 4.50850817 4.19200011 0 4.17040011 4.8192450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50642006 4.12041319 2.403 0 4.17040011 4.8192450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50670019  
1.00 4.12081308 4.13065031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50830423 1.19 0 4.12081308 4.13065031  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50650806 0 4.3001900 0 4.130011 4.24190802 3.185031 1.27 0 0 0 0  
0 0 0 0 0 0 0 0 0 4.50661412 4.15201900 4.19081413 2.11 0 4.11081306 4.20081819 4.8021850 1.31  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50802000 4.13192012 0 4.8130514 4.17120019 3.81413 0 4.18020804  
4.13020450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50802000 4.13192012 0 4.2172415 4.19140617  
4.150724 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50650814 4.12041917 4.8021850 1.31 1.27 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 4.50721305 4.14171200 4.19081413 0 4.17041917 4.8042100 3.115031 1.27 0 0 0 0 0  
0 0 0 0 0 0 0 4.50821405 4.19220017 1.4 0 4.4130608 4.13040417 4.8130650 1.31 1.27 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.50661412 4.15201904 1.17 0 4.13041922 4.14171018 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 0 4.50681201 4.4030304 1.3 0 4.18241819 4.4121850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50712012  
4.135917 4.14011419 0 4.8131904 4.17000219 4.8141350 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50661413 4.19171411 0 4.18241819 4.4121850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50650814  
4.15070017 4.12000204 4.20190802 4.111850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50671720 1.6 0  
4.3041108 4.21041724 0 4.18241819 4.4121850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50661108  
4.13080200 1.11 0 4.19170800 4.11185031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50810406 4.4130417  
4.190821 1.4 0 4.12040308 4.2081304 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50640617 4.8022011  
4.19201700 1.11 0 4.1081419 4.4020713 4.14111406 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50791100 2.1319 0 4.1170404 4.3081306 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641308 3.120011  
0 4.1170404 4.3081306 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50691414 1.3 0 4.19040207 4.13141114  
4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791420 4.11191724 0 4.18020804 4.13020450 1.31 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641620 4.22011 4.19201704 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 4.4021411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50750812 4.13141114 4.6245031 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50791415 4.20110019 3.81413 0 4.4021411 4.14062450 1.31 1.27 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50750001 4.3180200 2.1504 0 4.4021411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0  
0 0 0 4.50682114 4.11201908 4.14130017 1.24 0 4.4021411 4.14062450 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50681321 4.8171413 4.12041319 2.11 0 4.19142308 4.2141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.50681321 4.8171413 4.12041319 2.11 0 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 4.50681321 4.8171413 4.12041319 2.11 0 4.12080217 4.14010814 4.11140624 2.5031 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 4.50680214 4.19142308 4.2141114 4.6245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.50701704 2.413 0 4.2070412 4.8181917 3.245031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50821500 2.204 0  
4.15072418 4.8021850 1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50821500 2.204 0 4.22040019 4.7041750  
1.31 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50641819 4.17141819 4.190818 4.19080218 2.5031 1.27 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 4.50661412 4.15201900 4.19081413 2.11 0 4.5112008 1.3 0 4.3241300 4.12080218  
2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50760019 4.7041200 4.19080200 1.11 0 4.14151908  
4.12082500 4.19081413 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50781504 4.17001908 3.141318 0  
4.17041804 4.170207 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50712012 2.13 0 4.6041304 4.19080218  
2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50692013 4.2190814 3.130011 0 4.6041314 4.12080218 2.5031  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.50761411 4.4022011 2.17 0 4.6041304 4.19080218 2.5031 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 4.50660013 3.20417 0 4.6041304 4.19080218 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 0 4.50791824 4.2070800 4.19170802 0 4.6041304 4.19080218 2.5031 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0



4.11200431 0 4.5081104 2.5837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.8130304 3.239160 0 1.62 0  
4.17602100 3.112004 0 1.53 0 4.11041357 4.8130304 4.23609258 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.8130304 3.239260 0 1.62 0 4.6602100 3.112004 0 1.53 0 4.11041357 4.8130304 4.23609258 1.27 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.8130304 3.239360 0 1.62 0 4.1602100 3.112004 0 1.53 0 4.11041357 4.8130304  
4.23609258 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.5081104 4.63221708 4.19045705 4.50291929 4.19291943  
4.8130304 4.23609242 4.8130304 4.23916041 4.44291350 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.5081104 4.63221708 4.19045705 4.50291929 4.19291943 4.8130304 4.23609242 4.8130304 4.23926041  
4.44291350 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.5081104 4.63221708 4.19045705 4.50291929  
4.19291943 4.8130304 4.23609242 4.8130304 4.23936041 4.44291350 1.58 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 4.5141319 4.81001908 1.14 0 1.62 0 4.18191757 4.8130304 4.23916058 0 1.61 0 3.503750 0 1.61  
0 4.18191757 4.8130304 4.23926058 0 1.61 0 3.503750 1.61 0 4.18191757 4.8130304 4.23936058 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.5141319 4.81001908 1.14 1.27 1.27 1.27 1.27 0 0 0 0 0 0  
0 0 0 0 0 3.30405 0 4.6041960 4.8130304 4.23570831 0 2.931 0 4.6170803 4.60180825 3.45837 1.27 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.17041920 2.1713 0 1.8 0 1.56 0 4.6170803 4.60180825 1.4 0 1.61 0 1.9 1.27  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.30405 0 4.4231917 4.21960 4.17060160 4.21001120 4.4185702 4.14111417  
2.5837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2.805 0 4.11041357 4.2141114 2.1758 0 2.4962 0 1.97 0 2.1417 0  
4.2141114 4.17429041 0 2.4962 0 4.45394537 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.17000818 1.4 0  
4.85001120 4.4681717 4.14175745 4.72131520 1.19 0 4.18071420 2.1103 0 2.104 0 1.0 0 3.70423 0  
4.2141114 1.17 0 4.2140304 0 2.813 0 3.190704 0 4.5141712 2.19 0 4.50398181 4.70706565 3.504558 1.27  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.19172437 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.170403 0 1.62 0  
4.8131957 4.2141114 4.17429137 3.934131 0 3.919658 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.6170404 1.13 0 1.62 0 4.8131957 4.2141114 4.17429337 3.954131 0 3.919658 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 4.1112004 0 1.62 0 4.8131957 4.2141114 4.17429537 3.974131 0 3.919658 1.27 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.17040331 0 4.6170404 2.1331 0 4.1112004 1.27 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 4.4230204 2.1519 0 4.85001120 4.4681717 3.141737 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 4.17000818 1.4 0 4.85001120 4.4681717 4.14175745 4.72132100 3.110803 0 4.2141114 1.17 0  
4.2140304 1.63 0 3.817065 0 4.21001120 2.418 0 4.18071420 2.1103 0 2.104 0 3.70423 0 4.3080608  
2.1918 0 4.57905999 1.31 0 4.64596958 2.4558 1.27 1.27 1.27 0 0 0 0 0 0 0 0 0 0 3.30405 0 4.70785758  
1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.11140660 4.5081104 0 1.62 0 4.50021411 4.14176011 4.14066319  
3.231950 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2.3939 0 0 0 0 0 0 0 0 4.19141508 4.2600508 2.1104 0 1.62 0  
4.50081303 4.4236091 4.63192319 1.50 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2.3939 0 0 0 0 0 0 0  
4.19141508 4.2926005 3.81104 0 1.62 0 4.50081303 4.4236092 4.63192319 1.50 0 0 1.39 0 4.66141717  
4.4021904 1.3 0 4.19070818 0 4.11081304 0 2.1914 0 4.17040003 0 4.5171412 0 1.0 0 4.3080505  
4.4170413 1.19 0 4.5081104 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.6170803 4.86080319 1.7 0 1.62 0  
4.8131957 4.8131520 4.19575068 4.13190417 0 4.6170803 4.86080319 2.737 0 3.505858 1.27 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 1.39 0 4.81040003 0 4.19141508 1.2 0 4.5081104 1.18 0 2.1914 0 3.60419 0 3.190704 0  
4.19141508 2.218 0 3.1303 0 4.19141508 3.29218 1.27 0 0 0 0 0 0 0 0 0 0 2.3939 0 0 0 0 0 0  
4.22081907 0 4.14150413 4.57191415 4.8026005 4.8110431 0 4.50175031 0 4.4130214 4.3081306  
4.62452019 4.5984558 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 2.3939 0 0 0 0 0 0 0 0  
0 0 4.19141508 2.218 0 1.62 0 4.42110813 4.4631819 4.17081557 1.58 0 3.51417 0 4.11081304 0 2.813 0  
4.5081104 1.41 1.27 0 0 0 0 0 0 0 0 0 0 0 0 2.3939 1.27 0 0 0 0 0 0 0 0 0 0 0 0 2.3939 0 0 0 0  
0 0 0 4.22081907 0 4.14150413 4.57191415 4.8029260 4.5081104 1.31 0 4.50175031 0 4.4130214  
4.3081306 4.62452019 4.5984558 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 2.3939 0 0 0 0  
0 0 0 0 0 0 0 4.19141508 3.29218 0 1.62 0 4.42110813 4.4631819 4.17081557 1.58 0 3.51417 0  
4.11081304 0 2.813 0 4.5081104 1.41 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.13201283 4.14150802 1.18  
0 1.62 0 4.11041357 4.8130304 4.23609158 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.13201283 4.14150802  
2.9218 0 1.62 0 4.11041357 4.8130304 4.23609258 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 1.39 0  
4.81040003 0 4.2141114 1.17 0 3.111406 0 4.5081104 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.22081907 0  
4.14150413 4.57111406 4.60050811 2.431 0 4.50175058 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 4.11081304 1.18 0 1.62 0 4.5081104 4.63170400 4.3110813 4.4185758 1.27 1.27 0 0 0 0





0 0 0 0 0 0 0 0 4.12041320 3.10017 0 1.62 0 4.19106376 4.4132057 4.17141419 1.58 1.27 1.27 0 0 0 0 0 0 0  
0 0 0 0 1.39 0 4.66170400 2.1904 0 2.13 0 4.14151908 3.141318 0 4.12041320 0 3.1303 0 3.303 0 2.819 0  
2.1914 0 3.190704 0 4.12041320 3.10017 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.14151908 4.14131860 4.12041320 0  
1.62 0 4.19106376 4.4132057 4.12041320 4.1001731 0 4.19040017 4.14050562 2.9058 1.27 0 0 0 0 0 0 0 0  
0 0 0 0 4.39120413 4.20010017 4.63000303 4.60020018 4.2000304 4.57110001 4.4116250 4.78151908  
4.14131850 1.31 0 4.12041320 4.62141519 4.8141318 4.60120413 2.2058 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
1.39 0 4.66170400 2.1904 0 1.0 0 4.3171415 4.3142213 0 4.12041320 0 3.1303 0 3.303 0 2.819 0 2.1914 0  
3.190704 0 4.12041320 3.10017 1.27 0 0 0 0 0 0 0 0 0 0 4.3171415 4.3142213 0 1.62 0 4.19106376  
4.4132057 4.12041320 4.1001731 0 4.19040017 4.14050562 2.9058 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.1001763 4.30360 4.2001802 4.30457 4.11000104 4.11625078 4.15190814 4.13185031 0 4.12041320  
4.62031714 4.15031422 2.1358 0 1.39 0 3.841804 0 1.0 0 4.3080505 4.4170413 1.19 0 4.11000104 2.1131  
0 4.4630663 1.31 0 4.50671714 4.15031422 2.1350 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 0 1.39 0 3.640303 0  
4.2141212 4.130318 0 2.1914 0 4.3171415 4.3142213 0 4.12041320 1.27 0 0 0 0 0 0 0 0 0 0 0 0 1.27 0 0 0 0  
0 0 0 0 0 0 0 0 1.39 0 3.640303 0 4.2070402 4.10012019 3.191413 0 2.1914 0 4.14151908 3.141318 0  
4.12041320 1.27 0 0 0 0 0 0 0 0 0 0 0 4.3171415 4.3142213 4.63000303 4.60021412 4.12001303  
4.57110001 4.4116250 4.83140606 2.1104 0 4.76001320 4.113281 4.130314 1.12 0 4.66141114  
4.20175031 0 4.2141212 4.130362 4.12001320 4.116002 4.14111420 2.1758 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0  
0 4.3171415 4.3142213 4.63000303 4.60021412 4.12001303 4.57110001 4.4116250 4.81040517  
4.4180750 1.31 0 4.2141212 4.130362 4.17040517 4.4180760 4.2001321 3.1858 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
4.3171415 4.3142213 4.63000303 4.60021412 4.12001303 4.57110001 4.4116250 4.84130314 2.5031 0  
4.2141212 4.130362 4.20130314 4.60110018 2.1958 1.27 0 0 0 0 0 0 0 0 0 0 4.3171415 4.3142213  
4.63000303 4.60021412 4.12001303 4.57110001 4.4116250 4.82002104 0 4.82041818 4.8141350 1.31 0  
4.2141212 4.130362 4.18002104 4.60180418 4.18081413 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 4.3171415  
4.3142213 4.63000303 4.60021412 4.12001303 4.57110001 4.4116250 4.75140003 0 4.82041818  
4.8141350 1.31 0 4.2141212 4.130362 4.1140003 4.60180418 4.18081413 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.3171415 4.3142213 4.63000303 4.60021412 4.12001303 4.57110001 4.4116250 4.82002104 0 2.18 0  
4.72120006 3.45031 0 4.2141212 4.130362 4.18002104 4.60020013 4.21001858 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.3171415 4.3142213 4.63000303 4.60021412 4.12001303 4.57110001 4.4116250 4.66141215 3.81104 0  
4.66001321 4.185031 0 4.2141212 4.130362 4.2141215 4.8110470 4.78821287 1.58 1.27 0 0 0 0 0 0 0 0 0  
0 0 1.39 0 3.640303 0 3.190704 0 3.130422 0 4.2141212 3.1303 0 3.51417 0 4.17041808 4.25081306 0  
4.8120006 2.418 1.27 0 0 0 0 0 0 0 0 0 4.3171415 4.3142213 4.63000303 4.60021412 4.12001303  
4.57110001 4.4116250 4.82041104 2.219 0 3.1303 0 4.81041808 2.2504 0 4.72120006 3.45031 0  
4.2141212 4.130362 4.18041104 4.2196000 4.13036017 4.4180825 4.4600812 4.60458 1.27 0 0 0 0 0 0 0 0  
0 0 0 1.39 0 3.640303 0 3.190704 0 4.2141212 3.1303 0 3.51417 0 4.18041104 4.2190813 1.6 0  
4.8120006 2.418 1.27 0 0 0 0 0 0 0 0 4.3171415 4.3142213 4.63000303 4.60021412 4.12001303  
4.57110001 4.4116250 4.82041104 2.219 0 4.72120006 4.41805031 0 4.2141212 4.130362 4.18041104  
4.2196000 4.13036000 4.3036008 4.12000604 2.1858 1.27 0 0 0 0 0 0 0 0 0 0 1.39 0 3.640303 0  
3.190704 0 4.19140606 2.1104 0 4.12140304 0 4.2141212 3.1303 1.27 0 0 0 0 0 0 0 0 0 0 4.3171415  
4.3142213 4.63000303 4.60021412 4.12001303 4.57110001 4.4116250 4.83140606 2.1104 0 4.67170022 0  
4.66070017 4.18327212 4.60418 2.5031 0 4.2141212 4.130362 4.19140606 4.11046012 4.14030458 1.27 0  
0 0 0 0 0 0 0 0 4.39031714 4.15031422 4.13630003 4.3600214 4.12120013 4.3571100 4.1041162  
4.50682308 3.195031 0 4.2141212 4.130362 4.4230819 4.60001515 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.39031714 4.15031422 4.13630003 4.3600214 4.12120013 4.3571100 4.1041162 4.50831406 3.61104 0  
4.83241504 4.32661108 2.210 0 4.76140304 2.5031 0 4.2141212 4.130362 4.19140606 4.11046019  
4.24150813 4.6601214 3.30458 1.27 0 0 0 0 0 0 0 0 0 0 4.3171415 4.3142213 4.63000303 4.60021412  
4.12001303 4.57110001 4.4116250 4.64011420 1.19 0 4.79171406 4.17001250 1.31 0 4.2141212 4.130362  
4.18071422 4.60000114 3.201958 1.27 0 0 0 0 0 0 0 0 0 0 4.3171415 4.3142213 4.63000303  
4.60021412 4.12001303 4.57110001 4.4116250 2.7014 0 4.71141204 2.5031 0 4.2141212 4.130362  
4.7141204 1.58 1.27 0 0 0 0 0 0 0 0 0 0 4.3171415 4.3142213 4.63000303 4.60021412 4.12001303  
4.57110001 4.4116250 4.68030819 0 4.70170803 0 4.67081204 4.13180814 4.13185031 0 4.2141212





0 0 0 0 1.27 1.27 1.27 0 0 0 0 0 0 0 0 3.30405 0 4.18041920 4.15601214 4.3045718 3.225837 1.27 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 4.6111401 2.11 0 4.12140304 4.60041319 3.172431 0 4.12140304 4.60110001 3.41131 0  
4.18162000 4.17046018 4.8250431 0 4.6170803 4.60180825 2.431 0 4.2070017 4.60021420 3.131931 0  
4.11001819 4.60031700 2.2213 1.27 0 0 0 0 0 0 0 0 0 0 0 4.6111401 2.11 0 4.17141419 1.31 0 4.2141114  
4.17601214 4.3046021 3.1731 0 4.11140606 4.8130631 0 4.15002018 3.40331 0 4.2001321 4.186022  
4.8031907 1.31 0 4.2001321 4.186007 4.4080607 2.1931 0 4.2001321 2.18 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
4.6111401 2.11 0 4.6170803 4.60180825 4.4601100 4.1041131 0 4.6170803 4.60180825 4.4600413  
4.19172431 0 4.5141319 4.60180825 4.4601100 4.1041131 0 4.5141319 4.60180825 4.4600413  
4.19172431 0 4.30920 4.18196001 4.20191914 1.13 1.27 0 0 0 0 0 0 0 0 0 0 0 4.6111401 2.11 0  
4.18002104 4.60820120 4.19191413 1.31 0 4.11140003 4.60012019 4.19141331 0 4.2070017 4.60110001  
3.41131 0 4.2070017 4.60041319 3.172431 0 4.2070017 4.60131419 4.4601100 4.1041131 0 4.2070017  
4.60131419 2.431 0 4.2070017 4.60012019 3.191413 1.27 0 0 0 0 0 0 0 0 0 0 4.6111401 2.11 0  
4.17040517 4.4180760 4.1201919 3.141331 0 4.20130314 4.60012019 4.19141331 0 4.2141319  
4.17141160 4.5170012 2.431 0 4.18200112 4.8196001 4.20191914 2.1331 0 4.2141131 0 4.2141114 2.1760  
1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.6111401 2.11 0 4.8120006 4.4601804 4.11040219 4.8141360 4.5170012 1.4 0  
0 1.39 0 3.841804 0 3.190704 0 4.6111401 2.11 0 4.21001708 4.11104 1.27 1.27 1.27 0 0 0 0 0 0 0 0  
0 0 0 4.2201717 4.4131960 3.171422 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 4.2201717 4.4131960  
3.21411 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 4.2141114 2.1760 0 1.62 0 1.91 1.27 0 0 0 0 0 0 0 0 0 0  
1.27 0 0 0 0 0 0 0 0 0 0 0 1.39 0 4.67040508 2.1304 0 3.190704 0 4.18082504 0 2.1405 0 3.190704 0  
4.18162000 3.170418 0 3.1303 0 3.190704 0 4.6170803 1.27 0 0 0 0 0 0 0 0 0 0 4.39181620 4.170460  
4.18082504 0 1.62 0 2.9790 1.27 0 0 0 0 0 0 0 0 0 0 4.18162000 4.17046018 3.82504 0 1.62 0 4.8131957  
4.8131520 4.19575068 4.13190417 0 4.18200159 4.18162000 2.1704 0 4.18082504 1.37 0 3.505858 1.27 0  
0 0 0 0 0 0 0 0 0 0 0 4.39061708 4.3601808 2.2504 0 1.62 0 2.9190 1.27 0 0 0 0 0 0 0 0 0 0 4.6170803  
4.60180825 1.4 0 1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 4.6170803 0 4.18082504 0 2.124  
0 3.190704 0 4.13201201 2.417 0 2.1405 0 4.18200159 4.18162000 4.17041837 0 3.505858 1.27 0 0 0 0 0  
0 0 0 0 0 0 4.2070017 4.60021420 2.1319 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 4.11001819 4.60031700  
2.2213 0 1.62 0 2.4241 1.27 1.27 0 0 0 0 0 0 0 0 0 0 4.22081303 4.14226007 4.4080607 1.19 0 1.62 0  
4.18162000 4.17046018 3.82504 0 1.56 0 4.6170803 4.60180825 1.4 0 1.61 0 4.57181620 4.170460  
4.18082504 0 1.56 0 2.9258 1.27 0 0 0 0 0 0 0 0 0 0 4.22081303 4.14226022 4.8031907 0 1.62 0  
4.18162000 4.17046018 3.82504 0 1.56 0 4.6170803 4.60180825 1.4 1.27 0 0 0 0 0 0 0 0 0 0 0 0 1.27 1.27 0  
0 0 0 0 0 0 0 0 0 0 4.8055718 1.22 0 2.6262 0 3.915837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 1.39 0  
4.66170400 2.1904 0 1.0 0 4.19100813 3.190417 0 4.22081303 2.1422 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.17141419 0 1.62 0 4.19106383 3.105758 1.27 0 0 0 0 0 0 0 0 0 0 0 0 1.39 0 4.67040508 2.1304 0  
3.190704 0 4.5170012 1.4 0 3.51417 0 4.3081815 4.11002408 2.1306 0 3.190704 0 4.8120006 1.4 0  
4.18041104 4.2190814 1.13 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.39081200 4.6046018 4.4110402  
4.19081413 4.60051700 2.1204 0 1.62 0 4.19106369 4.17001204 4.57171414 2.1958 1.27 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 4.39081200 4.6046018 4.4110402 4.19081413 4.60051700 4.12046315 4.210574 4.18080304  
4.62451914 3.154558 0 0 1.39 0 4.64030920 2.1819 0 3.190704 0 4.18080304 0 2.18 0 3.150417 0  
4.24142017 0 4.11002414 2.2019 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.17141419 4.63190819 4.11045750  
3.665058 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.39171414 4.19630802 4.14130108 4.19120015 4.57501114  
4.6146308 4.2145058 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2.1822 0 1.62 0 1.90 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 4.18220819 2.207 0 1.62 0 1.90 1.27 1.27 0 0 0 0 0 0 0 0 0 0 3.393939 1.27 1.27 1.27 0 0 0 0 0 0 0 0  
0 0 0 1.39 0 3.700419 0 4.18021704 2.413 0 4.22080319 1.7 0 3.1303 0 4.7040806 2.719 1.27 0 0 0 0 0 0 0 0  
0 0 0 0 4.18021704 4.4136022 4.8031907 0 1.62 0 4.17141419 4.63220813 4.5146018 4.2170404  
4.13220803 4.19075758 1.27 0 0 0 0 0 0 0 0 0 0 4.18021704 4.4136007 4.4080607 1.19 0 1.62 0  
4.17141419 4.63220813 4.5146018 4.2170404 4.13070408 4.6071957 1.58 1.27 1.27 0 0 0 0 0 0 0 0 0 0  
1.39 0 4.66001102 4.20110019 1.4 0 4.15141808 4.19081413 0 2.1405 0 3.190704 0 4.22081303 2.1422  
1.27 0 0 0 0 0 0 0 0 0 0 4.15141808 4.19081413 4.60191415 0 1.62 0 4.8131957 4.18021704 4.4136007  
4.4080607 1.19 0 1.32 0 1.92 0 1.59 0 4.22081303 4.14226007 4.4080607 1.19 0 1.32 0 2.9258 1.27 0 0 0 0  
0 0 0 0 0 0 0 4.15141808 4.19081413 4.60170806 2.719 0 1.62 0 4.8131957 4.18021704 4.4136022

4.8031907 0 1.32 0 1.92 0 1.59 0 4.22081303 4.14226022 4.8031907 0 1.32 0 2.9258 1.27 0 0 0 0 0 0 0 0 0  
0 0 1.27 0 0 0 0 0 0 0 0 0 0 1.39 0 3.820419 0 3.190704 0 4.3081204 4.13180814 2.1318 0 2.1405 0  
3.190704 0 4.22081303 2.1422 0 3.1303 0 4.22070417 1.4 0 2.819 0 2.818 0 4.15110002 2.403 1.27 0 0 0 0  
0 0 0 0 0 0 0 4.17141419 4.63060414 4.12041917 4.24570545 4.43220813 4.3142260 4.22080319  
4.7442343 4.22081303 4.14226007 4.4080607 4.19446143 4.15141808 4.19081413 4.60170806 4.7194461  
4.43151418 4.8190814 4.13601914 4.15444558 1.27 1.27 0 0 0 0 0 0 0 0 0 0 3.393939 1.27 1.27 0 0 0 0  
0 0 0 0 0 0 0 1.39 0 4.66170400 2.1904 0 1.0 0 4.65141411 4.4001385 2.17 0 3.51417 0 3.190704 0  
4.2141114 1.17 0 4.12140304 0 3.1303 0 3.180419 0 2.819 0 2.1914 0 4.69001118 1.4 0 4.8130819  
4.8001111 1.24 1.27 0 0 0 0 0 0 0 0 0 0 0 4.2141114 4.17601214 4.3046021 2.17 0 1.62 0 4.65141411  
4.4001385 4.175721 4.112004 4.62690011 3.180458 1.27 1.27 0 0 0 0 0 0 0 0 0 1.39 0 4.82041920  
1.15 0 4.11140606 3.81306 1.27 0 0 0 0 0 0 0 0 0 0 4.11140606 4.8130663 4.1001808 4.2661413  
4.5080657 4.5081104 4.13001204 4.62451114 4.6060813 4.6631923 3.194531 0 4.11042104 4.11621114  
4.6060813 4.6637277 3.697858 1.27 1.27 0 0 0 0 0 0 0 0 0 0 4.15002018 2.403 0 1.62 0 4.19106365  
4.14141104 4.138500 3.175758 1.27 0 0 0 0 0 0 0 0 0 0 0 4.15002018 4.4036318 4.4195769 4.111804  
1.58 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 4.2001321 4.186022 4.8031907 0 1.62 0 4.6170803 4.60180825 1.4 0  
1.56 0 4.18162000 4.17046018 3.82504 1.27 0 0 0 0 0 0 0 0 0 0 4.2001321 4.186007 4.4080607 1.19 0  
1.62 0 4.6170803 4.60180825 1.4 0 1.56 0 4.18162000 4.17046018 3.82504 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.2001321 2.18 0 1.62 0 4.19106366 4.132100 4.18571714 3.141931 0 4.22080319 4.7620200 4.13210018  
4.60220803 3.190731 0 4.7040806 4.7196202 4.132100 4.18600704 4.8060719 1.58 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 3.214110 1.62 0 4.2001321 4.186022 4.8031907 1.27 0 0 0 0 0 0 0 0 0 0 4.2001321 4.186315  
4.21057 1.58 1.27 0 0 0 0 0 0 0 0 0 0 4.2001321 4.186301 4.8130357 4.50336520 4.19191413  
4.59913450 1.31 0 4.14136002 4.132100 4.18600211 4.8021058 1.27 0 0 0 0 0 0 0 0 0 0 1.27 0 0 0 0 0  
0 0 0 0 0 1.39 0 2.7213 0 3.190704 0 4.5201302 4.19081413 0 4.22070417 1.4 0 3.241420 0 4.2170400  
2.1904 0 3.190704 0 4.2001321 4.186363 1.63 1.27 0 0 0 0 0 0 0 0 0 0 4.2001321 4.186301 4.8130357  
4.50337404 4.24345031 0 4.14136010 4.4246015 4.17041818 1.58 1.27 0 0 0 0 0 0 0 0 0 0 4.2001321  
4.186305 4.14022018 4.60180419 2.5758 1.27 0 0 0 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 0 0 0 0 1.39 0  
4.66170400 2.1904 0 4.8131520 1.19 0 4.5080411 2.318 0 3.1303 0 4.1201919 3.141318 0 3.51417 0  
4.6170803 0 3.1303 0 4.5141319 0 4.18082504 0 4.8131520 1.19 1.27 0 0 0 0 0 0 0 0 0 0 0 4.6170803  
4.60180825 4.4601100 3.104110 1.62 0 4.19106375 4.10411 4.57171414 2.1931 0 4.19042319 4.62507017  
2.803 0 4.18082504 3.375058 1.27 0 0 0 0 0 0 0 0 0 0 4.6170803 4.60180825 4.4600413 3.191724 0  
1.62 0 4.19106368 4.13191724 4.57171414 2.1931 0 4.22080319 4.7629558 1.27 0 0 0 0 0 0 0 0 0 0  
4.5141319 4.60180825 4.4601100 3.10411 0 1.62 0 4.19106375 4.10411 4.57171414 2.1931 0 4.19042319  
4.62506914 2.1319 0 4.18082504 3.375058 1.27 0 0 0 0 0 0 0 0 0 0 4.5141319 4.60180825 4.4600413  
3.191724 0 1.62 0 4.19106368 4.13191724 4.57171414 2.1931 0 4.22080319 4.7629558 1.27 0 0 0 0 0 0 0  
0 0 0 0 4.30920 4.18196001 4.20191914 1.13 0 1.62 0 4.19106365 4.20191914 4.13571714 3.141931 0  
4.19042319 4.62456403 4.9201819 0 4.70170803 0 1.55 0 4.69141319 2.4531 0 4.2141212 4.130362  
4.30920 4.18196006 4.17080360 4.130360 4.5141319 1.58 1.27 1.27 0 0 0 0 0 0 0 0 0 0 1.39 0  
4.66170400 2.1904 0 3.190704 0 4.18002104 0 3.1303 0 4.11140003 0 4.1201919 3.141318 1.27 0 0 0 0 0  
0 0 0 0 0 4.18002104 4.60820120 4.19191413 0 1.62 0 4.19106365 4.20191914 4.13571714 3.141931 0  
4.19042319 4.62458200 2.2104 0 4.82041818 4.8141345 1.31 0 4.2141212 4.130362 4.18002104  
4.60180418 4.18081413 1.31 0 4.1066250 4.22070819 3.45031 0 4.15000323 3.629531 0 4.15000324  
3.629058 1.27 0 0 0 0 0 0 0 0 0 0 4.11140003 4.60012019 3.191413 0 1.62 0 4.19106365 4.20191914  
4.13571714 3.141931 0 4.19042319 4.62457514 2.3 0 4.82041818 4.8141345 1.31 0 4.2141212 4.130362  
4.11140003 4.60180418 4.18081413 1.31 0 4.1066250 4.22070819 3.45031 0 4.15000323 3.629531 0  
4.15000324 3.629058 1.27 1.27 0 0 0 0 0 0 0 0 0 1.39 0 4.66170400 2.1904 0 4.8131520 1.19 0  
4.5080411 2.318 0 3.1303 0 4.1201919 3.141318 0 3.51417 0 4.12140304 0 3.1303 0 4.2070017 4.21904  
1.17 0 4.8131520 1.19 1.27 0 0 0 0 0 0 0 0 0 4.12140304 4.60110001 2.411 0 1.62 0 4.19106375  
4.10411 4.57171414 2.1931 0 4.19042319 4.62506813 3.190417 0 3.636363 0 4.45726768 1.45 0 3.51417 0  
4.132913 4.72131904 4.17000219 3.82104 0 4.67042104 4.11141512 3.41319 0 4.68132108 4.17141312  
4.4131929 4.13051417 0 4.75040017 4.13081306 0 4.132419 4.7081306 3.495058 1.27 0 0 0 0 0 0 0 0 0 0

0 4.12140304 4.60041319 2.1724 0 1.62 0 4.19106368 4.13191724 4.57171414 2.1958 1.27 0 0 0 0 0 0 0  
 0 0 0 0 4.18200112 4.8196001 4.20191914 1.13 0 1.62 0 4.19106365 4.20191914 4.13571714 3.141931 0  
 4.19042319 4.62457568 4.64817745 1.31 0 4.2141212 4.130362 4.18200112 4.8196012 4.14030458 1.27  
 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 0 3.454545 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.5081104 4.13001204 0 1.62 0  
 4.18081215 4.11040308 4.111406 4.63001810 4.18191708 4.13065750 4.82002104 0 4.66001321  
 4.185031 0 4.50681319 2.417 0 4.5081104 4.13001204 3.375058 1.27 0 0 0 0 0 0 0 0 0 0 0 0 2.805 0  
 4.5081104 4.13001204 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 3.454545 1.27 0 0 0 0 0 0 0 0 0 0 0 0 1.27 0 0 0 0 0  
 0 0 0 0 0 4.2070017 4.60110001 2.411 0 1.62 0 4.19106375 4.10411 4.57171414 2.1931 0 4.19042319  
 4.62506813 3.190417 0 4.18241201 4.14113218 0 2.1914 0 4.3170022 1.37 0 2.5058 1.27 0 0 0 0 0 0 0 0 0 0  
 0 0 0 4.2070017 4.60041319 2.1724 0 1.62 0 4.19106368 4.13191724 4.57171414 2.1931 0 4.22080319  
 4.7629192 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.2070017 4.60131419 4.4601100 3.10411 0 1.62 0 4.19106375  
 4.10411 4.57171414 2.1931 0 4.19042319 4.62506813 3.190417 0 4.77141904 1.37 0 2.5058 1.27 0 0 0 0 0  
 0 0 0 0 0 0 4.2070017 4.60131419 1.4 0 1.62 0 4.19106368 4.13191724 4.57171414 2.1931 0 4.22080319  
 4.7629192 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.39020700 4.17600120 4.19191413 0 1.62 0 4.19106365  
 4.20191914 4.13571714 3.141931 0 4.19042319 4.62456717 2.22 0 4.66070017 4.21904 3.174531 0  
 4.2141212 4.130362 4.3170022 4.60020700 2.1758 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.17040517 4.4180760  
 4.1201919 2.1413 0 1.62 0 4.19106365 4.20191914 4.13571714 3.141931 0 4.19042319 4.62458104  
 4.5170418 3.74531 0 4.2141212 4.130362 4.17040517 4.4180760 4.2001321 3.1831 0 4.1066250  
 4.22070819 3.45031 0 4.15000323 3.629531 0 4.15000324 3.629058 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
 4.20130314 4.60012019 3.191413 0 1.62 0 4.19106365 4.20191914 4.13571714 3.141931 0 4.19042319  
 4.62458413 4.3144531 0 4.2141212 4.130362 4.20130314 4.60110018 2.1931 0 4.1066250 4.22070819  
 3.45031 0 4.15000323 3.629531 0 4.15000324 3.629058 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.2141319  
 4.17141160 4.5170012 1.4 0 1.62 0 4.19106369 4.17001204 4.57171414 2.1958 1.27 1.27 0 0 0 0 0 0 0 0 0  
 0 0 0 4.12140304 4.60110001 4.4116315 4.21057 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 4.12140304 4.60041319  
 4.17246315 4.21057 1.58 1.27 0 0 0 0 0 0 0 0 0 0 4.18200112 4.8196001 4.20191914 4.13631500  
 4.2105758 1.27 0 0 0 0 0 0 0 0 0 0 1.39 0 4.2170400 4.19046012 4.4132057 4.17141419 1.58 1.27 1.27 0  
 0 0 0 0 0 0 0 0 0 0 0 1.39 0 3.812013 0 3.190704 0 4.19100813 3.190417 0 4.12000813 0 4.11141415 1.27 0  
 0 0 0 0 0 0 0 0 0 0 0 4.17141419 4.63120008 4.13111414 3.155758 1.27 1.27 0 0 0 0 0 0 0 4.6111401 2.11  
 0 4.18220819 2.207 1.27 0 0 0 0 0 0 0 0 4.18220819 2.207 0 1.62 0 1.91 1.27 1.27 1.27 0 0 0 0 0 0 0  
 4.18041920 4.15601214 4.3045718 4.22081902 2.758 1.27 0 0 0 0 0 0 0 0 1.27 0 0 0 0 4.8055704  
 4.13191700 3.130204 0 2.6262 0 3.945837 1.27 0 0 0 0 0 0 0 4.39110013 4.6200006 3.45758 1.27 0 0 0  
 0 0 0 0 3.30405 0 4.6041960 4.18200117 4.130604 4.18600517 4.14126020 4.18041757 2.5837 1.27 0 0 0 0  
 0 0 0 0 0 0 0 4.18200117 4.130604 1.18 0 1.62 0 2.4241 1.27 0 0 0 0 0 0 0 0 0 0 4.13201201 4.4176014  
 4.5601820 4.1170013 3.60418 0 1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 3.190704 0  
 4.13201201 2.417 0 2.1405 0 4.18200117 4.130604 2.1837 0 3.505858 1.27 1.27 0 0 0 0 0 0 0 0 0 0  
 3.51417 0 1.8 0 2.813 0 4.17001306 4.4571320 4.12010417 4.60140560 4.18200117 4.130604 3.185837  
 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.11142204 4.17600114 3.201303 0 1.62 0 4.8131957 4.8131520  
 4.19570550 4.68131904 1.17 0 3.190704 0 4.11142204 1.17 0 4.1142013 1.3 0 2.1405 0 4.18200117  
 4.130604 0 2.4308 0 1.61 0 3.914437 0 3.505858 1.27 0 0 0 0 0 0 0 0 0 0 0 4.20151504 4.17600114  
 3.201303 0 1.62 0 4.8131957 4.8131520 4.19570550 4.68131904 1.17 0 3.190704 0 4.20151504 1.17 0  
 4.1142013 1.3 0 2.1405 0 4.18200117 4.130604 0 2.4308 0 1.61 0 3.914437 0 3.505858 1.27 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 4.18200117 4.130604 4.18630015 4.15041303 4.57571114 4.22041760 4.1142013 2.331 0  
 4.20151504 4.17600114 4.20130358 1.58 1.27 1.27 0 0 0 0 0 0 0 0 0 0 4.17041920 2.1713 0 4.18200117  
 4.130604 1.18 1.27 1.27 0 0 0 0 0 0 0 3.30405 0 4.12000813 3.575837 1.27 0 0 0 0 0 0 0 0 0  
 4.15170813 4.19575057 3.151958 0 2.6262 0 4.84130821 4.4171800 1.11 0 3.820419 0 1.62 0 4.66141215  
 4.20190417 2.5058 1.27 1.27 0 0 0 0 0 0 0 0 0 1.39 0 4.81040003 0 3.190704 0 4.5081104 1.27 0 0 0  
 0 0 0 0 0 0 0 4.6041919 3.42319 0 1.62 0 4.8131520 4.19575068 4.13190417 0 4.5081104 4.13001204 0  
 2.1405 0 3.190704 0 4.63192319 0 4.5081104 0 4.2141319 4.81308 2.1306 0 4.18081306 2.1104 0  
 4.2070017 4.21904 2.1718 0 3.150417 0 4.11081304 0 4.57841308 4.2140304 0 4.66070017 4.21904  
 4.17185837 0 2.5058 1.27 0 0 0 0 0 0 0 0 0 0 0 4.39220819 1.7 0 4.14150413 4.57500011 4.15070063



















0 0 0 0 1.27 0 0 0 0 4.8055704 4.13191700 3.130204 0 2.6262 0 4.91975837 1.27 0 0 0 0 0 0 0 0 4.22070811  
3.45795 0 1.34 0 3.925837 1.27 0 0 0 0 0 0 0 0 0 0 0 4.8131924 0 1.62 0 4.8131957 4.8131520 4.19575090  
0 3.51417 0 4.12000813 4.60120413 2.2031 0 1.91 0 2.1914 0 4.2141319 4.8132004 1.37 0 3.505858 1.27 0  
0 0 0 0 0 0 0 0 0 0 4.8055708 3.131924 0 2.6262 0 3.905837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4.1170400  
1.10 1.27 0 0 0 0 0 0 0 0 0 0 0 4.8055708 3.131924 0 2.6262 0 3.915837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.12000813 3.925758 1.27 0 0 0 0 4.8055704 4.13191700 3.130204 0 2.6262 0 4.91985837 1.27 0 0 0 0 0 0  
0 0 4.22070811 3.45791 0 1.33 0 3.925837 1.27 0 0 0 0 0 0 0 0 0 0 4.8131919 0 1.62 0 4.8131957  
4.8131520 4.19575090 0 3.51417 0 4.12000813 4.60120413 2.2031 0 1.91 0 2.1914 0 4.2141319 4.8132004  
1.37 0 3.505858 1.27 0 0 0 0 0 0 0 0 0 0 4.8055708 3.131919 0 2.6262 0 3.905837 1.27 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 4.1170400 1.10 1.27 0 0 0 0 0 0 0 0 0 0 4.8055708 3.131919 0 2.6262 0 3.915837 1.27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 4.12000813 4.91985758 1.27 0 0 0 0 4.8055704 4.13191700 3.130204 0 2.6262 0  
4.91995837 1.27 0 0 0 0 0 0 0 4.15170813 4.19575050 4.50291329 4.13830704 0 4.5141111 4.14220813  
1.6 0 2.818 0 2.13 0 4.4230012 3.151104 0 4.12001515 3.81306 0 3.51417 0 3.190704 0 4.5141319 0  
4.8130304 4.23372913 1.27 0 0 0 0 0 0 0 1.90 0 1.0 1.27 0 0 0 0 0 0 0 1.91 0 1.1 1.27 0 0 0 0 0 0 0 1.92  
0 1.2 1.27 0 0 0 0 0 0 0 1.93 0 1.3 1.27 0 0 0 0 0 0 0 1.94 0 1.4 1.27 0 0 0 0 0 0 0 1.95 0 1.5 1.27 0 0 0 0  
0 0 0 0 1.96 0 1.6 1.27 0 0 0 0 0 0 0 1.97 0 1.7 1.27 0 0 0 0 0 0 0 1.98 0 1.8 1.27 0 0 0 0 0 0 0 1.99 0 1.9  
1.27 0 0 0 0 0 0 0 0 2.9190 0 1.10 1.27 0 0 0 0 0 0 0 2.9191 0 1.11 1.27 0 0 0 0 0 0 0 2.9192 0 1.12 1.27 0  
0 0 0 0 0 0 2.9193 0 1.13 1.27 0 0 0 0 0 0 0 2.9194 0 1.14 1.27 0 0 0 0 0 0 0 2.9195 0 1.15 1.27 0 0 0 0  
0 0 0 2.9196 0 1.16 1.27 0 0 0 0 0 0 0 2.9197 0 1.17 1.27 0 0 0 0 0 0 0 2.9198 0 1.18 1.27 0 0 0 0 0 0 0  
2.9199 0 1.19 1.27 0 0 0 0 0 0 0 2.9290 0 1.20 1.27 0 0 0 0 0 0 0 2.9291 0 1.21 1.27 0 0 0 0 0 0 0 2.9292  
0 1.22 1.27 0 0 0 0 0 0 0 2.9293 0 1.23 1.27 0 0 0 0 0 0 0 2.9294 0 1.24 1.27 0 0 0 0 0 0 0 2.9295 0 1.25  
1.27 0 0 0 0 0 0 0 0 2.9296 0 1.45 1.27 0 0 0 0 0 0 0 2.9297 0 1.31 1.27 0 0 0 0 0 0 0 2.9298 0 1.32 1.27 0  
0 0 0 0 0 0 2.9299 0 1.33 1.27 0 0 0 0 0 0 0 2.9390 0 1.34 1.27 0 0 0 0 0 0 0 2.9391 0 1.35 1.27 0 0 0 0  
0 0 0 2.9392 0 1.36 1.27 0 0 0 0 0 0 0 2.9393 0 1.37 1.27 0 0 0 0 0 0 0 2.9394 0 1.38 1.27 0 0 0 0 0 0 0  
2.9395 0 1.39 1.27 0 0 0 0 0 0 0 2.9396 0 1.40 1.27 0 0 0 0 0 0 0 2.9397 0 1.41 1.27 0 0 0 0 0 0 0 2.9398  
0 1.42 1.27 0 0 0 0 0 0 0 2.9399 0 1.43 1.27 0 0 0 0 0 0 0 2.9490 0 1.44 1.27 0 0 0 0 0 0 0 2.9491 0 1.45  
1.27 0 0 0 0 0 0 0 0 2.9492 0 0 1.46 1.27 0 0 0 0 0 0 0 2.9493 0 1.47 1.27 0 0 0 0 0 0 0 2.9494 0 0 1.48  
1.27 0 0 0 0 0 0 0 0 2.9495 0 1.49 1.27 0 0 0 0 0 0 0 2.9496 0 1.50 1.27 0 0 0 0 0 0 0 2.9497 0 0 1.51 1.27  
0 0 0 0 0 0 0 0 2.9498 0 1.52 1.27 0 0 0 0 0 0 0 2.9499 0 1.53 1.27 0 0 0 0 0 0 0 2.9590 0 1.54 1.27 0 0 0 0  
0 0 0 0 2.9591 0 1.55 1.27 0 0 0 0 0 0 0 2.9592 0 1.56 1.27 0 0 0 0 0 0 0 2.9593 0 1.57 1.27 0 0 0 0 0 0 0  
2.9594 0 1.58 1.27 0 0 0 0 0 0 0 2.9595 0 1.59 1.27 0 0 0 0 0 0 0 2.9596 0 1.60 1.27 0 0 0 0 0 0 0 2.9597  
0 1.61 1.27 0 0 0 0 0 0 0 2.9598 0 1.62 1.27 0 0 0 0 0 0 0 2.9599 0 1.63 1.27 0 0 0 0 0 0 0 2.9690 0 1.64  
1.27 0 0 0 0 0 0 0 0 2.9691 0 1.65 1.27 0 0 0 0 0 0 0 2.9692 0 1.66 1.27 0 0 0 0 0 0 0 2.9693 0 1.67 1.27 0  
0 0 0 0 0 0 2.9694 0 1.68 1.27 0 0 0 0 0 0 0 2.9695 0 1.69 1.27 0 0 0 0 0 0 0 2.9696 0 1.70 1.27 0 0 0 0 0  
0 0 0 2.9697 0 1.71 1.27 0 0 0 0 0 0 0 2.9698 0 1.72 1.27 0 0 0 0 0 0 0 2.9699 0 1.73 1.27 0 0 0 0 0 0  
2.9790 0 1.74 1.27 0 0 0 0 0 0 0 2.9791 0 1.75 1.27 0 0 0 0 0 0 0 2.9792 0 1.76 1.27 0 0 0 0 0 0 0 2.9793  
0 1.77 1.27 0 0 0 0 0 0 0 2.9794 0 1.78 1.27 0 0 0 0 0 0 0 2.9795 0 1.79 1.27 0 0 0 0 0 0 0 2.9796 0 1.80  
1.27 0 0 0 0 0 0 0 0 2.9797 0 1.81 1.27 0 0 0 0 0 0 0 2.9798 0 1.82 1.27 0 0 0 0 0 0 0 2.9799 0 1.83 1.27 0  
0 0 0 0 0 0 2.9890 0 1.84 1.27 0 0 0 0 0 0 0 2.9891 0 1.85 1.27 0 0 0 0 0 0 0 2.9892 0 1.86 1.27 0 0 0 0 0  
0 0 0 2.9893 0 1.87 1.27 0 0 0 0 0 0 0 2.9894 0 1.88 1.27 0 0 0 0 0 0 0 2.9895 0 1.89 1.27 0 0 0 0 0 0  
2.9896 0 1.90 1.27 0 0 0 0 0 0 0 2.9897 0 1.91 1.27 0 0 0 0 0 0 0 2.9898 0 1.92 1.27 0 0 0 0 0 0 0 2.9899  
0 1.93 1.27 0 0 0 0 0 0 0 2.9990 0 1.94 1.27 0 0 0 0 0 0 0 2.9991 0 1.95 1.27 0 0 0 0 0 0 0 2.9992 0 1.96  
1.27 0 0 0 0 0 0 0 2.9993 0 1.97 1.27 0 0 0 0 0 0 0 2.9994 0 1.98 1.27 0 0 0 0 0 0 0 2.9995 0 1.99 1.27 0  
0 0 0 0 0 0 2.9996 0 2.2929 1.27 0 0 0 0 0 0 0 4.50505058 1.27 0 0 0 0 0 0 0 4.6170803 4.60180825 1.4  
0 1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 3.190704 0 4.6170803 0 4.18082504 0  
4.57132012 3.10417 0 2.1405 0 4.18162000 3.170418 0 4.111413 1.6 0 3.141304 0 4.18080304 2.5837 0  
3.505858 1.27 0 0 0 0 0 0 0 4.18162000 4.17046018 3.82504 0 1.62 0 4.8131957 4.8131520 4.19575068  
4.13190417 0 3.190704 0 4.15082304 1.11 0 4.11041306 2.1907 0 2.1405 0 4.4000207 0 4.18200159  
4.18162000 3.170437 0 3.505858 1.27 0 0 0 0 0 0 4.5141319 4.60180825 1.4 0 1.62 0 4.8131957  
4.8131520 4.19575068 4.13190417 0 3.190704 0 4.5141319 0 4.18082504 0 3.51417 0 3.190704 0

4.18241201 4.14111837 0 3.505858 1.27 0 0 0 0 0 0 0 0 4.5081104 4.60130012 1.4 0 1.62 0 4.8131520  
4.19575068 4.13190417 0 3.190704 0 4.13001204 0 2.1405 0 3.190704 0 4.2141305 4.8062017 4.190814  
1.13 0 4.5081104 0 2.1914 0 4.18002104 0 4.57631923 3.195837 0 2.5058 1.27 1.27 0 0 0 0 0 0 0  
4.2141305 4.8062017 4.190814 2.1318 0 1.62 0 2.4241 1.27 0 0 0 0 0 0 0 3.51417 0 1.8 0 2.813 0  
4.17001306 4.4570617 4.8036018 3.82504 0 1.56 0 4.6170803 4.60180825 3.45837 1.27 0 0 0 0 0 0 0 0 0  
0 0 4.15170813 4.19570550 4.66141305 4.8062017 4.190814 1.13 0 3.51417 0 4.18162000 2.1704 0  
4.43086191 3.445058 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.18241201 4.14111608 4.13030423 0 1.62 0 4.8131957  
4.8131520 4.19575068 4.13190417 0 4.18241201 2.1411 0 4.8130304 1.23 0 2.5790 0 2.1914 0 4.99965837  
0 3.505858 1.27 0 0 0 0 0 0 0 0 0 0 0 4.18162000 4.17046002 4.14111417 4.60081303 2.423 0 1.62 0 1.90  
0 1.61 0 4.8131957 4.91969797 4.97929196 0 1.32 0 4.57182412 4.1141160 4.8130304 1.23 0 1.61 0  
3.915858 1.27 0 0 0 0 0 0 0 0 0 0 0 4.5141319 4.60021411 4.14176008 4.13030423 0 1.62 0 4.91969797  
4.97929196 0 1.59 0 4.8131957 4.91969797 4.97929196 0 1.32 0 4.57182412 4.1141160 4.8130304 1.23 0  
1.61 0 3.915858 1.27 0 0 0 0 0 0 0 0 0 0 0 4.2141305 4.8062017 4.190814 4.13186300 4.15150413  
4.3575718 4.16200017 4.4600214 4.11141760 4.8130304 2.2331 0 4.18241201 4.14116008 4.13030423  
1.31 0 4.5141319 4.60021411 4.14176008 4.13030423 2.5858 1.27 0 0 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 0  
4.18002104 4.60021413 4.5080620 4.17001908 4.14136019 4.14600508 4.11049199 4.57050811  
4.4601300 3.120431 0 4.6170803 4.60180825 2.431 0 4.18162000 4.17046018 4.8250431 0 4.5141319  
4.60180825 2.431 0 4.2141305 4.8062017 4.190814 3.131858 1.27 0 0 0 0 0 0 0 4.15170813 4.19570550  
4.66141305 4.8062017 4.190814 1.13 0 4.18002104 1.3 0 2.1914 0 4.43050811 4.4601300 4.12044450 1.58  
1.27 0 0 0 0 4.8055704 4.13191700 3.130204 0 2.6262 0 4.92915837 1.27 0 0 0 0 0 0 0 4.22070811  
3.45795 0 1.34 0 3.925837 1.27 0 0 0 0 0 0 0 0 0 0 4.8131924 1.18 0 1.62 0 4.8131957 4.8131520  
4.19575090 0 3.51417 0 4.12000813 4.60120413 2.2031 0 1.91 0 2.1914 0 4.2141319 4.8132004 1.37 0  
3.505858 1.27 0 0 0 0 0 0 0 0 0 0 4.8055708 4.13192418 0 2.6262 0 3.905837 1.27 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 4.1170400 1.10 1.27 0 0 0 0 0 0 0 0 0 0 0 4.8055708 4.13192418 0 2.6262 0 3.915837 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 1.39 0 4.76000813 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.8130819 4.80001160 1.0 0 1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 4.8130819 3.800110  
1.0 0 2.5700 0 1.34 0 3.15837 0 3.505858 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.8130819 4.80001160 1.1 0  
1.62 0 4.8131957 4.8131520 4.19575068 4.13190417 0 4.8130819 3.800110 1.1 0 2.5701 0 1.33 0 3.5837 0  
3.505858 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.20180417 4.60081904 4.17001908 3.141318 0 1.62 0  
4.8131957 4.8131520 4.19575068 4.13190417 0 4.13201201 2.417 0 2.1405 0 4.30308 4.19081413 2.11 0  
4.20180417 0 4.8190417 4.190814 3.131837 0 3.505858 1.27 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 3.51812 0  
1.62 0 4.79171423 4.8120011 4.79170812 4.4698276 4.57081308 4.19080011 3.600031 0 4.8130819  
4.80001160 2.131 0 4.20180417 4.60081904 4.17001908 4.14131858 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.5181263 4.17201357 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.2141417 2.318 0 1.62 0 4.15170415  
4.170460 4.2141417 4.3081300 4.19041892 4.91570518 4.12631517 4.8120418 1.58 1.27 1.27 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 3.50806 0 1.62 0 4.15111963 4.5080620 4.17045758 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
2.23 0 1.62 0 4.5080663 4.30360 4.18200115 4.11141957 4.91919131 0 4.15171409 4.4021908 4.14136245  
4.93034558 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 3.1308 0 1.62 0 4.69201302 4.64130812 4.190814  
4.13570508 2.631 0 4.130812 4.190492 2.9131 0 4.5170012 4.4186211 4.4135702 4.14141703 3.185831 0  
4.17041504 4.196269 4.111804 1.58 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.15111963 4.18071422 2.5758 1.27  
0 0 0 0 4.8055704 4.13191700 3.130204 0 2.6262 0 4.92925837 1.27 0 0 0 0 0 0 0 4.22070811 3.45795 0  
1.34 0 3.925837 1.27 0 0 0 0 0 0 0 0 0 0 4.8131924 3.189292 0 1.62 0 4.8131957 4.8131520 4.19575090  
0 3.51417 0 4.12000813 4.60120413 2.2031 0 1.91 0 2.1914 0 4.2141319 4.8132004 1.37 0 3.505858 1.27 0  
0 0 0 0 0 0 0 0 0 4.8055708 4.13192418 2.9292 0 2.6262 0 3.905837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.1170400 1.10 1.27 0 0 0 0 0 0 0 0 0 0 4.8055708 4.13192418 2.9292 0 2.6262 0 3.915837 1.27 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 4.12000813 4.92925758 1.27 0 0 0 0 4.8055704 4.13191700 3.130204 0 2.6262 0  
4.92935837 1.27 0 0 0 0 0 0 0 4.22070811 3.45795 0 1.34 0 3.925837 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.8131924 3.189293 0 1.62 0 4.8131957 4.8131520 4.19575090 0 3.51417 0 4.12000813 4.60120413  
2.2031 0 1.91 0 2.1914 0 4.2141319 4.8132004 1.37 0 3.505858 1.27 0 0 0 0 0 0 0 0 0 4.8055708  
4.13192418 2.9293 0 2.6262 0 3.905837 1.27 0 0 0 0 0 0 0 0 0 0 0 0 4.1170400 1.10 1.27 0 0 0 0 0























































3.130802 0 4.64110423 4.130304 1.17 0 4.66141415 3.41731 0 4.64120011 2.402 0 4.81218 0 2.1914 0  
4.15171421 3.80304 0 4.20180417 1.18 0 4.22081907 0 1.0 0 4.17140120 2.1819 0 3.180419 0 2.1405 0  
4.19141411 1.18 0 3.51417 0 4.1141907 0 4.15171405 4.4181808 4.14130011 0 3.1303 0 4.4032002  
4.190814 3.130011 0 4.15201715 4.14180418 1.63 1.27 1.27 2.3939 0 4.69040019 4.20170418 1.27  
4.71041704 0 3.1704 0 3.190704 0 4.5040019 4.20170418 0 3.1303 0 4.5201302 4.19081413 4.110819  
3.80418 0 4.14050504 3.170403 0 2.124 0 4.64120011 3.40237 1.27 1.27 3.393939 0 2.9063 0 4.72120006  
1.4 0 4.70041304 4.17001914 1.17 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.70041304  
4.17001904 1.18 0 4.8120006 2.418 0 4.22070802 1.7 0 3.1704 0 4.17041620 4.8170403 0 3.51417 0  
3.190704 0 3.201804 0 2.1405 0 4.14190704 1.17 0 4.5040019 4.20170418 0 2.813 0 3.190704 0 4.151511  
4.8020019 4.8141331 0 4.15001719 4.8022011 4.171124 0 3.190704 0 4.79171406 4.17001212 3.81306 0  
4.68130608 2.1304 0 4.57781519 3.81413 0 3.925863 1.27 1.27 3.393939 0 2.9163 0 4.72120006 1.4 0  
4.66141215 4.8110417 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.66141215 4.8110418 0  
4.12201119 4.8151104 0 4.8120006 2.418 0 4.8131914 0 1.0 0 4.18081306 2.1104 0 4.5081104 0 2.1417 0  
4.15170418 4.4131900 4.19081413 1.63 1.27 1.27 3.393939 0 2.9263 0 4.79171406 4.17001212 3.81306 0  
4.68130608 2.1304 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 1.64 0 4.2141215 4.17040704  
4.13180821 1.4 0 4.4130608 2.1304 0 3.51417 0 4.15171406 4.17001212 3.81306 0 4.19001810 2.1831 0  
4.11042104 4.17000608 2.1306 0 3.190704 0 4.8120006 2.418 0 4.6041304 4.17001904 1.3 0 2.124 0  
3.190704 0 4.72120006 1.4 0 4.70041304 4.17001914 2.1763 1.27 1.27 3.393939 0 2.9363 0 4.66141812  
4.14111406 1.24 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.83141411 1.18 0 3.1303 0  
4.5201302 4.19081413 4.110819 3.80418 0 3.51417 0 4.2141812 4.14111406 4.8020011 0 4.17041804  
4.170207 0 3.1303 0 4.18081220 4.11001908 4.14131863 0 3.660013 0 2.104 0 4.20180403 0 2.813 0  
4.2141309 4.20130219 3.81413 0 4.22081907 0 4.78151908 3.141318 0 2.9031 0 2.9131 0 3.1303 0 2.9263  
1.27 1.27 3.393939 0 2.9463 0 4.82041620 4.4131908 2.11 0 4.75001306 4.20000604 0 4.70041304  
4.17001908 2.1413 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.83141411 1.18 0 3.1303 0  
4.18041620 4.4130204 1.18 0 2.1405 0 4.11001306 4.20000604 1.63 0 4.56568600 4.17130813 4.6565637  
0 4.8307018 0 4.19141411 0 4.12201819 0 4.5081119 2.417 0 3.142019 0 4.15171405 3.1304 0 4.2141319  
4.4131963 0 1.64 0 4.20180417 0 0 4.63192319 0 0 4.5081104 0 4.12201819 0 4.4230818 1.19 0  
4.22081907 2.813 0 3.190704 0 4.22141710 3.81306 0 4.3081704 4.2191417 2.2431 0 4.11081819 3.81306  
0 4.84130802 3.140304 0 4.2070017 4.21904 2.1718 0 4.57141304 0 3.150417 0 4.11081304 2.5863 1.27  
1.27 3.393939 0 2.9563 0 4.83042319 0 4.68030819 2.1417 1.27 4.56566704 4.18021708 4.15190814  
4.13565637 0 1.64 0 4.1200811 4.19590813 0 4.19042319 0 4.4030819 2.1417 0 3.51417 0 4.22170819  
3.81306 0 3.1303 0 4.4030819 3.81306 0 4.19042319 0 4.5081104 2.1863 1.27 1.27 3.393939 0 2.9663 0  
4.70041304 4.17001904 0 4.66141212 3.1303 0 4.76001917 2.823 0 4.79112006 2.813 0 3.51417 0  
3.190704 0 4.79171406 4.17001212 3.81306 0 4.68130608 2.1304 1.27 4.56566704 4.18021708  
4.15190814 4.13565637 0 4.66170400 3.190418 0 4.2141212 3.1303 0 4.12001917 2.823 0 4.15112006  
3.81318 0 2.1914 0 4.4231904 2.1303 0 3.190704 0 4.2001500 4.1081108 4.19080418 0 2.1405 0 3.190704  
0 4.79171406 4.17001212 3.81306 0 4.68130608 3.130463 1.27 1.27 3.393939 0 2.9763 0 4.68230819 0  
4.79171406 3.170012 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.68230819 1.18 0 3.190704  
0 4.64120011 2.402 0 4.151511 4.8020019 4.8141363 1.27 1.27 3.393939 0 2.9863 0 2.9367 0 3.666467 0  
3.666476 0 4.68130608 2.1304 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 1.64 0 2.9367 0  
4.66646732 3.666476 0 4.4130608 2.1304 0 3.51417 0 4.2141215 4.20190417 4.59000803 2.403 0  
4.3041808 2.613 0 3.1303 0 4.12001320 4.5000219 4.20170813 2.663 1.27 1.27 3.393939 0 2.9963 0  
4.80200013 3.192012 0 4.66081702 3.200819 0 4.80081810 2.819 0 4.86141710 4.65041302 1.7 1.27  
4.56566704 4.18021708 4.15190814 4.13565637 0 1.64 0 4.22141710 4.1041302 1.7 0 3.51417 0  
4.2170400 4.19081306 0 3.1303 0 4.18081220 4.11001908 2.1306 0 4.16200013 3.192012 0 4.2081702  
4.20081918 0 4.20180813 1.6 0 4.80081810 3.81963 1.27 1.27 3.393939 0 3.919063 0 4.66110018  
4.18080200 1.11 0 4.66081702 3.200819 0 4.82020704 4.12671700 1.22 0 4.86141710 4.65041302 1.7 1.27  
4.56566704 4.18021708 4.15190814 4.13565637 0 1.64 0 4.22141710 4.1041302 1.7 0 3.51417 0  
4.3041808 4.6130813 1.6 0 3.1303 0 4.18081220 4.11001908 2.1306 0 4.2110018 4.18080200 1.11 0  
4.2081702 4.20081918 0 4.20180813 1.6 0 4.82020704 4.12671700 2.2263 1.27 1.27 3.393939 0 3.919163

0 4.66141812 4.14111406 4.8020011 0 4.82081220 4.11001908 2.1413 0 2.124 0 4.83042319 1.27  
4.56566704 4.18021708 4.15190814 4.13565637 0 4.82081220 4.11001904 1.18 0 4.2141812 4.14111406  
4.8020011 0 4.15070413 4.14120413 1.0 0 4.20180813 1.6 0 4.19042319 4.59010018 2.403 0 4.8131520  
2.1918 0 3.1303 0 4.14201915 4.20191863 1.27 1.27 3.393939 0 3.919263 0 4.75001706 1.4 0 4.75001306  
4.20000604 0 4.76140304 1.11 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.72131904  
4.6170019 2.418 0 1.0 0 4.11001706 1.4 0 4.11001306 4.20000604 0 4.12140304 1.11 0 3.51417 0  
4.21001708 3.142018 0 4.11001306 4.20000604 0 4.15171402 4.4181808 2.1306 0 4.19001810 2.1863 1.27  
1.27 3.393939 0 3.919363 0 4.67001900 0 4.82021708 2.1519 0 2.124 0 4.66201819 2.1412 0 4.82191720  
4.2192017 2.418 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.70041304 4.17001904 1.18 0  
4.3001900 0 4.18021708 3.151918 0 4.20180813 1.6 0 4.2201819 2.1412 0 4.3001900 0 4.18191720  
4.2192017 3.41863 1.27 1.27 3.393939 0 3.919463 0 4.70041304 4.17001904 0 4.65081300 2.1724 0  
4.84130816 2.2004 0 4.82191708 3.130618 0 2.1914 0 1.0 0 4.2201819 4.14126319 2.2319 0 4.69081104  
1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.66170400 3.190418 0 4.20130816 2.2004 0  
4.1081300 2.1724 0 4.18191708 3.130618 0 3.1303 0 4.18002104 1.18 0 4.19070412 0 2.1914 0 1.0 0  
4.2201819 2.1412 0 0 4.63192319 0 0 4.5081104 1.63 1.27 1.27 3.393939 0 3.919663 0 4.66141321  
3.41719 0 4.63142019 1.1 0 4.69081104 0 2.1914 0 4.65081300 4.17247820 1.19 0 4.69081104 0  
4.57682319 4.4131808 2.1413 0 2.1405 0 4.78151908 2.1413 0 3.919258 1.27 4.56566704 4.18021708  
4.15190814 4.13565637 0 4.66141321 4.4171918 0 0 4.63142019 1.1 0 0 4.5081104 1.18 0 2.1914 0 0  
4.1081300 4.17241420 1.19 0 0 4.5081104 2.1831 0 4.4231904 4.13030813 1.6 0 3.190704 0 4.5201302  
4.19081413 4.110819 1.24 0 2.1405 0 3.190704 0 4.75001706 1.4 0 4.75001306 4.20000604 0 4.76140304  
2.1163 1.27 1.27 3.393939 0 3.919763 0 4.83042319 0 4.67001900 4.1001804 1.27 4.56566704 4.18021708  
4.15190814 4.13565637 0 4.76001300 3.60418 0 3.1303 0 4.12001308 4.15201100 3.190418 0 4.19042319  
0 4.3001900 4.1001804 2.1863 1.27 1.27 3.393939 0 3.919863 0 4.79171423 4.8120011 0 4.79170812 1.4 0  
4.77201201 3.41718 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.66001102 4.20110019  
2.418 0 3.1303 0 4.130011 4.24250418 0 4.15171423 4.8120011 0 4.15170812 1.4 0 4.13201201 4.4171863  
1.27 1.27 3.393939 0 3.919963 0 4.66141305 4.8062017 4.190814 2.1318 0 3.51417 0 4.76140304 0 1.92  
1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.79171421 4.8030418 0 4.2141305 4.8062017  
4.190814 1.13 0 4.18041919 4.8130618 0 3.51417 0 3.190704 0 4.79171406 4.17001212 3.81306 0  
4.68130608 2.1304 0 4.57781519 3.81413 0 3.925863 1.27 1.27 3.393939 0 3.929163 0 4.79171423  
4.8120011 0 4.79170812 2.418 0 4.64130812 4.191417 1.27 4.56566704 4.18021708 4.15190814  
4.13565637 0 4.64130812 4.190418 0 3.190704 0 4.1040700 4.21081417 0 3.1303 0 4.3081819 4.17080120  
4.19081413 0 2.1405 0 4.15171423 4.8120011 0 4.15170812 1.4 0 4.13201201 4.4171863 1.27 1.27  
3.393939 0 3.929263 0 4.76001320 2.11 0 4.83201708 2.1306 0 4.66141215 4.11041904 0 4.69081308  
2.1904 0 4.82190019 1.4 0 4.76000207 3.81304 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0  
4.64111114 2.2218 0 3.190704 0 4.20180417 0 2.1914 0 4.12001320 4.111124 0 4.2141318 4.19172002  
1.19 0 1.0 0 4.83201708 4.13065902 4.14121511 3.41904 0 4.5081308 2.1904 0 4.18190019 1.4 0  
4.12000207 4.8130463 1.27 1.27 3.393939 0 3.929363 0 4.64201914 4.12001908 1.2 0 4.83201708 2.1306  
0 4.66141215 4.11041904 0 4.69081308 2.1904 0 4.82190019 1.4 0 4.76000207 3.81304 1.27 4.56566704  
4.18021708 4.15190814 4.13565637 0 4.64201914 4.12001908 4.2001111 1.24 0 4.6041304 4.17001904  
1.18 0 1.0 0 4.83201708 4.13065902 4.14121511 3.41904 0 4.5081308 2.1904 0 4.18190019 1.4 0  
4.12000207 4.8130463 1.27 1.27 3.393939 0 3.929463 0 4.79241907 2.1413 0 4.68130001 3.110403 0  
4.66141212 3.1303 0 4.75081304 0 4.72131904 4.17050002 1.4 1.27 4.56566704 4.18021708 4.15190814  
4.13565637 0 4.79171421 4.8030418 0 1.0 0 4.2141212 3.1303 0 4.11081304 0 4.8131904 4.17050002 1.4  
0 4.22081907 0 4.79241907 2.1413 0 4.2001500 4.1081108 4.19080418 1.63 1.27 1.27 3.393939 0  
3.929563 0 4.64201914 4.12001908 1.2 0 4.66141319 3.171411 0 4.68130608 4.13040417 3.81306 1.27  
4.56566704 4.18021708 4.15190814 4.13565637 0 4.83141411 1.18 0 3.1303 0 4.5201302 4.19081413  
4.110819 3.80418 0 3.51417 0 4.201914 4.12001908 1.2 0 4.2141319 3.171411 0 4.4130608 4.13040417  
4.8130663 1.27 1.27 3.393939 0 3.929663 0 4.64201914 4.12001904 1.3 0 4.67041808 2.613 0 3.1303 0  
4.83040207 4.13141114 2.624 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.69000208  
4.11081900 3.190418 0 4.201914 4.12001904 1.3 0 4.3041808 2.613 0 3.1303 0 4.19040207 4.13141114

4.6080200 1.11 0 4.3042104 4.11141512 3.41319 0 4.19001810 2.1863 1.27 1.27 3.393939 0 3.929763 0  
4.66141318 4.19172002 1.19 0 4.88142017 0 3.782213 0 4.75081306 4.20081819 2.802 0 4.75001306  
4.20000604 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.68130001 3.110418 0 4.20180417  
1.18 0 2.1914 0 4.2141318 4.19172002 1.19 0 4.19070408 1.17 0 3.142213 0 4.11081306 4.20081819 2.802  
0 4.11001306 4.20000604 2.1863 1.27 1.27 3.393939 0 3.929863 0 4.66141321 3.41719 0 4.83042319 0  
4.68130214 4.3081306 0 2.1914 0 1.0 0 4.66141321 4.4170604 2.1319 0 4.72120006 1.4 0 4.57691714 1.12  
0 4.78151908 2.1413 0 3.919258 1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 4.66141321  
4.4171918 0 4.19042319 0 4.4130214 4.3081306 1.31 0 4.3041708 3.210403 0 4.5171412 0 3.190704 0  
4.75001706 1.4 0 4.75001306 4.20000604 0 4.76140304 2.1131 0 4.8131914 0 1.0 0 4.2141321 4.4170604  
2.1319 0 4.8120006 2.463 1.27 1.27 3.393939 0 3.929963 0 4.76001907 4.4120019 3.80218 0 3.750001  
1.27 4.56566704 4.18021708 4.15190814 4.13565637 0 1.64 0 3.110001 0 4.4132108 4.17141312 3.41319  
0 3.51417 0 4.2141303 4.20021908 2.1306 0 4.12001907 4.4120019 4.8020011 0 4.4231504 4.17081204  
3.131918 0 3.1303 0 4.21081820 4.110825 4.190814 3.131863 1.27 1.27 2.3939 0 4.72131819 4.111100  
4.19081413 1.27 2.9163 0 4.56566611 3.141304 0 3.190704 0 4.17041514 4.18081914 4.17245656 1.37  
1.27 0 0 0 0 0 0 2.1807 1.27 0 0 0 3.60819 0 4.2111413 1.4 0 4.7191915 4.18373232 4.6081907 4.20016302  
4.14123267 4.78767264 4.87687067 4.68320215 4.24630608 1.19 1.27 2.9263 0 4.56567700 4.21080600  
2.1904 0 2.1914 0 3.190704 0 4.3081704 4.2191417 1.24 0 2.813 0 4.19041712 4.8130011 0 2.1417 0  
4.2141318 4.14110463 2.5656 1.27 2.9363 0 4.56561508 1.15 0 4.8131819 3.1111 0 2.5917 0 4.17041620  
4.8170412 4.4131918 4.63192319 2.5656 1.27 1.27 2.3939 0 4.75080204 3.131804 1.27 4.83070818 0  
4.15171409 3.40219 0 2.818 0 4.11080204 4.13180403 0 4.20130304 1.17 0 3.190704 0 3.767283 0  
4.75080204 3.131804 0 1.59 0 3.180404 0 3.190704 0 4.42757266 4.68778268 4.41577572 4.66687782  
2.6858 0 4.5081104 0 3.51417 0 4.3041900 4.8111863 1.27 1.27 4.18041920 4.15631524 0 1.47 1.27 1.27  
4.5171412 0 4.18041920 4.15191414 2.1118 0 4.8121514 2.1719 0 4.18041920 2.1531 0 4.5081303  
4.60150002 4.10000604 1.18 1.27 1.27 4.18041920 2.1557 1.27 0 0 0 0 4.13001204 4.62506412 4.110402  
2.5031 1.27 0 0 0 0 4.2104178 4.8141362 4.50906391 4.63905031 1.27 0 0 0 0 4.15000210 4.60418  
4.62050813 4.3601500 4.210006 4.4185758 1.31 1.27 0 0 0 0 4.8131819 4.111160 4.17041620 4.8170418  
2.6242 1.27 0 0 0 0 0 0 0 4.50010400 4.20190805 4.20111814 4.20159450 1.31 1.27 0 0 0 0 0 0 0  
4.50100424 4.1140017 3.35031 1.27 0 0 0 0 0 0 0 4.50120019 4.15111419 4.11080150 1.31 1.27 0 0 0 0 0  
0 0 0 4.50132012 4.15245031 1.27 0 0 0 0 0 0 0 0 4.50152406 4.120450 1.31 1.27 0 0 0 0 0 0 0  
4.50790811 4.11142250 1.31 1.27 0 0 0 0 0 0 0 4.50151114 4.19112450 1.31 1.27 0 0 0 0 0 0 0  
4.50151820 4.19081150 1.31 1.27 0 0 0 0 0 0 0 4.50152400 4.20191406 4.20085031 1.27 0 0 0 0 0 0 0  
4.50792480 4.19955031 1.27 0 0 0 0 0 0 0 4.50160818 4.10081950 1.31 1.27 0 0 0 0 0 0 0 4.50170416  
4.20041819 3.185031 1.27 0 0 0 0 0 0 0 4.50180207 4.4120317 4.225031 1.27 0 0 0 0 0 0 0 4.50180207  
4.4032011 3.45031 1.27 0 0 0 0 2.4131 1.27 0 0 0 0 4.4131917 4.24601514 4.8131918 2.6243 1.27 0 0 0 0 0  
0 0 0 4.45021413 4.18141104 4.60180217 4.8151918 2.4537 0 1.42 1.27 0 0 0 0 0 0 0 0 0 4.45162000  
4.13192012 4.60001708 4.19071204 4.19080262 4.66371200 4.8134531 1.27 0 0 0 0 0 0 0 2.4131 1.27 0  
0 0 0 2.4431 1.27 0 0 0 0 4.201907 4.14176250 3.761763 0 4.67141208 3.130802 0 4.64110423 4.130304  
1.17 0 4.66141415 4.4175031 1.27 0 0 0 0 4.201907 4.14176004 4.12000811 4.62500300 4.2060304  
4.63021414 4.15041738 4.6120008 4.11630214 3.125031 1.27 0 0 0 0 4.3041802 4.17081519 4.8141362  
2.5064 0 4.79241907 2.1413 0 4.151511 4.8020019 3.81413 0 3.51417 0 4.6041304 4.17001908 2.1306 0  
4.16200013 3.192012 0 4.2081702 4.20081918 0 3.1303 0 4.15041705 4.14171208 2.1306 0 4.170819  
4.7120419 2.802 0 4.6041412 4.4191724 0 4.20180813 1.6 0 4.18190019 1.4 0 4.18070015 3.41831 0  
3.1303 0 4.12200207 0 4.12141704 3.635031 1.27 0 0 0 0 4.11080204 4.13180462 4.50767283 2.5031 1.27  
0 0 0 0 4.10042422 4.14170318 4.62501620 4.131920 1.12 0 4.2081702 4.20081918 0 4.170819 4.7120419  
2.802 0 4.6041412 4.4191724 0 4.8120006 1.4 0 4.6041304 4.17001908 2.1413 0 4.2141812 4.14111406  
1.24 0 4.11001306 4.20000604 0 4.6041304 4.17001908 4.14135031 1.27 0 0 0 0 4.20171162 4.50071919  
4.15183732 4.32060819 4.7200163 4.2141232 4.3141208 4.230406 4.3043202 4.15245031 1.27 1.58 1.27  
1.27 4.17041620 4.8170412 4.4131918 4.63192319 0 1.47 1.27 1.27 4.1040020 4.19080520 4.11181420  
2.1594 1.27 4.10042401 4.14001703 1.27 4.12001915 4.11141911 2.801 1.27 4.13201215 1.24 1.27  
4.15240600 2.1204 1.27 4.79081111 2.1422 1.27 4.15111419 2.1124 1.27 4.15182019 2.811 1.27

4.15240020 4.19140620 1.8 1.27 4.79248019 1.95 1.27 4.16081810 2.819 1.27 4.17041620 4.4181918 1.27  
4.18020704 4.12031700 1.22 1.27 4.18020704 4.3201104 1.27 1.27 4.66141417 4.3081300 4.19041860  
4.79241907 4.14136319 2.23190 1.47 1.27 4.91319031 1.90 1.27 4.92319031 1.91 1.27 4.93319031  
1.92 1.27 4.94319031 1.93 1.27 4.95319031 1.94 1.27 4.96319031 1.95 1.27 4.97319031 1.96 1.27  
4.98319031 1.97 1.27 4.99319031 1.98 1.27 4.91903190 2.3199 1.27 4.91913190 3.319190 1.27  
4.91923190 3.319191 1.27 4.91933190 3.319192 1.27 4.91943190 3.319193 1.27 4.91953190 3.319194  
1.27 4.91963190 3.319195 1.27 4.91973191 2.3190 1.27 4.91983191 2.3191 1.27 4.91993191 2.3192 1.27  
4.92903191 2.3193 1.27 4.92913191 2.3194 1.27 4.92923191 2.3195 1.27 4.92933191 2.3196 1.27  
4.92943191 2.3197 1.27 4.92953191 2.3198 1.27 4.92963191 2.3199 1.27 4.92973191 3.319190 1.27  
4.92983191 3.319191 1.27 4.92993191 3.319192 1.27 4.93903191 3.319193 1.27 4.93913191 3.319194  
1.27 4.93923191 3.319195 1.27 4.93933192 2.3190 1.27 4.93943192 2.3191 1.27 4.93953192 2.3192 1.27  
4.93963192 2.3193 1.27 4.93973192 2.3194 1.27 4.93983192 2.3195 1.27 4.93993192 2.3196 1.27  
4.94903192 2.3197 1.27 4.94913192 2.3198 1.27 4.94923192 2.3199 1.27 4.94933192 3.319190 1.27  
4.94943192 3.319191 1.27 4.94953192 3.319192 1.27 4.94963192 3.319193 1.27 4.94973192 3.319194  
1.27 4.94983192 3.319195 1.27 4.94993193 2.3190 1.27 4.95903193 2.3191 1.27 4.95913193 2.3192 1.27  
4.95923193 2.3193 1.27 4.95933193 2.3194 1.27 4.95943193 2.3195 1.27 4.95953193 2.3196 1.27  
4.95963193 2.3197 1.27 4.95973193 2.3198 1.27 4.95983193 2.3199 1.27 4.95993193 3.319190 1.27  
4.96903193 3.319191 1.27 4.96913193 3.319192 1.27 4.96923193 3.319193 1.27 4.96933193 3.319194  
1.27 4.96943193 3.319195 1.27 4.96953194 2.3190 1.27 4.96963194 2.3191 1.27 4.96973194 2.3192 1.27  
4.96983194 2.3193 1.27 4.96993194 2.3194 1.27 4.97903194 2.3195 1.27 4.97913194 2.3196 1.27  
4.97923194 2.3197 1.27 4.97933194 2.3198 1.27 4.97943194 2.3199 1.27 4.97953194 3.319190 1.27  
4.97963194 3.319191 1.27 4.97973194 3.319192 1.27 4.97983194 3.319193 1.27 4.97993194 3.319194  
1.27 4.98903194 3.319195 1.27 4.98913195 2.3190 1.27 4.98923195 2.3191 1.27 4.98933195 2.3192 1.27  
4.98943195 2.3193 1.27 4.98953195 2.3194 1.27 4.98963195 2.3195 1.27 4.98973195 2.3196 1.27  
4.98983195 2.3197 1.27 4.98993195 2.3198 1.27 4.99903195 2.3199 1.27 4.99913195 3.319190 1.27  
4.99923195 3.319191 1.27 4.99933195 3.319192 1.27 4.99943195 3.319193 1.27 4.99953195 3.319194  
1.27 4.99963195 3.319195 1.27 4.99973196 2.3190 1.27 4.99983196 2.3191 1.27 4.99993196 2.3192 1.27  
4.91909031 3.963193 1.27 4.91909131 3.963194 1.27 4.91909231 3.963195 1.27 4.91909331 3.963196  
1.27 4.91909431 3.963197 1.27 4.91909531 3.963198 1.27 4.91909631 3.963199 1.27 4.91909731  
4.96319190 1.27 4.91909831 4.96319191 1.27 4.91909931 4.96319192 1.27 4.91919031 4.96319193 1.27  
4.91919131 4.96319194 1.27 4.91919231 4.96319195 1.27 4.91919331 3.973190 1.27 4.91919431  
3.973191 1.27 4.91919531 3.973192 1.27 4.91919631 3.973193 1.27 4.91919731 3.973194 1.27  
4.91919831 3.973195 1.27 4.91919931 3.973196 1.27 4.91929031 3.973197 1.27 4.91929131 3.973198  
1.27 4.91929231 3.973199 1.27 4.91929331 4.97319190 1.27 4.91929431 4.97319191 1.27 4.91929531  
4.97319192 1.27 4.91929631 4.97319193 1.27 4.91929731 4.97319194 1.27 4.91929831 4.97319195 1.27  
4.91929931 3.983190 1.27 4.91930931 3.983191 1.27 4.91939131 3.983192 1.27 4.91939231 3.983193  
1.27 4.91939331 3.983194 1.27 4.91939431 3.983195 1.27 4.91939531 3.983196 1.27 4.91939631  
3.983197 1.27 4.91939731 3.983198 1.27 4.91939831 3.983199 1.27 4.91939931 4.98319190 1.27  
4.91949031 4.98319191 1.27 4.91949131 4.98319192 1.27 4.91949231 4.98319193 1.27 4.91949331  
4.98319194 1.27 4.91949431 4.98319195 1.27 4.91949531 3.993190 1.27 4.91949631 3.993191 1.27  
4.91949731 3.993192 1.27 4.91949831 3.993193 1.27 4.91949931 3.993194 1.27 4.91959031 3.993195  
1.27 4.91959131 3.993196 1.27 4.91959231 3.993197 1.27 4.91959331 3.993198 1.27 4.91959431  
3.993199 1.27 4.91959531 4.99319190 1.27 4.91959631 4.99319191 1.27 4.91959731 4.99319192 1.27  
4.91959831 4.99319193 1.27 4.91959931 4.99319194 1.27 4.91960931 4.99319195 1.27 4.91969131  
4.91903190 1.27 4.91969231 4.91903191 1.27 4.91969331 4.91903192 1.27 4.91969431 4.91903193 1.27  
4.91969531 4.91903194 1.27 4.91969631 4.91903195 1.27 4.91969731 4.91903196 1.27 4.91969831  
4.91903197 1.27 4.91969931 4.91903198 1.27 4.91979031 4.91903199 1.27 4.91979131 4.91903191 1.90  
1.27 4.91979231 4.91903191 1.91 1.27 4.91979331 4.91903191 1.92 1.27 4.91979431 4.91903191 1.93  
1.27 4.91979531 4.91903191 1.94 1.27 4.91979631 4.91903191 1.95 1.27 4.91979731 4.91913190 1.27  
4.91979831 4.91913191 1.27 4.91979931 4.91913192 1.27 4.91989031 4.91913193 1.27 4.91989131

4.91913194 1.27 4.91989231 4.91913195 1.27 4.91989331 4.91913196 1.27 4.91989431 4.91913197 1.27  
4.91989531 4.91913198 1.27 4.91989631 4.91913199 1.27 4.91989731 4.91913191 1.90 1.27 4.91989831  
4.91913191 1.91 1.27 4.91989931 4.91913191 1.92 1.27 4.91999031 4.91913191 1.93 1.27 4.91999131  
4.91913191 1.94 1.27 4.91999231 4.91913191 1.95 1.27 4.91999331 4.91923190 1.27 4.91999431  
4.91923191 1.27 4.91999531 4.91923192 1.27 4.91999631 4.91923193 1.27 4.91999731 4.91923194 1.27  
4.91999831 4.91923195 1.27 4.91999931 4.91923196 1.27 4.92909031 4.91923197 1.27 4.92909131  
4.91923198 1.27 4.92909231 4.91923199 1.27 4.92909331 4.91923191 1.90 1.27 4.92909431 4.91923191  
1.91 1.27 4.92909531 4.91923191 1.92 1.27 4.92909631 4.91923191 1.93 1.27 4.92909731 4.91923191  
1.94 1.27 4.92909831 4.91923191 1.95 1.27 4.92909931 4.91933190 1.27 4.92919031 4.91933191 1.27  
4.92919131 4.91933192 1.27 4.92919231 4.91933193 1.27 4.92919331 4.91933194 1.27 4.92919431  
4.91933195 1.27 4.92919531 4.91933196 1.27 4.92919631 4.91933197 1.27 4.92919731 4.91933198 1.27  
4.92919831 4.91933199 1.27 4.92919931 4.91933191 1.90 1.27 4.92929031 4.91933191 1.91 1.27  
4.92929131 4.91933191 1.92 1.27 4.92929231 4.91933191 1.93 1.27 4.92929331 4.91933191 1.94 1.27  
4.92929431 4.91933191 1.95 1.27 4.92929531 4.91943190 1.27 4.92929631 4.91943191 1.27 4.92929731  
4.91943192 1.27 4.92929831 4.91943193 1.27 4.92929931 4.91943194 1.27 4.92939031 4.91943195 1.27  
4.92939131 4.91943196 1.27 4.92939231 4.91943197 1.27 4.92939331 4.91943198 1.27 4.92939431  
4.91943199 1.27 4.92939531 4.91943191 1.90 1.27 4.92939631 4.91943191 1.91 1.27 4.92939731  
4.91943191 1.92 1.27 4.92939831 4.91943191 1.93 1.27 4.92939931 4.91943191 1.94 1.27 4.92949031  
4.91943191 1.95 1.27 4.92949131 4.91953190 1.27 4.92949231 4.91953191 1.27 4.92949331 4.91953192  
1.27 4.92949431 4.91953193 1.27 4.92949531 4.91953194 1.27 4.92949631 4.91953195 1.27 4.92949731  
4.91953196 1.27 4.92949831 4.91953197 1.27 4.92949931 4.91953198 1.27 4.92959031 4.91953199 1.27  
4.92959131 4.91953191 1.90 1.27 4.92959231 4.91953191 1.91 1.27 4.92959331 4.91953191 1.92 1.27  
4.92959431 4.91953191 1.93 1.27 4.92959531 4.91953191 1.94 1.27 4.92959631 4.91953191 1.95 1.27  
1.27 4.12000813 4.2151559 4.14201915 4.20196315 1.24 0 1.47 1.27 1.27 4.8121514 2.1719 0 4.9181413  
1.27 4.5171412 0 4.17041514 4.17191100 4.1631108 4.1631500 4.6041808 3.250418 0 4.8121514 2.1719 0  
4.11041919 2.417 0 1.27 4.5171412 0 4.17041514 4.17191100 4.1631108 4.1631819 4.24110418 0  
4.8121514 2.1719 0 4.6041982 4.121511 4.4821924 4.11048207 4.4041931 0 4.79001700 4.6170015  
4.7821924 2.1104 1.27 4.5171412 0 4.17041514 4.17191100 4.1631511 4.192415 2.2018 0 4.8121514  
2.1719 0 4.82081215 4.11046714 4.2830412 4.15110019 2.431 0 4.79001700 4.6170015 2.731 0  
4.82150002 3.41731 0 4.79000604 4.65170400 1.10 1.27 1.27 3.30405 0 4.17040003 4.60091814  
4.13600508 4.11045705 4.8110413 4.120458 1.37 1.27 0 0 0 0 4.22081907 0 4.14150413 4.57050811  
4.4130012 2.431 0 4.45174558 0 2.18 0 4.5081104 1.37 1.27 0 0 0 0 0 0 0 4.3001900 0 1.62 0 4.9181413  
4.63111400 4.3570508 3.110458 1.27 0 0 0 0 4.17041920 2.1713 0 4.3001900 1.27 1.27 3.30405 0  
4.2170400 4.19046015 4.3055703 4.190031 0 4.14201915 4.20196005 4.8110413 4.120458 1.37 1.27 0 0 0  
0 3.31402 0 1.62 0 4.82081215 4.11046714 4.2830412 4.15110019 4.4571420 4.19152019 4.60050811  
4.4130012 2.431 0 4.15000604 4.18082504 4.62110419 4.19041758 1.27 0 0 0 0 4.18192411 2.418 0 1.62 0  
4.6041982 4.121511 4.4821924 4.11048207 4.4041957 1.58 1.27 0 0 0 0 4.2201819 4.14126018  
4.19241104 1.18 0 1.62 0 1.43 1.27 0 0 0 0 0 0 4.50190819 4.11045037 0 4.79001700 4.6170015  
4.7821924 3.110457 1.27 0 0 0 0 0 0 0 0 0 4.45830819 4.11044531 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
4.15001704 4.13196218 4.19241104 4.18424583 4.8191104 3.454131 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
4.5141319 4.82082504 4.62929431 1.27 0 0 0 0 0 0 0 0 0 0 0 4.18150002 4.4640519 4.4176291 1.92 1.27  
0 0 0 0 0 0 0 0 2.5831 1.27 0 0 0 0 0 0 0 0 0 4.50011403 3.245037 0 4.79001700 4.6170015 4.7821924  
3.110457 1.27 0 0 0 0 0 0 0 0 0 0 4.45651403 4.24830423 3.194531 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
4.15001704 4.13196218 4.19241104 4.18424565 4.14032483 4.4231945 2.4131 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
4.5141319 4.82082504 4.62919231 1.27 0 0 0 0 0 0 0 0 0 0 0 4.18150002 4.4640519 4.4176291 2.9231  
1.27 0 0 0 0 0 0 0 0 0 0 0 4.11040003 4.8130662 2.9195 1.27 0 0 0 0 0 0 0 2.5831 1.27 0 0 0 0 0 0 0  
4.50182001 4.19081911 3.45037 0 4.79001700 4.6170015 4.7821924 3.110457 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.45822001 4.19081911 3.44531 1.27 0 0 0 0 0 0 0 0 0 0 4.15001704 4.13196218 4.19241104  
4.18424583 4.8191104 3.454131 1.27 0 0 0 0 0 0 0 0 0 0 4.5141319 4.82082504 4.62919831 1.27 0 0 0  
0 0 0 0 0 0 0 0 4.18150002 4.4640519 4.4176296 1.27 0 0 0 0 0 0 0 1.58 1.27 0 0 0 0 1.44 1.27 1.27 0 0 0

0 4.18191417 1.24 0 1.62 0 2.4241 1.27 1.27 0 0 0 0 3.51417 0 4.18191720 4.2192017 4.4600803 1.31 0  
4.3041900 3.81118 0 2.813 0 4.3001900 4.63081904 4.12185758 1.37 1.27 0 0 0 0 0 0 0 0 4.18191417  
4.24630015 4.15041303 4.57790017 4.61700 4.15075718 4.19172002 4.19201704 4.60080331 0  
4.2201819 4.14126018 4.19241104 4.18424519 4.8191104 4.45415858 1.27 0 0 0 0 0 0 0 0 4.18191417  
4.24630015 4.15041303 4.57821500 4.2041757 2.9131 0 4.91925858 1.27 0 0 0 0 0 0 0 4.18191417  
4.24630015 4.15041303 4.57790017 4.61700 4.15075705 4.50830812 4.4181900 3.121537 0 4.43030419  
4.81118 4.42451908 4.12041819 4.121545 4.41445031 0 4.2201819 4.14126018 4.19241104 4.18424518  
4.20011908 4.19110445 3.415858 1.27 0 0 0 0 0 0 0 4.18191417 4.24630015 4.15041303 4.57821500  
4.2041757 2.9131 0 4.91925858 1.27 0 0 0 0 0 0 0 0 4.15171415 4.4171908 2.418 0 1.62  
0 4.3041900 4.8111842 4.50151714 4.15041719 4.8041850 1.41 1.27 0 0 0 0 0 0 0 3.51417 0 4.10042431  
0 4.21001120 1.4 0 2.813 0 4.15171415 4.4171908 4.4186308 4.19041218 3.575837 1.27 0 0 0 0 0 0 0 0 0  
0 0 1.39 0 4.71001303 2.1104 0 4.12201119 4.8591108 2.1304 0 4.15171415 4.4171924 0 4.21001120  
2.418 0 2.124 0 4.17041511 4.20813 1.6 0 4.13042211 4.8130418 0 4.22081907 0 4.33011732 1.34 0  
4.19000618 1.27 0 0 0 0 0 0 0 0 0 0 0 0 2.805 0 4.8180813 4.18190013 4.2045721 4.112004 1.31 0  
4.18191758 1.37 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0 4.21001120 1.4 0 1.62 0 4.21001120 4.4631704  
4.15110002 4.4574529 3.134531 0 4.45330117 4.32344558 1.27 0 0 0 0 0 0 0 0 0 0 0 4.18191417  
4.24630015 4.15041303 4.57790017 4.61700 4.15075705 4.50431004 3.244437 0 4.43210011 4.20044450  
1.31 0 4.2201819 4.14126018 4.19241104 4.18424501 4.14032445 3.415858 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.18191417 4.24630015 4.15041303 4.57821500 4.2041757 2.9131 0 4.91925858 1.27 1.27 0 0 0 0 0 0 0 0  
4.18191417 4.24630015 4.15041303 4.57790006 4.4651704 4.105758 1.58 1.27 1.27 0 0 0 0 4.3140263  
4.1200811 4.3571819 4.14172458 1.27 1.27 2.805 0 4.60601300 4.12046060 0 2.6262 0 4.50606012  
4.81360 3.605037 1.27 0 0 0 4.8131520 4.19600508 4.11041300 2.1204 0 1.62 0 4.8131520 4.19575068  
4.131904170 4.8131520 1.19 0 4.5081104 4.13001204 4.63091814 1.13 0 4.57860819 1.7 0 4.4231904  
4.13180814 3.135837 0 2.5058 1.27 0 0 0 0 4.9181413 4.60030019 1.0 0 1.62 0 4.17040003 4.60091814  
4.13600508 4.11045708 4.13152019 4.60050811 4.4130012 2.458 1.27 0 0 0 1.27 0 0 0 0 1.39 0  
4.79170813 1.19 0 4.73827877 0 4.3001900 0 2.1914 0 4.21041708 2.524 0 4.18191720 4.2192017 1.4 1.27  
0 0 0 0 4.15170813 4.19570918 4.14136303 4.20121518 4.57091814 4.13600300 3.190031 0 4.8130304  
4.13196294 2.5858 1.27 0 0 0 1.27 0 0 0 4.2170400 4.19046015 4.3055709 4.18141360 4.3001900 1.31  
0 4.45142019 4.15201963 4.15030545 1.58 1.27 1.27 4.12000813 4.63021515 0 1.47 1.27 1.27 2.3256 1.27  
0 1.56 0 4.72131819 4.111100 4.19081413 0 4.72131819 4.17200219 4.8141318 0 3.51417 0 4.76001907  
4.4120019 4.8020814 1.13 1.27 0 1.56 1.27 0 1.56 0 4.79170417 4.4162008 4.18081904 2.1837 1.27 0 1.56  
0 2.9163 0 3.666161 0 4.66141215 4.8110417 0 4.57066161 0 2.1417 0 4.85081820 2.11 0 4.82192003  
3.81458 1.27 0 1.56 0 2.9263 0 4.66760010 1.4 0 3.570805 0 4.20180813 1.6 0 1.0 0 4.66760010 1.4 0  
4.15171409 4.4021958 1.27 0 1.56 0 2.9363 0 4.21021510 1.6 0 4.57051417 0 4.12001300 4.6081306 0  
3.666161 0 4.11080117 4.170804 2.1858 1.27 0 1.56 0 2.9463 0 4.73827877 0 3.51417 0 4.76140304  
2.1713 0 3.666161 0 4.11080117 3.1724 0 4.57131114 4.7120013 4.13320918 3.141358 1.27 0 1.56 1.27 0  
1.56 0 4.82190415 0 2.9137 0 4.72131819 3.1111 0 4.21021510 1.6 1.27 0 1.56 0 1.59 0 4.66111413 1.4 0  
3.190704 0 4.21021510 1.6 0 4.17041514 4.18081914 3.172437 1.27 0 1.56 0 0 0 3.60819 0 4.2111413 1.4 0  
0 4.7191915 4.18373232 4.6081907 4.20016302 4.14123212 4.8021714 4.18140519 4.32210215  
4.10066306 2.819 1.27 0 1.56 0 1.59 0 4.77002108 4.6001904 0 2.1914 0 3.190704 0 4.21021510 1.6 0  
4.3081704 4.2191417 2.2437 1.27 0 1.56 0 0 0 2.203 0 4.21021510 1.6 1.27 0 1.56 0 1.59 0 4.65141419  
4.18191700 1.15 0 4.21021510 2.637 1.27 0 1.56 0 0 0 4.63290114 4.14191819 4.17001559 4.21021510  
4.6630100 1.19 0 3.571413 0 4.86081303 4.14221858 1.27 0 1.56 0 0 0 4.63320114 4.14191819  
4.17001559 4.21021510 4.6631807 0 3.571413 0 4.84130823 1.58 1.27 0 1.56 0 1.59 0 4.72131904  
4.6170019 1.4 0 4.21021510 1.6 0 4.22081907 0 4.24142017 0 4.18070411 2.1137 1.27 0 1.56 0 0 0  
4.63292102 3.151006 0 4.8131904 4.6170019 1.4 0 4.8131819 3.1111 1.27 0 1.56 1.27 0 1.56 0 4.82190415  
0 2.9237 0 4.72131819 3.1111 0 4.73827877 0 3.51417 0 4.76140304 2.1713 0 3.666161 0 4.11080117  
3.1724 1.27 0 1.56 0 1.59 0 3.841804 0 4.21021510 1.6 0 2.1914 0 4.8131819 3.1111 0 3.190704 0  
4.13111407 4.12001313 4.59091814 1.13 0 4.15000210 4.60437 1.27 0 1.56 0 0 0 4.63292102 3.151006 0  
4.8131819 3.1111 0 4.13111407 4.12001313 4.59091814 1.13 1.27 0 1.56 1.27 0 1.56 0 4.82190415 0

2.9337 0 4.66141215 3.81104 0 3.190704 0 4.79171406 3.170012 1.27 0 1.56 0 1.59 0 4.78150413 0 1.0 0  
4.2141212 3.1303 0 4.15171412 2.1519 0 4.57021203 1.58 0 2.1417 0 4.79142204 4.17820704 3.111163  
1.27 0 1.56 0 1.59 0 4.77002108 4.6001904 0 2.1914 0 3.190704 0 4.3081704 4.2191417 1.24 0 4.2141319  
4.81308 2.1306 0 3.190704 0 4.18142017 2.204 0 4.2140304 0 4.57046306 2.6331 0 4.12000813  
4.63021515 2.5863 1.27 0 1.56 0 1.59 0 4.66141215 3.81104 0 3.190704 0 4.15171406 3.170012 0  
4.20180813 1.6 0 4.6616137 1.27 0 1.56 0 0 0 3.66161 0 4.59726737 4.32210215 4.10061232 4.21021510  
4.6320813 4.18190011 4.11040332 4.23969459 4.22081303 4.14221832 4.81302111 3.200304 0  
4.12000813 4.63021515 0 2.5914 0 4.76001907 4.4120019 4.8020814 4.13630423 1.4 1.27 0 1.56 0 0 0  
4.57810415 4.11000204 0 4.67373221 4.2151006 4.12322102 4.15100632 4.8131819 4.111104 4.3322396  
4.94592208 4.13031422 4.18320813 4.2112003 1.4 0 4.22081907 0 3.190704 0 4.2141717 3.40219 0  
4.15001907 0 2.1914 0 4.24142017 0 4.21021510 1.6 0 4.8131819 4.111100 4.19081413 1.58 1.27 0 1.56 0  
1.59 0 2.7205 0 4.20180813 1.6 0 4.76828566 0 4.57850818 3.200011 0 4.82192003 4.8145837 1.27 0 1.56  
0 0 0 1.59 0 4.78150413 0 3.190704 0 4.67042104 4.111141504 1.17 0 4.66141212 3.1303 0 4.79171412  
2.1519 0 3.51417 0 4.85081820 2.110 0 4.82192003 3.81463 1.27 0 1.56 0 0 0 1.59 0 4.77002108 4.6001904  
0 2.1914 0 3.190704 0 4.3081704 4.2191417 1.24 0 4.2141319 4.81308 2.1306 0 3.190704 0 4.18142017  
2.204 0 4.2140304 1.63 1.27 0 1.56 0 0 0 1.59 0 4.66141215 3.81104 0 3.190704 0 4.15171406 3.170012 0  
4.20180813 1.6 0 3.21137 1.27 0 1.56 0 0 0 0 2.211 0 4.32687118 1.2 0 4.12000813 4.63021515 0 2.3272  
0 4.67373221 4.2151006 4.12322102 4.15100632 4.8131819 4.111104 4.3322396 4.94592208 4.13031422  
4.18320813 4.2112003 1.4 0 4.32110813 1.10 0 4.32788483 4.37760019 4.7041200 4.19080208  
4.14136304 2.2304 1.27 0 1.56 0 0 0 4.57810415 4.11000204 0 4.67373221 4.2151006 4.12322102  
4.15100632 4.8131819 4.111104 4.3322396 4.94592208 4.13031422 4.18320813 4.2112003 1.4 0  
4.22081907 0 3.190704 0 4.2141717 3.40219 0 4.15001907 0 2.1914 0 4.24142017 0 4.21021510 1.6 0  
4.8131819 4.111100 4.19081413 1.58 1.27 0 1.56 1.27 0 1.56 0 4.82190415 0 2.94370 3.812013 0  
3.190704 0 4.79171406 3.170012 1.27 0 1.56 0 1.59 0 4.64051904 1.17 0 4.18200202 4.4181805 2.2011 0  
4.2141215 4.8110019 4.8141331 0 3.172013 0 3.190704 0 4.4230402 4.20190001 3.110437 1.27 0 1.56 0 0  
0 4.63297600 4.19070412 4.190802 4.8141363 3.42304 1.27 0 1.56 1.27 0 1.56 0 4.83171420 4.1110418  
4.7141419 4.8130637 1.27 0 1.56 0 1.59 0 4.68131820 2.1704 0 3.1111 0 4.15001907 1.18 0 3.1704 0  
4.2141717 3.40219 0 3.1303 0 4.15140813 1.19 0 2.1914 0 3.190704 0 4.151517 4.14151708 3.1904 0  
4.3081704 4.2191417 4.8041863 1.27 0 1.56 0 1.59 0 4.68131820 2.1704 0 3.1111 0 4.3041504 4.13030413  
4.2080418 0 3.1704 0 4.8131819 4.111104 1.3 0 3.1303 0 4.20204 4.18180801 3.110463 1.27 0 1.56 0 1.59  
0 4.66070402 1.10 0 3.51417 0 3.1324 0 4.2141215 4.8110019 3.81413 0 4.4171714 2.1718 0 3.1303 0  
4.17041814 3.112104 0 4.12081818 3.81306 0 4.8130211 3.200304 0 4.15001907 1.18 0 2.1417 0  
4.11080117 4.170804 2.1863 1.27 0 1.56 1.27 0 1.56 0 4.68230012 4.15110437 1.27 0 1.56 0 1.34 0  
4.12100308 1.17 0 4.2141305 2.806 1.27 0 1.56 0 1.34 0 4.63297600 4.19070412 4.190802 4.8141363  
3.42304 1.27 0 1.56 0 1.27 0 1.56 0 3.881420 0 4.18071420 2.1103 0 3.131422 0 2.104 0 4.11104 0 2.1914  
0 4.8131904 4.17000219 0 4.22081907 0 3.190704 0 4.12041320 0 3.1303 0 4.15041705 3.141712 0  
4.21001708 3.142018 0 4.14150417 4.190814 2.1318 0 2.18 0 4.3041802 4.17080104 1.3 0 2.813 0  
3.190704 0 4.15171406 4.17001263 1.27 0 1.56 1.27 0 1.56 0 3.691417 0 4.5201719 3.70417 0 4.7041115 0  
3.1303 0 4.3140220 4.12041319 4.190814 2.1331 0 4.17040504 1.17 0 2.1914 0 3.190704 0 4.14050508  
4.2080011 0 4.3140220 4.12041319 4.190814 1.13 0 2.1405 0 3.190704 0 4.73827877 0 3.514170  
4.76140304 2.1713 0 3.666161 0 4.11080117 3.1724 0 3.1303 0 4.21021510 2.663 1.27 0 2.5632 1.27  
4.39081302 4.11200304 0 4.33081418 4.19170400 2.1234 1.27 4.39081302 4.11200304 0 4.33181917  
4.8130634 1.27 4.39081302 4.11200304 0 4.33180419 1.34 1.27 4.39081302 4.11200304 0 4.33201314  
4.17030417 4.4036012 3.1534 1.27 4.39081302 4.11200304 0 4.33120412 4.14172434 1.27 4.39081302  
4.11200304 0 4.33020717 4.14131434 1.27 4.39081302 4.11200304 0 4.33181819 4.17040012 1.34 1.27  
4.39081302 4.11200304 0 4.33081412 4.130815 1.34 1.27 4.39081302 4.11200304 0 4.33051819  
4.17040012 1.34 1.27 4.39081302 4.11200304 0 4.50673732 4.21021510 4.6123221 4.2151006  
4.32081318 4.19001111 4.4033223 4.96945922 4.8130314 4.22183208 4.13021120 4.3043213 4.11140712  
4.131332 4.9181413 4.63071515 1.50 1.27 4.39081302 4.11200304 0 4.33050811 4.4182418 4.19041234  
1.27 1.27 4.20180813 1.6 0 4.9181413 0 1.62 0 4.13111407 4.12001313 4.37370918 3.141336 1.27 1.27

2.3232 0 1.64 0 4.5201302 4.19081413 0 2.1914 0 3.60419 0 3.190704 0 4.2201717 3.41319 0 4.19081204  
4.18190012 1.15 0 2.813 0 3.190704 0 4.3041808 3.170403 0 4.5141712 2.19 1.27 4.18190337 4.37181917  
3.81306 0 4.6041966 4.20171704 4.13198308 4.12041819 4.121557 1.58 0 1.43 1.27 0 0 0 0 4.201914 0  
3.131422 0 1.62 0 4.18190337 4.37020717 4.14131437 4.37182418 4.19041260 4.2111402 4.10373713  
4.14225758 1.36 1.27 0 0 0 0 4.201914 0 4.8136019 4.8120460 1.19 0 1.62 0 4.18190337 4.37020717  
4.14131437 4.37182418 4.19041260 4.2111402 4.10373719 4.14601908 4.12046019 4.57131422 2.5836  
1.27 1.27 0 0 0 0 4.18190337 4.37181917 4.8130618 4.19170400 1.12 0 3.181836 1.27 0 0 0 0 2.1818 0  
2.3333 0 4.18190337 4.37152019 4.60190812 4.4571819 4.3373706 4.12190812 4.4575508 4.13601908  
4.12046019 2.5831 0 4.50538859 4.53125953 4.3835371 4.37537637 4.53828950 2.5836 1.27 0 0 0 0  
4.17041920 2.1713 0 4.18186318 4.19175758 1.36 1.27 1.44 1.27 1.27 2.3232 0 4.65001804 0 4.2110018  
1.18 0 3.51417 0 2.7982 0 4.57791714 4.15041719 1.24 0 4.82041958 1.27 4.2110018 1.18 0 4.79171415  
4.4171924 3.820419 0 1.43 1.27 4.15200111 3.80237 1.27 0 0 0 0 4.21081719 3.200011 0 4.21140803 0  
4.3081815 4.11002457 1.58 0 4.2141318 1.19 0 1.62 0 2.9036 1.27 0 0 0 0 4.21081719 3.200011 0  
4.9181413 0 4.19146009 4.18141357 1.58 0 4.2141318 1.19 0 1.62 0 2.9036 1.27 0 0 0 0 4.21081719  
3.200011 0 4.21140803 0 4.3041104 4.19046015 4.17141504 4.17192457 4.2141318 1.19 0 4.18190337  
4.37181917 4.8130655 0 4.10042458 0 1.62 0 2.9036 1.27 0 0 0 0 4.21081719 3.200011 0 4.40791714  
4.15041719 4.24820419 2.5758 0 1.62 0 4.3040500 4.20111936 1.27 2.4436 1.27 1.27 2.3232 0 2.6413 0  
4.4230012 3.151104 0 2.1405 0 1.0 0 4.20180417 4.59030405 4.8130403 0 4.15171415 4.4171924 0  
3.180419 0 4.20180813 1.6 0 4.73827877 1.27 4.2110018 1.18 0 4.67241300 4.12080279 4.17141504  
4.17192482 2.419 0 1.37 0 4.15200111 2.802 0 4.79171415 4.4171924 3.820419 0 1.43 1.27 4.15170821  
4.190437 1.27 0 0 0 0 4.9181413 0 4.15171415 4.4171908 3.41836 1.27 4.15200111 3.80237 1.27 0 0 0 0  
4.67241300 4.12080279 4.17141504 4.17192482 4.4195702 4.14131819 0 4.9181413 1.55 0 4.15171415  
2.1858 0 1.37 0 4.15171415 4.4171908 4.4185715 4.17141518 1.58 0 2.4344 1.27 1.27 0 0 0 0 4.21140803  
0 4.3081815 4.11002457 1.58 0 4.2141318 1.19 0 4.14201417 4.17080304 0 1.43 1.27 0 0 0 0 0 0 0 0  
4.18190337 4.37021420 1.19 0 2.3333 0 4.15171415 4.4171908 4.4186303 4.20121557 2.9458 0 2.3333 0  
4.18190337 4.37041303 2.1136 1.27 0 0 0 0 1.44 1.27 1.27 0 0 0 0 4.9181413 0 4.19146009 4.18141357  
1.58 0 4.2141318 1.19 0 4.14201417 4.17080304 0 1.43 1.27 0 0 0 0 0 0 0 4.17041920 2.1713 0  
4.15171415 4.4171908 3.41836 1.27 0 0 0 0 1.44 1.27 1.27 0 0 0 0 4.21140803 0 4.3041104 4.19046015  
4.17141504 4.17192457 4.2141318 1.19 0 4.18190337 4.37181917 4.8130655 0 4.10042458 0 4.14210417  
4.17080304 0 1.43 1.27 0 0 0 0 0 0 0 4.15171415 4.4171908 4.4186304 4.17001804 4.57100424 2.5836  
1.27 0 0 0 0 1.44 1.27 2.4436 1.27 1.27 2.3232 0 4.66141415 2.417 0 4.2110018 1.18 0 2.1914 0 4.4130200  
4.15182011 3.1904 0 3.190704 0 4.18191720 4.2192017 1.4 1.27 4.2110018 1.18 0 4.66141415 2.417 0  
1.43 1.27 4.15170821 4.190437 1.27 0 0 0 0 4.18190337 4.37181917 3.81306 0 3.80336 1.27 0 0 0 0  
4.18190337 4.37201308 4.16200460 4.15191733 4.79171415 4.4171924 4.82041934 0 4.15171415  
4.4171924 4.82041936 1.27 0 0 0 0 4.18190337 4.37181917 3.81306 0 4.19081204 4.18190012 2.1536 1.27  
1.27 0 0 0 0 4.18190019 2.802 0 4.18190337 4.37180419 4.33181903 4.37371819 4.17081306 1.34 0  
4.8038204 2.1936 1.27 1.27 4.15200111 3.80237 1.27 0 0 0 0 4.66141415 4.4175718 4.19033737  
4.18191708 2.1306 0 3.80331 0 4.18190337 4.37201308 4.16200460 4.15191733 4.79171415 4.4171924  
4.82041934 0 3.151858 1.27 0 0 0 0 0 0 0 1.37 0 4.8038204 4.19033737 4.12142104 4.57080358 2.5831 0  
4.15171415 4.4171924 4.82041957 4.18190337 4.37121421 4.4571518 3.585831 0 4.19081204  
4.18190012 4.15570604 4.19662017 4.17041319 4.83081204 4.18190012 4.15575858 0 1.43 1.27 0 0 0 0 0  
0 0 0 2.805 0 4.57080382 4.4196305 4.8130357 4.19070818 4.59340803 1.58 0 2.4962 0 4.8038204  
4.19630413 4.3575858 0 1.43 1.27 0 0 0 0 0 0 0 0 0 4.19071714 1.22 0 4.18190337 4.37081321  
4.110803 4.60001706 4.20120413 4.19575072 1.67 0 4.111704 3.324 0 4.4230818 4.19186350 2.5836 1.27  
0 0 0 0 0 0 1.44 1.27 0 0 0 0 0 0 4.8038204 4.19630813 4.18041719 4.57190708 4.18593408  
3.35836 1.27 0 0 0 1.44 1.27 1.27 0 0 0 0 4.2141318 1.19 0 4.18190337 4.37181917 4.8130655 0  
4.6041972 3.35758 0 4.2141318 1.19 0 1.43 0 4.17041920 2.1713 0 3.80336 0 1.44 1.27 0 0 0 0 4.2141318  
1.19 0 4.18190337 4.37181917 4.8130655 0 4.6041983 4.8120418 4.19001215 2.5758 0 4.2141318 1.19 0  
1.43 0 4.17041920 2.1713 0 4.19081204 4.18190012 2.1536 0 1.44 1.27 0 0 0 0 4.2141318 1.19 0  
4.79171415 4.4171924 4.82041956 0 4.6041979 4.17141504 4.17192482 4.4195758 0 4.2141318 1.19 0

1.43 0 4.17041920 2.1713 0 4.15171415 4.4171924 4.82041963 4.6041957 2.5836 0 1.44 1.27 1.27 0 0 0 0  
4.21140803 0 4.3081815 4.11002457 1.58 0 4.2141318 1.19 0 1.43 1.27 0 0 0 0 0 0 0 4.18190337  
4.37021420 1.19 0 2.3333 0 4.50726737 0 1.50 0 2.3333 0 2.803 0 2.3333 0 4.18190337 4.37041303 2.1136  
1.27 0 0 0 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 4.50830812 4.4181900 3.121537 0 1.50 0  
2.3333 0 4.19081204 4.18190012 1.15 0 2.3333 0 4.18190337 4.37041303 2.1136 1.27 0 0 0 0 0 0 0 0  
4.18190337 4.37021420 1.19 0 2.3333 0 4.50791714 4.15041719 4.8041837 0 2.5036 1.27 0 0 0 0 1.44 1.27 1.27 0 0 0 0  
4.15171415 4.4171924 4.82041959 4.34030818 4.15110024 3.575836 1.27 0 0 0 0 1.44 1.27 1.27 0 0 0 0  
4.18190019 2.802 0 4.21140803 0 4.17041804 4.19720382 4.4195758 0 1.43 1.27 0 0 0 0 0 0 0 0 4.8038204  
4.19630211 4.4001757 2.5836 1.27 0 0 0 0 1.44 1.27 1.27 0 0 0 0 4.18190019 2.802 0 4.21140803 0  
4.20150300 4.19047203 4.82041957 4.2141318 1.19 0 4.18190337 4.37180419 4.33181903 4.37371819  
4.17081306 2.3455 0 4.13042272 4.3820419 1.58 0 1.43 1.27 0 0 0 0 0 0 0 4.8038204 1.19 0 1.62 0  
4.13042272 4.3820419 1.36 1.27 0 0 0 0 1.44 1.27 1.27 0 0 0 0 0 4.21140803 0 4.18002104 4.57021413  
2.1819 0 4.18190337 4.37181917 4.8130655 0 4.5081104 4.13001204 1.58 0 4.2141318 1.19 0 1.43 1.27 0  
0 0 0 0 0 0 0 4.9181413 0 2.936 1.27 0 0 0 0 0 0 0 4.18190337 4.37080518 4.19170400 1.12 0 4.8136908  
4.11045705 4.8110413 4.120458 1.36 1.27 0 0 0 0 0 0 0 0 0 2.805 0 4.57081369 4.8110463 4.8186014  
4.15041357 2.5858 0 1.43 1.27 0 0 0 0 0 0 0 0 0 0 0 4.8136908 2.1104 0 2.3434 0 2.936 1.27 0 0 0 0 0 0 0  
0 0 0 0 0 4.8136908 4.11046302 4.11141804 3.575836 1.27 0 0 0 0 0 0 0 1.44 1.27 0 0 0 0 0 0 0 1.27 0 0  
0 0 0 0 0 4.9420803 4.41425019 4.8120418 4.19001215 2.5041 0 1.62 0 4.19081204 4.18190012 2.1536  
1.27 0 0 0 0 0 0 0 0 4.9420803 4.41425015 4.17141504 4.17190804 3.185041 0 1.62 0 4.15171415  
4.4171924 4.82041959 4.34191460 4.9181413 3.575836 1.27 1.27 0 0 0 0 0 0 0 4.18190337 4.37140518  
4.19170400 1.12 0 4.14201969 4.8110457 4.5081104 4.13001204 2.5836 1.27 0 0 0 0 0 0 0 2.805 0  
4.57142019 4.69081104 4.63081860 4.14150413 3.575836 0 1.43 1.27 0 0 0 0 0 0 0 0 0 0 4.14201969  
3.81104 0 2.3333 0 4.9630320 4.12155794 2.5836 1.27 0 0 0 0 0 0 0 1.44 1.27 0 0 0 0 1.44 1.27 1.27 0 0 0  
0 4.18190019 2.802 0 4.18190337 4.37201314 4.17030417 4.4036012 4.153318 4.19033737 4.18191708  
3.130631 0 4.18190337 4.37201308 4.16200460 4.15191733 4.66141415 4.4173434 0 4.11140003  
4.64111157 4.2141318 1.19 0 4.18190337 4.37181917 4.8130655 0 4.5081104 4.13001204 1.58 0 1.43 1.27  
0 0 0 0 0 0 0 4.18190337 4.37080518 4.19170400 1.12 0 4.5081104 4.57050811 4.4130012 3.45836 1.27  
0 0 0 0 0 0 0 2.805 0 4.57490508 4.11046308 4.18601415 4.4135758 1.58 0 1.43 1.27 0 0 0 0 0 0 0 0 0  
0 4.19071714 1.22 0 4.18190337 4.37172013 4.19081204 4.60041717 4.14175750 4.66142011 1.3 0  
3.131419 0 4.14150413 0 4.5081104 0 3.51417 0 4.17040003 4.8130650 2.5836 1.27 0 0 0 0 0 0 0 0 1.44  
1.27 0 0 0 0 0 0 0 4.9181413 0 2.936 1.27 0 0 0 0 0 0 0 4.5081104 0 2.3434 0 2.936 1.27 1.27 0 0 0 0 0  
0 0 4.18190337 4.37201314 4.17030417 4.4036012 4.153318 4.19033737 4.18191708 3.130631 0  
4.18190337 4.37201308 4.16200460 4.15191733 4.66141415 4.4173434 0 4.2141415 4.4177600 2.1536  
1.27 1.27 0 0 0 0 0 0 0 3.51417 0 4.57002019 2.1455 0 4.42080331 0 4.3001900 1.41 0 1.37 0 4.9630819  
4.4121857 2.5858 0 1.43 1.27 0 0 0 0 0 0 0 0 0 4.18190337 4.37181917 3.81306 0 4.19081204  
4.18190012 1.15 0 1.62 0 4.3001900 4.42501908 4.12041819 4.121550 2.4136 1.27 0 0 0 0 0 0 0 0 0 0 0  
4.9181413 0 4.15171415 4.4171908 2.418 0 1.62 0 4.3001900 4.42501517 4.14150417 4.19080418  
3.504136 1.27 0 0 0 0 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 0 0 0 4.201914 0 4.15171415 4.4171924  
3.820419 0 1.62 0 4.18190337 4.3720100 4.4602013 4.8162004 4.33672413 4.120802 4.79171415  
4.4171924 4.82041934 4.57151714 4.15041719 4.8041858 1.36 1.27 0 0 0 0 0 0 0 0 0 0 4.201914 0  
4.2141415 2.417 0 1.62 0 4.18190337 4.37120010 4.4602013 4.8162004 4.33661414 4.15041734  
4.57080331 0 4.18190337 4.37121421 4.4571517 4.14150417 4.19248204 4.19585836 1.27 0 0 0 0 0 0 0 0  
0 0 0 0 4.2141415 4.4175934 4.19081204 4.18190012 1.15 0 1.62 0 4.19081204 4.18190012 2.1536 0  
2.3232 0 4.68131820 2.1704 0 3.190704 0 4.19081204 4.18190012 1.15 0 2.818 0 4.15170418 4.4172104  
1.3 1.27 0 0 0 0 0 0 0 0 0 0 1.27 0 0 0 0 0 0 0 0 0 0 4.2141415 4.4177600 4.15420803 1.41 0 1.62 0  
4.18190337 4.37121421 4.4570214 4.14150417 2.5836 1.27 0 0 0 0 0 0 0 1.44 1.27 1.27 0 0 0 0 0 0  
4.17041920 2.1713 0 4.2141415 4.4177600 2.1536 1.27 0 0 0 0 1.44 1.27 1.27 0 0 0 0 4.21140803 0  
4.20150300 4.19047917 4.14150417 4.19248204 4.19570214 3.131819 0 4.9181413 1.55 0 4.13042279  
4.17141518 1.58 0 1.43 1.27 0 0 0 0 0 0 4.15171415 4.4171924 3.820419 0 1.62 0 4.18190337  
4.37120010 4.4602013 4.8162004 4.33672413 4.120802 4.79171415 4.4171924 4.82041934 4.57130422

4.79171415 3.185836 1.27 0 0 0 1.44 1.27 1.27 0 0 0 0 4.21140803 0 4.3041104 4.19047917 4.14150417  
4.19245702 4.14131819 0 4.18190337 4.37181917 4.8130655 0 4.10042458 0 1.43 1.27 0 0 0 0 0 0 0  
4.15171415 4.4171924 4.82041959 4.34030411 4.4190460 4.15171415 4.4171924 4.57100424 2.5836 1.27  
0 0 0 1.44 1.27 1.27 0 0 0 0 4.21140803 0 4.2070013 4.6047203 4.57021413 2.1819 0 4.18190337  
4.37181917 4.8130655 0 4.13042272 2.358 0 1.43 1.27 0 0 0 0 0 0 0 2.805 0 4.57080382 4.4196305  
4.8130357 4.13042272 2.358 0 2.4962 0 4.8038204 4.19630413 4.3575858 0 1.43 1.27 0 0 0 0 0 0 0 0 0  
0 4.19071714 1.22 0 4.18190337 4.37081321 4.110803 4.60001706 4.20120413 4.19575072 1.67 0  
4.111704 3.324 0 4.4230818 4.19186350 2.5836 1.27 0 0 0 0 0 0 0 0 1.44 1.27 0 0 0 0 0 0 0 0 0 4.8038204  
4.19630417 4.180457 4.8035836 1.27 0 0 0 0 0 0 0 2.803 0 1.62 0 4.13042272 2.336 1.27 0 0 0 0 0 0 0  
4.8038204 4.19630813 4.18041719 4.57080358 1.36 1.27 0 0 0 0 1.44 1.27 2.4436 1.27 1.27 4.18190337  
4.37180419 4.33181903 4.37371819 4.17081306 1.34 0 4.66141415 4.4173737 4.8038204 2.1936 1.27  
1.27 4.21140803 0 4.3081815 4.11002476 4.4132057 1.58 0 1.43 1.27 0 0 0 0 4.18190337 4.37021420 1.19  
0 2.3333 0 4.50760413 4.20372913 2.5036 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 3.509163 0  
4.66170400 2.1904 0 3.130422 0 4.18191720 4.2192017 4.4291350 1.36 1.27 0 0 0 0 4.18190337  
4.37021420 1.19 0 2.3333 0 3.509263 0 4.67081815 3.110024 0 4.18191720 4.2192017 4.4291350 1.36  
1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 3.509363 0 4.84150300 2.1904 0 4.18191720  
4.2192017 1.4 0 4.15171415 4.4171908 4.4182913 2.5036 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0  
2.3333 0 3.509463 0 4.67041104 2.1904 0 4.18191720 4.2192017 4.4291350 1.36 1.27 0 0 0 0 4.18190337  
4.37021420 1.19 0 2.3333 0 3.509563 0 4.66070013 2.604 0 4.18191720 4.2192017 1.4 0 4.72672913  
2.5036 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 3.509663 0 4.82002104 0 4.18191720  
4.2192017 1.4 0 2.1914 0 4.5081104 4.29135036 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0  
3.509763 0 4.75140003 0 4.18191720 4.2192017 1.4 0 4.5171412 0 4.5081104 4.29135036 1.27 0 0 0 0  
4.18190337 4.37021420 1.19 0 2.3333 0 3.509863 0 4.67041104 2.1904 0 1.0 0 4.15171415 4.4171924 0  
4.5171412 0 4.18191720 4.2192017 4.4291350 1.36 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0  
3.509963 0 4.71041115 0 4.3140220 4.12041319 4.190814 4.13291350 1.36 1.27 0 0 0 0 4.18190337  
4.37021420 1.19 0 2.3333 0 4.50919063 0 4.68230819 4.29135036 1.27 1.44 1.27 1.27 4.21140803 0  
4.70411115 4.67140220 4.12041319 4.190814 3.135758 0 1.43 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0  
2.3333 0 4.50710411 1.15 0 4.67140220 4.12041319 4.190814 4.13372913 2.5036 1.27 0 0 0 0 4.18190337  
4.37021420 1.19 0 2.3333 0 3.509163 0 4.66170400 2.1904 0 3.130422 0 4.18191720 4.2192017 2.437 0  
4.64111114 2.2218 0 3.241420 0 2.1914 0 4.2170400 2.1904 0 1.0 0 3.130422 0 4.12001907 4.4120019  
4.8020011 0 4.18191720 4.2192017 1.4 0 2.124 0 4.4131904 4.17081306 0 4.15171415 4.4171908 2.418 0  
4.8131904 4.17000219 4.8210411 4.24632913 2.5036 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0  
3.509263 0 4.67081815 3.110024 0 4.18191720 4.2192017 2.437 0 4.67081815 4.11002418 0 3.190704 0  
4.3041900 3.81118 0 2.1405 0 1.0 0 4.18041104 4.2190403 0 4.18191720 4.2192017 4.4632913 2.5036  
1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 3.509363 0 4.84150300 2.1904 0 4.18191720  
4.2192017 1.4 0 4.15171415 4.4171908 3.41837 0 4.64111114 2.2218 0 3.241420 0 2.1914 0 4.20150300  
2.1904 0 3.190704 0 4.15171415 4.4171908 2.418 0 2.1405 0 2.13 0 4.4230818 4.19081306 0 4.18191720  
4.2192017 4.4632913 2.5036 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 3.509463 0 4.67041104  
2.1904 0 4.18191720 4.2192017 2.437 0 4.67041104 3.190418 0 2.13 0 4.4230818 4.19081306 0  
4.18191720 4.2192017 1.4 0 4.5171412 0 3.190704 0 4.18241819 4.4126329 3.135036 1.27 0 0 0 0  
4.18190337 4.37021420 1.19 0 2.3333 0 3.509563 0 4.66070013 2.604 0 4.18191720 4.2192017 1.4 0  
3.726737 0 4.66070013 3.60418 0 3.190704 0 2.7267 0 2.1405 0 2.13 0 4.4230818 4.19081306 0  
4.18191720 4.2192017 2.431 0 4.4131820 4.17081306 0 2.1314 0 4.3201511 4.8020019 4.4186329  
3.135036 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 3.509663 0 4.82002104 0 4.18191720  
4.2192017 1.4 0 2.1914 0 4.5081104 1.37 0 4.82002104 1.18 0 3.190704 0 4.3041900 3.81118 0 2.1405 0  
1.0 0 4.18041104 4.2190403 0 4.18191720 4.2192017 1.4 0 2.1914 0 1.0 0 4.73827877 0 4.5081104  
4.63291350 1.36 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 3.509763 0 4.75140003 0  
4.18191720 4.2192017 1.4 0 4.5171412 0 4.5081104 1.37 0 4.75140003 1.18 0 1.0 0 4.18191720 4.2192017  
1.4 0 4.5171412 0 1.0 0 4.73827877 0 4.5081104 0 3.1303 0 4.30318 0 2.819 0 2.1914 0 3.190704 0  
4.18241819 4.4126329 3.135036 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 3.509863 0

4.67041104 2.1904 0 1.0 0 4.15171415 4.4171924 0 4.5171412 0 4.18191720 4.2192017 2.437 0  
4.67041104 3.190418 0 1.0 0 4.18150402 4.8050802 0 4.15171415 4.4171924 0 4.5171412 0 2.13 0  
4.4230818 4.19081306 0 4.18191720 4.2192017 4.4632913 2.5036 1.27 0 0 0 0 4.18190337 4.37021420  
1.19 0 2.3333 0 3.509963 0 4.71041115 0 4.3140220 4.12041319 4.190814 2.1337 0 4.67081815  
4.11002418 0 4.19070818 0 4.7041115 0 4.3140220 4.12041319 4.190814 4.13632913 2.5036 1.27 0 0 0 0  
4.18190337 4.37021420 1.19 0 2.3333 0 4.50919063 0 4.68230819 1.37 0 4.68230819 1.18 0 3.190704 0  
4.151511 4.8020019 4.8141363 4.29135036 1.27 1.44 1.27 1.27 4.9181413 0 4.6041984 4.18041779  
4.17141504 4.17190804 3.185758 0 1.43 1.27 0 0 0 0 4.9181413 0 4.20180417 4.79171415 2.1836 1.27 0 0  
0 0 4.18190337 4.37181917 3.81306 0 4.10042436 1.27 0 0 0 0 4.18190337 4.37181917 3.81306 0  
4.21001120 2.436 1.27 0 0 0 0 4.1141411 0 4.1141411 4.4001385 4.112004 1.36 1.27 0 0 0 0 3.81319 0  
4.8131985 4.112004 1.36 1.27 1.27 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0 4.50681319 2.417 0  
3.190704 0 4.15171415 4.4171908 2.418 0 3.51417 0 3.190704 0 4.18191720 4.2192017 2.463 0  
4.83241504 0 4.45031413 2.445 0 4.22070413 0 4.5081308 4.18070403 2.6350 0 2.3333 0 4.18190337  
4.37041303 2.1136 1.27 0 0 0 0 4.22070811 1.4 0 4.57191720 2.458 0 1.43 1.27 0 0 0 0 0 0 0 0 4.18190337  
4.37021420 1.19 0 2.3333 0 4.50681319 2.417 0 4.15171415 4.4171924 0 4.13001204 0 3.571417 0  
4.19241504 0 4.45031413 2.445 0 2.1914 0 4.5081308 4.18075837 0 2.5036 1.27 0 0 0 0 0 0 0 0  
4.18190337 4.37060419 4.11081304 4.57181903 4.37370208 2.1331 0 4.10042458 1.36 1.27 0 0 0 0 0 0 0 0 0  
2.805 0 4.57100424 0 2.6262 0 4.50031413 3.45058 0 4.1170400 2.1036 1.27 1.27 0 0 0 0 0 0 0 0  
4.18190337 4.37021420 1.19 0 2.3333 0 4.50681319 2.417 0 4.15171415 4.4171924 0 4.19241504 0  
4.57181917 4.8130631 0 4.8131931 0 4.1141411 2.5837 0 2.5036 1.27 0 0 0 0 0 0 0 4.18190337  
4.37181917 3.81306 0 4.19241504 1.36 1.27 0 0 0 0 0 0 4.18190337 4.37060419 4.11081304  
4.57181903 4.37370208 2.1331 0 4.19241504 2.5836 1.27 1.27 0 0 0 0 0 0 0 2.805 0 4.57192415 1.4 0  
2.6262 0 4.50181917 4.8130650 1.58 0 1.43 1.27 0 0 0 0 0 0 0 0 0 0 4.18190337 4.37021420 1.19 0  
2.3333 0 4.50681319 2.417 0 4.18191708 2.1306 0 4.21001120 1.4 0 4.57192415 1.4 0 4.45687767 1.45 0  
2.1914 0 4.5081308 2.1807 0 4.12201119 4.8591108 2.1304 0 4.8131520 3.195837 0 2.5036 1.27 0 0 0 0  
0 0 0 0 0 0 0 4.18190337 4.37181917 3.81306 0 4.11081304 1.36 1.27 0 0 0 0 0 0 0 0 0 0 0 4.21001120  
4.4630211 4.40011757 2.5836 1.27 0 0 0 0 0 0 0 0 0 0 4.22070811 1.4 0 4.57181903 4.37370604  
4.19110813 4.4571819 4.3373702 3.81331 0 4.11081304 2.5858 0 1.43 1.27 0 0 0 0 0 0 0 0 0 0 0 0 0  
2.805 0 4.57110813 1.4 0 2.6262 0 4.50687767 2.5058 0 4.1170400 2.1036 1.27 0 0 0 0 0 0 0 0 0 0 0 0  
0 4.21001120 1.4 0 2.6162 0 4.11081304 0 1.61 0 4.50291350 1.36 1.27 0 0 0 0 0 0 0 0 0 0 0 1.44 1.27 0 0  
0 0 0 0 0 0 0 0 0 4.20180417 4.79171415 4.18421004 2.2441 0 1.62 0 4.21001120 2.436 1.27 0 0 0 0 0 0  
0 1.44 0 4.4111804 0 2.805 0 4.57192415 1.4 0 2.6262 0 4.50081319 2.5058 0 1.43 1.27 0 0 0 0 0 0 0 0 0  
0 4.18190337 4.37021420 1.19 0 2.3333 0 4.50681319 2.417 0 4.8131904 3.60417 0 4.21001120 2.437 0  
2.5036 1.27 0 0 0 0 0 0 0 0 0 4.18190337 4.37020813 0 2.3434 0 4.8131985 4.112004 1.36 1.27 0 0 0 0  
0 0 0 0 0 0 0 4.20180417 4.79171415 4.18421004 2.2441 0 1.62 0 4.8131985 4.112004 1.36 1.27 0 0 0 0 0  
0 0 0 0 0 0 4.18190337 4.37020813 4.63080613 4.14170457 2.5836 0 2.3232 0 4.72061314 2.1704 0  
3.190704 0 4.13042211 3.81304 0 4.2070017 4.21904 1.17 0 4.11040519 0 2.813 0 3.190704 0 4.8131520  
1.19 0 4.1200505 2.417 1.27 0 0 0 0 0 0 1.44 0 4.4111804 0 2.805 0 4.57192415 1.4 0 2.6262 0  
4.50011414 3.115058 0 1.43 1.27 0 0 0 0 0 0 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0  
4.50681319 2.417 0 4.1141411 3.40013 0 4.21001120 1.4 0 2.5790 0 3.51417 0 4.5001118 2.431 0 1.91 0  
3.51417 0 4.19172004 2.5837 0 2.5036 1.27 0 0 0 0 0 0 0 0 0 0 4.18190337 4.37020813 0 2.3434 0  
4.1141411 4.4001385 4.112004 1.36 1.27 0 0 0 0 0 0 0 0 0 4.20180417 4.79171415 4.18421004  
2.2441 0 1.62 0 4.1141411 4.4001385 4.112004 1.36 1.27 0 0 0 0 0 0 0 0 0 0 4.18190337 4.37020813  
4.63080613 4.14170457 2.5836 0 2.3232 0 4.72061314 2.1704 0 3.190704 0 4.13042211 3.81304 0  
4.2070017 4.21904 1.17 0 4.11040519 0 2.813 0 3.190704 0 4.8131520 1.19 0 4.1200505 2.417 1.27 0 0 0 0  
0 0 0 0 1.44 0 4.4111804 0 1.43 1.27 0 0 0 0 0 0 0 0 0 0 4.18190337 4.37021420 1.19 0 2.3333 0  
4.50721321 4.110803 0 4.19241504 1.63 0 4.79110400 2.1804 0 4.4131904 1.17 0 4.45181917 4.8130645  
1.31 0 4.45081319 2.4531 0 2.1417 0 4.45011414 4.11456350 0 2.3333 0 4.18190337 4.37041303 2.1136  
1.27 0 0 0 0 0 0 0 1.44 1.27 0 0 0 0 0 1.44 1.27 1.27 0 0 0 0 4.17041920 2.1713 0 4.20180417 4.79171415  
2.1836 1.27 1.44 1.27 1.27 3.81319 0 4.12000813 2.5758 0 1.43 1.27 0 0 0 0 3.191724 0 1.43 1.27 0 0 0 0







4.82111419 0 4.15141808 4.19081413 1.18 1.27 4.18111419 3.916023 0 1.62 0 4.57181100 4.1601104  
 4.13061907 0 1.59 0 4.18111419 4.60110413 4.6190758 0 1.32 0 1.93 1.27 4.18111419 3.926023 0 1.62 0  
 1.92 0 1.56 0 4.57181100 4.1601104 4.13061907 0 1.59 0 4.18111419 4.60110413 4.6190758 0 1.32 0 1.93  
 1.27 1.27 1.39 0 4.66170400 2.1904 0 4.5080620 2.1704 0 3.1303 0 4.230818 1.27 4.5080631 0 2.23 0 1.62  
 0 4.15111963 4.18200115 4.11141918 4.57050806 4.18082504 4.62579190 1.31 0 3.965858 1.27 1.27 1.39  
 0 4.66170400 2.1904 0 3.190704 0 4.18110001 0 2.18 0 1.0 0 4.17040219 4.130611 1.4 1.27 4.18110001 0  
 1.62 0 4.15001902 4.7041863 4.81040219 4.130611 4.4575790 1.31 0 3.905831 0 4.18110001 4.60110413  
 4.6190731 0 4.18110001 4.60190708 4.2101304 3.181831 0 4.11081304 4.22080319 4.7629131 0  
 4.4030604 4.2141114 4.17624501 4.11000210 2.4531 0 4.5000204 4.2141114 4.17624506 4.17002445  
 1.58 1.27 4.236300 4.3036015 4.190207 4.57181100 2.158 1.27 1.27 1.39 0 4.66170400 2.1904 0 3.190704  
 0 4.5081718 1.19 0 4.18111419 1.27 4.18111419 1.91 0 1.62 0 4.15001902 4.7041863 4.81040219  
 4.130611 4.4575718 4.11141991 3.602331 0 4.18110001 4.60190708 4.2101304 2.1818 0 1.59 0  
 4.18111419 4.60030415 4.19075831 0 4.18111419 4.60110413 4.6190731 0 4.18111419 4.60030415  
 3.190731 0 4.11081304 4.22080319 4.7629131 0 4.4030604 4.2141114 4.17624501 4.11000210 2.4531 0  
 4.5000204 4.2141114 4.17624522 4.7081904 2.4558 1.27 4.236300 4.3036015 4.190207 4.57181114  
 3.199158 1.27 1.27 1.39 0 4.66170400 2.1904 0 3.190704 0 4.18040214 2.1303 0 4.18111419 1.27  
 4.18111419 1.92 0 1.62 0 4.15001902 4.7041863 4.81040219 4.130611 4.4575718 4.11141992 3.602331 0  
 4.18110001 4.60190708 4.2101304 2.1818 0 1.59 0 4.18111419 4.60030415 4.19075831 0 4.18111419  
 4.60110413 4.6190731 0 4.18111419 4.60030415 3.190731 0 4.11081304 4.22080319 4.7629131 0  
 4.4030604 4.2141114 4.17624501 4.11000210 2.4531 0 4.5000204 4.2141114 4.17624522 4.7081904  
 2.4558 1.27 4.236300 4.3036015 4.190207 4.57181114 3.199258 1.27 1.27 1.39 0 3.820419 0 3.190704 0  
 4.11081208 2.1918 0 3.1303 0 4.181504 2.219 0 4.17001908 1.14 1.27 4.236318 4.4196023 4.11081257  
 4.59919031 0 4.18110001 4.60110413 3.61907 0 1.61 0 3.919058 1.27 4.236318 4.4196024 4.11081257  
 4.59919031 0 4.18110001 4.60190708 4.2101304 2.1818 0 1.61 0 3.919058 1.27 4.236318 4.4196000  
 4.18150402 4.19574504 4.16200011 2.4558 1.27 1.27 1.39 0 3.640303 0 4.11000104 2.1118 0 3.1303 0  
 4.19081911 1.4 1.27 4.236318 4.4196023 4.11000104 4.11574575 4.4130619 1.7 0 4.57121258 2.4558 1.27  
 4.236318 4.4196024 4.11000104 4.11574583 4.7080210 4.13041818 0 4.57121258 2.4558 1.27 4.236318  
 4.4196019 4.8191104 4.57458219 4.81311 3.41818 0 4.82190404 1.11 0 4.82110001 0 4.22081907 0  
 4.82111419 3.184558 1.27 1.27 1.39 0 4.82071422 0 3.190704 0 4.15111419 1.27 4.15111963 4.6170803  
 4.57831720 2.458 1.27 4.15111963 4.18071422 2.5758 1.27 1.27 4.68132012 4.4170019 4.14176315 1.24 0  
 1.47 1.27 1.27 4.8121514 2.1719 0 4.12001915 4.11141911 4.8016315 4.24151114 1.19 0 2.18 0 3.151119  
 1.27 4.5171412 0 4.12151160 4.19141411 4.10081918 4.63121511 4.14199303 4.63001719 2.9303 0  
 4.8121514 2.1719 0 4.79141124 4.93676614 4.11110402 4.19081413 1.27 4.8121514 2.1719 0 4.13201215  
 1.24 0 2.18 0 2.1315 1.27 1.27 1.39 0 4.82110001 0 4.3081204 4.13180814 2.1318 1.27 4.18110001  
 4.60110413 3.61907 0 1.62 0 3.919395 1.27 4.18110001 4.60220803 2.1907 0 1.62 0 2.9693 1.27  
 4.18110001 4.60190708 4.2101304 2.1818 0 1.62 0 1.96 1.27 1.27 1.39 0 4.82111419 0 4.3081204  
 4.13180814 2.1318 1.27 4.18111419 4.60110413 3.61907 0 1.62 0 1.95 1.27 4.18111419 4.60220803  
 2.1907 0 1.62 0 2.9693 1.27 4.18111419 4.60030415 2.1907 0 1.62 0 1.93 1.27 1.27 1.39 0 4.82111419 0  
 4.15141808 4.19081413 1.18 1.27 4.18111419 3.916023 0 1.62 0 4.57181100 4.1601104 4.13061907 0 1.59  
 0 4.18111419 4.60110413 4.6190758 0 1.32 0 1.93 1.27 4.18111419 3.926023 0 1.62 0 1.92 0 1.56 0  
 4.57181100 4.1601104 4.13061907 0 1.59 0 4.18111419 4.60110413 4.6190758 0 1.32 0 1.93 1.27  
 4.18111419 2.6024 0 1.62 0 1.90 1.27 4.18111419 2.6025 0 1.62 0 4.18110001 4.60190708 4.2101304  
 2.1818 0 1.59 0 4.18111419 4.60030415 2.1907 1.27 1.27 1.39 0 4.66170400 2.1904 0 4.5080620 2.1704  
 1.27 3.50806 0 1.62 0 4.15111963 4.5080620 4.17045758 1.27 2.23 0 1.62 0 4.5080663 4.30360  
 4.18200115 4.11141957 4.91919131 0 4.15171409 4.4021908 4.14136245 4.93034558 1.27 1.27 1.39 0  
 4.66170400 2.1904 0 3.190704 0 4.21041719 4.8020418 0 2.1405 0 3.190704 0 4.18110001 1.27  
 4.21041719 4.8020418 0 1.62 0 4.13156300 4.17170024 2.5742 1.27 0 0 0 3.429031 0 2.9031 0 3.904131  
 0 4.42181100 4.1601104 4.13061907 1.31 0 2.9031 0 3.904131 0 4.42181100 4.1601104 4.13061907 1.31 0  
 4.18110001 4.60220803 3.190731 0 3.904131 0 3.429031 0 4.18110001 4.60220803 3.190731 0 3.904131 0  
 0 1.39 0 4.65141919 2.1412 0 4.5000204 1.27 0 0 0 0 3.429031 0 2.9031 0 4.18110001 4.60190708

4.2101304 4.18184131 0 4.42181100 4.1601104 4.13061907 1.31 0 2.9031 0 4.18110001 4.60190708  
4.2101304 4.18184131 0 4.42181100 4.1601104 4.13061907 1.31 0 4.18110001 4.60220803 3.190731 0  
4.18110001 4.60190708 4.2101304 4.18184131 0 3.429031 0 4.18110001 4.60220803 3.190731 0  
4.18110001 4.60190708 4.2101304 3.181841 0 0 1.39 0 3.831415 0 4.5000204 1.27 2.4158 1.27 1.27 1.39 0  
4.67040508 2.1304 0 3.190704 0 1.96 0 4.5000204 1.18 0 2.1405 0 3.190704 0 4.18110001 1.27 4.5000204  
1.18 0 1.62 0 1.42 1.27 0 0 0 4.42210417 4.19080204 4.18429041 1.31 0 4.21041719 4.8020418  
4.42914131 0 4.21041719 4.8020418 4.42954131 0 4.21041719 4.8020418 4.42944141 1.31 0 0 1.39 0  
4.69171413 1.19 0 4.5000204 1.27 0 0 0 4.42210417 4.19080204 4.18429141 1.31 0 4.21041719  
4.8020418 4.42924131 0 4.21041719 4.8020418 4.42964131 0 4.21041719 4.8020418 4.42954141 1.31 0 0  
1.39 0 4.81080607 1.19 0 4.5000204 1.27 0 0 0 4.42210417 4.19080204 4.18429241 1.31 0 4.21041719  
4.8020418 4.42934131 0 4.21041719 4.8020418 4.42974131 0 4.21041719 4.8020418 4.42964141 1.31 0 0  
1.39 0 4.65000210 0 4.5000204 1.27 0 0 0 4.42210417 4.19080204 4.18429341 1.31 0 4.21041719  
4.8020418 4.42904131 0 4.21041719 4.8020418 4.42944131 0 4.21041719 4.8020418 4.42974141 1.31 0 0  
1.39 0 4.75040519 0 4.5000204 1.27 0 0 0 4.42210417 4.19080204 4.18429041 1.31 0 4.21041719  
4.8020418 4.42914131 0 4.21041719 4.8020418 4.42924131 0 4.21041719 4.8020418 4.42934141 1.31 0 0  
1.39 0 4.65141919 2.1412 0 4.5000204 1.27 0 0 0 4.42210417 4.19080204 4.18429441 1.31 0 4.21041719  
4.8020418 4.42954131 0 4.21041719 4.8020418 4.42964131 0 4.21041719 4.8020418 4.42974141 1.31 0 0  
1.39 0 3.831415 0 4.5000204 1.27 1.41 1.27 1.39 0 3.640303 0 3.190704 0 4.5000204 1.18 0 2.1405 0  
3.190704 0 4.18110001 0 2.1914 0 3.190704 0 4.15111419 1.27 4.236300 4.3036002 4.14111104  
4.2190814 4.13930357 4.79141124 4.93676614 4.11110402 4.19081413 4.57050002 3.41831 0 4.5000204  
4.2141114 4.17186245 4.6170024 2.4531 0 4.11081304 4.22080319 4.7186291 1.31 0 4.4030604  
4.2141114 4.17186245 4.1110002 3.104531 0 4.111507 4.629063 3.955858 1.27 1.27 1.39 0 4.67040508  
2.1304 0 3.190704 0 4.18111419 1.18 0 2.18 0 4.18120011 3.110417 0 4.1142304 1.18 0 3.1303 0 3.303 0  
4.19070412 0 2.1914 0 3.190704 0 4.15111419 1.27 4.18111419 4.91602104 4.17190802 2.418 0 1.62 0  
4.13156300 4.17170024 2.5742 1.27 0 0 0 4.42181114 4.19916023 1.31 0 4.18111419 3.602431 0  
4.18111419 4.60254131 0 4.42181114 4.19916023 0 1.61 0 4.18111419 4.60110413 4.6190731 0  
4.18111419 3.602431 0 4.18111419 4.60254131 0 4.42181114 4.19916023 0 1.61 0 4.18111419  
4.60110413 4.6190731 0 4.18111419 2.6024 0 1.61 0 4.18111419 4.60220803 3.190731 0 4.18111419  
4.60254131 0 4.42181114 4.19916023 1.31 0 4.18111419 2.6024 0 1.61 0 4.18111419 4.60220803  
3.190731 0 4.18111419 4.60254131 1.27 0 0 0 0 4.42181114 4.19916023 1.31 0 4.18111419 3.602431 0  
4.18110001 4.60190708 4.2101304 4.18184131 0 4.42181114 4.19916023 0 1.61 0 4.18111419  
4.60110413 4.6190731 0 4.18111419 3.602431 0 4.18110001 4.60190708 4.2101304 4.18184131 0  
4.42181114 4.19916023 0 1.61 0 4.18111419 4.60110413 4.6190731 0 4.18111419 2.6024 0 1.61 0  
4.18111419 4.60220803 3.190731 0 4.18110001 4.60190708 4.2101304 4.18184131 0 4.42181114  
4.19916023 1.31 0 4.18111419 2.6024 0 1.61 0 4.18111419 4.60220803 3.190731 0 4.18110001  
4.60190708 4.2101304 3.181841 1.27 2.4158 1.27 1.27 4.18111419 4.92602104 4.17190802 2.418 0 1.62 0  
4.13156300 4.17170024 2.5742 1.27 0 0 0 4.42181114 4.19926023 1.31 0 4.18111419 3.602431 0  
4.18111419 4.60254131 0 4.42181114 4.19926023 0 1.61 0 4.18111419 4.60110413 4.6190731 0  
4.18111419 3.602431 0 4.18111419 4.60254131 0 4.42181114 4.19926023 0 1.61 0 4.18111419  
4.60110413 4.6190731 0 4.18111419 2.6024 0 1.61 0 4.18111419 4.60220803 3.190731 0 4.18111419  
4.60254131 0 4.42181114 4.19926023 1.31 0 4.18111419 2.6024 0 1.61 0 4.18111419 4.60220803  
3.190731 0 4.18111419 4.60254131 1.27 0 0 0 0 4.42181114 4.19926023 1.31 0 4.18111419 3.602431 0  
4.18110001 4.60190708 4.2101304 4.18184131 0 4.42181114 4.19926023 0 1.61 0 4.18111419  
4.60110413 4.6190731 0 4.18111419 3.602431 0 4.18110001 4.60190708 4.2101304 4.18184131 0  
4.42181114 4.19926023 0 1.61 0 4.18111419 4.60110413 4.6190731 0 4.18111419 2.6024 0 1.61 0  
4.18111419 4.60220803 3.190731 0 4.18110001 4.60190708 4.2101304 4.18184131 0 4.42181114  
4.19926023 1.31 0 4.18111419 2.6024 0 1.61 0 4.18111419 4.60220803 3.190731 0 4.18110001  
4.60190708 4.2101304 3.181841 1.27 2.4158 1.27 1.27 4.18111419 4.60050002 3.41891 0 1.62 0 1.42 1.27  
0 0 0 0 4.42181114 4.19916021 4.4171908 4.2041842 3.904131 0 4.18111419 4.91602104 4.17190802  
4.4184291 2.4131 0 4.18111419 4.91602104 4.17190802 4.4184295 2.4131 0 4.18111419 4.91602104

4.17190802 4.4184294 3.414131 1.27 0 0 0 0 4.42181114 4.19916021 4.4171908 4.2041842 3.914131 0  
4.18111419 4.91602104 4.17190802 4.4184292 2.4131 0 4.18111419 4.91602104 4.17190802 4.4184296  
2.4131 0 4.18111419 4.91602104 4.17190802 4.4184295 3.414131 1.27 0 0 0 0 4.42181114 4.19916021  
4.4171908 4.2041842 3.924131 0 4.18111419 4.91602104 4.17190802 4.4184293 2.4131 0 4.18111419  
4.91602104 4.17190802 4.4184297 2.4131 0 4.18111419 4.91602104 4.17190802 4.4184296 3.414131  
1.27 0 0 0 0 4.42181114 4.19916021 4.4171908 4.2041842 3.934131 0 4.18111419 4.91602104 4.17190802  
4.4184290 2.4131 0 4.18111419 4.91602104 4.17190802 4.4184294 2.4131 0 4.18111419 4.91602104  
4.17190802 4.4184297 3.414131 1.27 0 0 0 0 4.42181114 4.19916021 4.4171908 4.2041842 3.904131 0  
4.18111419 4.91602104 4.17190802 4.4184291 2.4131 0 4.18111419 4.91602104 4.17190802 4.4184292  
2.4131 0 4.18111419 4.91602104 4.17190802 4.4184293 3.414131 1.27 0 0 0 0 4.42181114 4.19916021  
4.4171908 4.2041842 3.944131 0 4.18111419 4.91602104 4.17190802 4.4184295 2.4131 0 4.18111419  
4.91602104 4.17190802 4.4184296 2.4131 0 4.18111419 4.91602104 4.17190802 4.4184297 2.4141 1.27  
1.41 1.27 1.27 4.18111419 4.60050002 3.41892 0 1.62 0 1.42 1.27 0 0 0 0 4.42181114 4.19926021  
4.4171908 4.2041842 3.904131 0 4.18111419 4.92602104 4.17190802 4.4184291 2.4131 0 4.18111419  
4.92602104 4.17190802 4.4184295 2.4131 0 4.18111419 4.92602104 4.17190802 4.4184294 3.414131  
1.27 0 0 0 0 4.42181114 4.19926021 4.4171908 4.2041842 3.914131 0 4.18111419 4.92602104 4.17190802  
4.4184292 2.4131 0 4.18111419 4.92602104 4.17190802 4.4184296 2.4131 0 4.18111419 4.92602104  
4.17190802 4.4184295 3.414131 1.27 0 0 0 0 4.42181114 4.19926021 4.4171908 4.2041842 3.924131 0  
4.18111419 4.92602104 4.17190802 4.4184293 2.4131 0 4.18111419 4.92602104 4.17190802 4.4184297  
2.4131 0 4.18111419 4.92602104 4.17190802 4.4184296 3.414131 1.27 0 0 0 0 4.42181114 4.19926021  
4.4171908 4.2041842 3.934131 0 4.18111419 4.92602104 4.17190802 4.4184290 2.4131 0 4.18111419  
4.92602104 4.17190802 4.4184294 2.4131 0 4.18111419 4.92602104 4.17190802 4.4184297 3.414131  
1.27 0 0 0 0 4.42181114 4.19926021 4.4171908 4.2041842 3.904131 0 4.18111419 4.92602104 4.17190802  
4.4184291 2.4131 0 4.18111419 4.92602104 4.17190802 4.4184292 2.4131 0 4.18111419 4.92602104  
4.17190802 4.4184293 3.414131 1.27 0 0 0 0 4.42181114 4.19926021 4.4171908 4.2041842 3.944131 0  
4.18111419 4.92602104 4.17190802 4.4184295 2.4131 0 4.18111419 4.92602104 4.17190802 4.4184296  
2.4131 0 4.18111419 4.92602104 4.17190802 4.4184297 2.4141 1.27 1.41 1.27 1.27 4.236300 4.3036002  
4.14111104 4.2190814 4.13930357 4.79141124 4.93676614 4.11110402 4.19081413 4.57181114  
4.19600500 4.2041891 1.31 0 4.5000204 4.2141114 4.17186245 4.22070819 3.44531 0 4.11081304  
4.22080319 4.7186291 1.31 0 4.4030604 4.2141114 4.17186245 4.1110002 4.10455858 1.27 4.236300  
4.3036002 4.14111104 4.2190814 4.13930357 4.79141124 4.93676614 4.11110402 4.19081413  
4.57181114 4.19600500 4.2041892 1.31 0 4.5000204 4.2141114 4.17186245 4.22070819 3.44531 0  
4.11081304 4.22080319 4.7186291 1.31 0 4.4030604 4.2141114 4.17186245 4.1110002 4.10455858 1.27  
1.27 1.39 0 3.820419 0 3.190704 0 4.11081208 2.1918 0 3.1303 0 4.11000104 2.1118 1.27 4.236318  
4.4196023 4.11081257 3.429031 0 4.18110001 4.60110413 4.6190741 1.58 1.27 4.236318 4.4196024  
4.11081257 3.429031 0 4.18110001 4.60220803 4.19074158 1.27 4.236318 4.4196025 4.11081257  
3.429031 0 4.18110001 4.60190708 4.2101304 4.18184158 1.27 4.236318 4.4196023 4.11000104  
4.11574575 4.4130619 1.7 0 4.57121258 2.4558 1.27 4.236318 4.4196024 4.11000104 4.11574586  
4.8031907 0 4.57121258 2.4558 1.27 4.236318 4.4196025 4.11000104 4.11574583 4.7080210 4.13041818  
0 4.57121258 2.4558 1.27 4.236318 4.4196019 4.8191104 4.57459367 0 4.85080422 0 2.1405 0  
4.82190008 4.13110418 1.18 0 4.82190404 1.11 0 4.82190001 0 4.22081907 0 4.82111419 3.184558 1.27  
1.27 1.39 0 4.68162000 1.11 0 4.181504 2.219 0 4.17001908 1.14 1.27 4.236318 4.4196001 4.14236000  
4.18150402 4.19574218 4.11000160 4.11041306 3.190731 0 4.18110001 4.60220803 3.190731 0  
4.18110001 4.60190708 4.2101304 4.18184158 0 0 1.39 0 4.64181504 2.219 0 4.17001908 1.14 0 2.818 0  
4.91379137 1.91 1.27 1.27 1.39 0 4.82071422 0 3.190704 0 4.15111419 1.27 4.15111963 4.18071422  
2.5758 1.27 1.27 4.75726682 4.68778268 0 1.47 1.27 1.27 3.767283 0 4.75080204 3.131804 1.27 1.27  
4.66141524 4.17080607 1.19 0 3.570258 0 4.92909294 0 4.67787672 4.64876870 2.6768 1.27 1.27  
4.79041712 4.8181808 2.1413 0 2.818 0 4.7041704 2.124 0 4.6170013 4.19040331 0 4.5170404 0 2.1405 0  
4.2070017 3.60431 0 2.1914 0 3.1324 0 4.15041718 2.1413 0 4.14011900 4.8130813 1.6 0 1.0 0 4.2141524  
1.27 2.1405 0 4.19070818 0 4.18140519 4.22001704 0 3.1303 0 4.181814 4.2080019 2.403 0 4.3140220

4.12041319 4.190814 1.13 0 4.5081104 1.18 0 4.57190704 0 4.50821405 4.19220017 4.4505831 0 2.1914 0  
4.3040011 1.27 2.813 0 3.190704 0 4.82140519 4.22001704 0 4.22081907 3.142019 0 4.17041819  
4.17080219 4.8141331 0 4.8130211 4.20030813 1.6 0 4.22081907 3.142019 0 4.11081208 4.19001908  
2.1413 0 3.190704 0 4.17080607 2.1918 1.27 2.1914 0 4.20180431 0 4.2141524 1.31 0 4.12140308 3.52431  
0 4.12041706 2.431 0 4.15200111 4.8180731 0 4.3081819 4.17080120 3.190431 0 4.18200111 4.8020413  
3.180431 0 4.130332 2.1417 0 4.18041111 1.27 4.2141508 2.418 0 2.1405 0 3.190704 0 4.82140519  
4.22001704 1.31 0 3.1303 0 2.1914 0 4.15041712 2.819 0 4.15041718 3.141318 0 2.1914 0 4.22071412 0  
3.190704 0 4.82140519 4.22001704 0 2.818 1.27 4.5201713 4.8180704 1.3 0 2.1914 0 2.314 0 3.181431 0  
4.18200109 3.40219 0 2.1914 0 3.190704 0 4.5141111 4.14220813 1.6 0 4.2141303 4.8190814 3.131837  
1.27 1.27 3.830704 0 4.11421 1.4 0 4.2141524 4.17080607 1.19 0 4.13141908 2.204 0 3.1303 0 4.19070818  
0 4.15041712 4.8181808 2.1413 0 4.13141908 2.204 0 4.18070011 1.11 0 2.104 0 4.8130211 4.20030403 0  
2.813 0 3.1111 1.27 4.2141508 2.418 0 2.1417 0 4.18200118 4.19001319 3.80011 0 4.15141719 4.8141318  
0 2.1405 0 3.190704 0 4.82140519 4.22001704 1.63 1.27 1.27 3.837168 0 4.82786983 4.86648168 0 2.7282  
0 4.79817885 4.72676867 0 3.506482 0 4.72825031 0 4.86728371 3.788483 0 4.86648181 4.64778388 0  
2.7869 0 3.6477788 0 4.74727767 1.31 0 4.68877981 3.688282 0 2.7881 1.27 4.72767975 4.72686731 0  
4.72776675 4.84677277 1.70 0 3.658483 0 3.777883 0 4.75727672 3.836867 0 2.8378 0 3.837168 0  
4.86648181 4.64778372 2.6882 0 2.7869 0 4.76688166 4.71647783 4.64657275 4.72838831 1.27  
4.69728377 3.688282 0 3.697881 0 1.64 0 4.79648183 4.72668475 2.6481 0 4.79848179 3.788268 0  
3.647767 0 4.77787772 4.77698172 4.77706876 4.68778363 0 2.7277 0 2.7778 0 4.68856877 1.83 0  
4.82716475 1.75 0 3.837168 1.27 4.64848371 3.788182 0 2.7881 0 4.66787988 4.81727071 1.83 0  
4.71787567 3.688182 0 2.6568 0 4.75726465 2.7568 0 3.697881 0 3.647788 0 4.66756472 2.7631 0  
4.67647664 3.706882 0 2.7881 0 4.78837168 1.81 1.27 4.75726465 4.72757283 2.8831 0 4.86716883  
3.716881 0 2.7277 0 2.6477 0 4.64668372 2.7877 0 2.7869 0 4.66787783 4.81646683 1.31 0 4.83788183 0  
2.7881 0 4.78837168 4.81867282 2.6831 0 4.64817282 3.727770 0 4.69817876 1.31 1.27 3.788483 0  
2.7869 0 2.7881 0 2.7277 0 4.66787777 4.68668372 2.7877 0 4.86728371 0 3.837168 0 4.82786983  
4.86648168 0 2.7881 0 3.837168 0 3.848268 0 2.7881 0 4.78837168 1.81 0 4.67686475 4.72777082 0  
2.7277 0 3.837168 1.27 4.82786983 4.86648168 1.63 1.27

67797883 69788482 27 27 91 0 67641624 0 58680906 9200111 0 84151511 0 83200119 9151458 27 92 0  
82696567 77696412 3 27 93 0 19052020 16641624 27 94 0 18051720 9180512 5142018 64202419 27 95 0  
67151517 4091400 20051960 80252007 15146419 2419 0 58871517 1118 0 23092007 0 200804 0  
67641624 0 80181506 18011312 91406 0 69140708 140558 27 96 0 13010913 3161659 15212015  
21206415 24 27 97 0 13010913 64031615 0 58680119 0 0 83201820 3202117 518 0 77011400 7051858 27  
98 0 77050307 1140902 1126714 13162119 5186415 24 0 58771503 51258 27 99 0 67151315 21200517  
641624 0 58771503 51258 27 9290 0 69142112 5180119 15186415 24 0 58771503 51258 27 9291 0  
76736768 788368 27 60606059 60606059 60606059 60606059 60606059 60606059 60606059 60606059  
60606059 60606059 60606059 60606059 60606059 60606059 60606059 60606059 60606059 60606059  
6059 27 27 67641624 0 47 27 27 6181512 0 20110913 200517 0 9131614 1819 0 56 27 9131614 1819 0  
20110913 200517 0 118 0 2010 27 9131614 1819 0 18011403 1512 27 9131614 1819 0 1518 27 9131614  
1819 0 13012007 27 6181512 0 807375 0 9131614 1819 0 73130106 4 27 9131614 1819 0 19210215  
18150304 1918 27 9131614 1819 0 914 0 39 0 5241604 18091304 1419 27 9131614 1819 0 16250700 1304  
27 9131614 1819 0 9131614 18201208 1 27 9131614 1819 0 18051720 5192018 27 9131614 1819 0  
23050201 18152318 517 27 9131614 1819 0 31921 27 6181512 0 21994 0 9131614 1819 0 66050120  
20090620 12831520 15 27 9131614 1819 0 4012004 20091304 27 9131614 1819 0 192518 27 9131614  
1819 0 10191513 27 9131614 1819 0 1804 27 6181512 0 807375 0 9131614 1819 0 73130106 5711800 1  
27 9131614 1819 0 16090310 1204 27 6181512 0 20110913 200517 0 9131614 1819 0 19091315 12050408  
1121506 27 6181512 0 20110913 200517 0 9131614 1819 0 77051420 31 0 67080502 11022119 20151431  
0 66151511 5011485 117 0 40701517 0 87091403 152318 0 7982 27 39 0 6181512 0 20110913 200517 0  
9131614 1819 0 77051420 31 0 67080502 11022119 201513 0 40701517 0 13010378 82 27 39 0 6181512 0  
20110913 200517 0 9131614 1819 0 66151511 5011485 117 0 40701517 0 13010378 82 27 6181512 0  
20110913 20051863 3151214 18030814 15190517 0 9131614 1819 0 1191102 15121517 27 9131614 1819  
0 12150706 91406 27 9131614 1819 0 20091304 27 9131614 1819 0 16250120 20150720 8 27 9131614  
1819 0 11052501 15011803 27 6181512 0 20110913 200517 0 9131614 1819 0 6091204 4090111 1506 27  
6181512 0 807375 0 9131614 1819 0 73130106 58410 27 9131614 1819 0 13012015 12152011 9026415  
25161214 19 0 118 0 161219 27 6181512 0 13161260 20151511 11092018 64131611 15209403 0 9131614  
1819 0 65240518 9467 27 6181512 0 17091910 919 0 9131614 1819 0 81210113 20211366 9180320 919  
27 6181512 0 17091910 9206421 9192100 12092600 20091513 0 9131614 1819 0 3091802 21092060  
4180122 517 27 9131614 1819 0 19030804 13041800 22 27 9131614 1819 0 19030804 13041800  
23640511 5130513 2018 0 118 0 51212 27 6181512 0 19030804 13041800 22 0 9131614 1819 0 12150708  
2 27 6181512 0 9200517 20151511 18 0 9131614 1819 0 16181503 210319 27 6181512 0 13012015  
12152011 901 0 9131614 1819 0 16251611 1519 0 118 0 161219 27 9131614 1819 0 14211315 24 0 118 0  
1415 27 9131614 1819 0 7121501 27 6181512 0 3151211 5032008 151418 0 9131614 1819 0 4050600  
21122003 90319 27 6181512 0 13012015 12152011 9026400 14091300 20091513 0 9131614 1819 0  
70211402 65140912 1200914 13 27 9131614 1819 0 16192119 911 27 9131614 1819 0 19030804 4211204  
27 6181512 0 16012007 120901 0 9131614 1819 0 80012007 27 9131614 1819 0 19082119 911 27  
9131614 1819 0 9131614 18201208 2642119 911 27 39 0 9131614 1819 0 16250120 20150720 8 27 39 0  
6181512 0 807375 0 9131614 1819 0 73130106 531 0 5241604 18091304 1419 27 27 6181512 0 807375 0  
9131614 1819 0 73130106 531 0 73130106 5681800 22 27 9131614 1819 0 16121519 12256406 18011607  
61150209 5032018 0 118 0 714 27 6181512 0 80258119 95 0 9131614 1819 0 81208708 4070519 1931 0  
81206714 1804 27 6181512 0 13012015 12152011 9026401 1031104 14041963 2010310 5140460  
17209600 706 0 9131614 1819 0 70090720 18056700 14220118 81846506 6 0 118 0 70090720 18056700  
14220118 27 6181512 0 13012015 12152011 9026405 9072117 4 0 9131614 1819 0 70090720 1804 27  
6181512 0 80258119 96648119 87090406 52018 0 9131614 1819 0 81651615 12090300 20091513 31 0  
81870903 7052031 0 81866614 24760124 15212031 0 81760913 5690408 2031 0 81760101 51231 0  
81802118 8662119 20151431 0 81830317 15121264 18050131 0 81840523 20690408 19 27 15196404  
14220917 15144345 81846164 80734641 0 62 0 46162516 209645 27 20051315 12012004 18 0 62 0 43 27

0 0 0 0 51020118 9035137 0 43510903 5131 0 51140112 55131 0 51220111 21055141 31 27 0 0 0 0  
51040519 1091204 45137 0 43510903 5131 0 51140112 55131 0 51040518 3180915 20091513 5131 0  
51200912 5192000 13165141 27 44 27 27 16180913 20585150 51720517 4 0 918 0 0 0 12091919 0 1505 0  
151404 0 8211403 180503 0 4091919 9140319 0 6090511 418 0 23092007 913 0 200804 0 18050111 12 0  
1505 0 13012007 5130119 9030111 0 18051904 1180307 0 11403 0 4052204 12151612 5142037 27 27  
9263 0 65120704 2180108 2 0 71051512 5201824 27 9363 0 65120704 2180108 2 0 84151614 12150724  
27 9463 0 65120704 2180108 2 0 78211301 517 0 84080514 1824 27 9563 0 65120704 2180108 2 0  
67151301 9140119 15180902 18 27 9663 0 65140111 25200902 0 78211301 517 0 84080514 1824 27 9763  
0 65161611 90503 0 77012007 5130119 90318 27 9863 0 65161617 15240912 1200914 13 0 84080514  
1824 27 9963 0 65180919 8130519 902 0 71051512 5201824 27 10063 0 65192512 16201519 902 0  
65140111 25190918 27 929163 0 66091512 1200804 13012008 318 27 929263 0 66180108 3 0 84080514  
1824 27 929363 0 67011202 21122118 0 1505 0 86011808 1200914 1418 27 929463 0 67012004 7151824  
0 84080514 1824 27 929563 0 67080114 18 0 84080514 1824 27 929663 0 67150408 1406 0 84080514  
1824 27 929763 0 67151301 9140119 15180900 11 0 84151614 12150724 27 929863 0 67151301 9140119  
15180902 18 27 929963 0 67151315 120523 0 65140111 25190918 27 930063 0 67151315 21200119  
9151400 11 0 65120704 21800 27 939163 0 67151315 21200119 9151400 11 0 71051512 5201824 27  
939263 0 67151315 21200119 9151400 11 0 78211301 517 0 84080514 1824 27 939363 0 67151419  
181511 0 84080514 1824 27 939463 0 67182515 20150717 1160824 27 939563 0 68090605 5180513  
20090111 0 65120704 21800 27 939663 0 68090605 5180513 20090111 0 69172100 20091513 18 27  
939763 0 68090605 5180513 20090111 0 71051512 5201824 27 939863 0 68090605 5180513 20090111 0  
84151614 12150724 27 939963 0 68091902 18052004 0 71051512 5201824 27 940063 0 68091902  
18052004 0 77012007 5130119 90318 27 949163 0 68251400 13090300 11 0 83251919 51318 27 949263 0  
69121208 16200902 0 67211821 518 27 949363 0 69142112 5180119 92204 0 67151301 9140119  
15180902 18 27 949463 0 69180714 40902 0 84080514 1824 27 949563 0 69241604 18091304 14200111 0  
77012007 5130119 90318 27 949663 0 70091408 2004 0 71051512 5201824 27 949763 0 70122108 3 0  
68251400 13090318 27 949863 0 70152117 90517 0 65140111 25190918 27 949963 0 70180102 200111 0  
71051512 5201824 27 950063 0 70211402 20091513 111 0 65140111 25190918 27 959163 0 70212625 24  
0 77012007 5130119 90318 27 959263 0 71011304 0 84080514 1824 27 959363 0 71051512 5201808 2 0  
65140111 25190918 27 959463 0 71051512 5201808 2 0 84151614 12150724 27 959563 0 71180115 7 0  
84080514 1824 27 959663 0 71181520 15 0 84080514 1824 27 959763 0 72011812 15140902 0 65140111  
25190918 27 959863 0 72151314 12150708 30111 0 65120704 21800 27 959963 0 72151314 20151624 0  
84080514 1824 27 960063 0 72251604 18021511 902 0 71051512 5201824 27 969163 0 73140308  
4051402 4 0 71051512 5201824 27 969263 0 73140614 18130119 91513 0 84080514 1824 27 969363 0  
73142004 7180111 0 69172100 20091513 18 27 969463 0 73142004 7180101 1204 0 83251919 51318 27  
969563 0 75608407 5151824 27 969663 0 75141519 0 84080514 1824 27 969763 0 760904 0 65120704  
2180118 27 969863 0 760904 0 71181520 1618 27 969963 0 76091404 117 0 65120704 21800 27 970063 0  
76091404 117 0 80181506 18011312 91406 27 979163 0 76150708 2 27 979263 0 77011408 6151203 0  
84080514 1824 27 979363 0 77012007 5130119 9030111 0 66091511 150724 27 979463 0 77012007  
5130119 9030111 0 70091400 140304 27 979563 0 77012007 5130119 9030111 0 76150708 2 27 979663 0  
77012007 5130119 9030111 0 80082518 90318 27 979763 0 77012017 923 0 84080514 1824 27 979863 0  
77050118 211804 0 84080514 1824 27 979963 0 77150404 11 0 84080514 1824 27 980063 0 78151402  
15131320 20012008 2204 0 71051512 5201824 27 989163 0 78151411 9140500 17 0 68251400 13090318  
27 989263 0 78211301 517 0 84080514 1824 27 989363 0 78211304 18090300 11 0 65140111 25190918  
27 989463 0 79160517 1201517 0 65120704 2180118 27 989563 0 79160517 1201517 0 84080514 1824 27  
989663 0 79162008 13092600 20091513 27 989763 0 79180408 14011824 0 68090605 5180513 20090111  
0 69172100 20091513 18 27 989863 0 80011819 90111 0 68090605 5180513 20090111 0 69172100  
20091513 18 27 989963 0 80051802 15120119 91513 0 84080514 1824 27 990063 0 80051819 21180200  
20091513 0 84080514 1824 27 999163 0 80181501 1020911 92024 0 84080514 1824 27 999263 0  
81210113 202112 0 65120704 21800 27 999363 0 81210113 202112 0 71181520 1618 27 999463 0  
81210113 202112 0 84151614 12150724 27 999563 0 81210520 5091406 0 84080514 1824 27 999663 0

82050111 0 65140111 25190918 27 999763 0 82051617 5190513 20012008 1513 0 84080514 1824 27  
999863 0 82090512 1141408 113 0 71051512 5201824 27 999963 0 82091406 0 84080514 1824 27  
1000063 0 830519 0 84080514 1824 27 1009163 0 83091406 21120117 92024 0 84080514 1824 27  
1009263 0 83160502 20180111 0 84080514 1824 27 1009363 0 83200119 9192008 30111 0 77050307  
1140902 18 27 1009463 0 83201502 8011919 902 0 80181502 5191904 18 27 1009563 0 83251315  
12050319 902 0 71051512 5201824 27 1009663 0 83251919 51318 0 84080514 1824 27 1009763 0  
84051418 1517 0 65140111 25190918 27 1009863 0 84151614 12150708 30111 0 71181520 1618 27  
1009963 0 84151614 12150724 27 1010063 0 84181515 9030111 0 71051512 5201824 27 92919163 0  
87012204 12052018 0 11403 0 77211219 9180518 15122119 91513 0 65140111 25190918 27 27  
84080518 4 0 6090511 418 0 18051617 5190513 19 0 0 0 22011919 0 12011403 19030115 4 0 1505 0  
13012007 5130119 9030111 0 9141720 91824 0 11403 0 8012204 0 5242004 14190921 4 0 1161611  
9030119 9151418 0 1031814 1918 0 22011808 152118 0 19030904 14200905 902 0 11403 0 5140708  
14050517 91406 0 4151300 9141963 27 27 72051804 0 918 0 0 0 4051902 18091619 91513 0 1505 0  
20230511 2204 0 2181500 3 0 6090511 418 0 1505 0 19202103 24 0 20080119 0 5140314 13160118 18 0  
13011424 0 1505 0 200804 0 19160502 9011208 260503 0 1180500 18 0 1505 0 13012007 5130119  
9030111 0 18051904 1180307 0 11403 0 4052204 12151612 51419 0 10211919 0 12091919 50437 27 27  
9263 0 65120704 2180108 2 0 83201820 3202117 51937 0 84080918 0 6090511 3 0 9140311 21040518 0  
19202103 90518 0 913 0 1120704 2180108 2 0 7051512 5201824 31 0 1120704 2180108 2 0 20151614  
12150724 31 0 1120704 2180108 2 0 14211301 517 0 20080514 182531 0 11403 0 1120704 2180108 2 0  
3151301 9140119 15180902 1963 0 7319 0 9142214 12220518 0 200804 0 5241611 15180119 91513 0  
1505 0 13012007 5130119 9030111 0 19201820 3202117 518 0 20080119 0 11804 0 6211403 1130513  
20011211 24 0 1120704 2180108 2 0 913 0 14012020 180531 0 9142214 12220913 6 0 15160517 1200914  
1418 0 23092007 913 0 19052018 0 20080119 0 6151211 1522 0 19160502 9060902 0 1240914 131931 0  
12091104 0 7181520 161931 0 18091406 1931 0 11403 0 6090511 41963 0 84080518 4 0 19202103 90518  
0 11804 0 3182102 90111 0 61517 0 21140404 18192000 14040913 6 0 7051512 5201808 30111 0  
16181515 5182008 518 0 20081814 210707 0 1120704 2180108 2 0 5241617 5191908 15141963 27 27  
9363 0 78211301 517 0 84080514 182537 0 67151315 18091908 1406 0 1140111 25200902 0 14211301  
517 0 20080514 182531 0 1120704 2180108 2 0 14211301 517 0 20080514 182531 0 11403 0 1180919  
8130519 902 0 7051512 5201824 31 0 20080918 0 6090511 3 0 6150320 190518 0 1513 0 200804 0  
16181515 5182008 518 0 11403 0 18051200 20091513 19080915 18 0 1505 0 14211301 5181931 0  
16011819 9032111 1181224 0 200804 0 9142004 7051818 63 0 7319 0 21200911 9260518 0 20050307  
14091720 518 0 6181512 0 0 0 2181500 3 0 18011406 4 0 1505 0 13012007 5130119 9030111 0 4091902  
9161208 140518 0 2014 0 19151221 4 0 16181501 12051318 0 18051200 200503 0 2014 0 4092208  
19090208 12092024 31 0 3151406 18210513 3051931 0 11403 0 200804 0 4091919 18090220 20091513 0  
1505 0 16180912 51963 27 27 9463 0 67151301 9140119 15180902 18 0 11403 0 71180115 7 0 84080514  
182537 0 84080918 0 9140311 21040518 0 5142112 5180119 92204 0 3151301 9140119 15180902 1931 0  
1120704 2180108 2 0 3151301 9140119 15180902 1931 0 11403 0 7180115 7 0 20080514 182563 0 7319 0  
19202103 90518 0 3151301 9140119 15180900 11 0 19201820 3202117 518 0 11403 0 20080508 17 0  
1120704 2180108 2 0 16181515 5182008 518 0 2014 0 21140404 18192000 1403 0 3151405 9072117  
1200914 1418 0 11403 0 18051200 20091513 18 0 12091104 0 20081518 4 0 6152113 3 0 913 0 7180115 7  
0 19201820 3202117 51931 0 4051908 7141931 0 11403 0 3150404 1963 27 27 9563 0 84151614  
12150708 30111 0 83202103 9051937 0 84080918 0 6090511 3 0 3152204 1818 0 1180500 18 0 19210307  
0 118 0 1120704 2180108 2 0 20151614 12150724 31 0 4090605 5180513 20090111 0 20151614 12150724  
31 0 11403 0 7051512 5201808 2 0 20151614 12150724 63 0 7319 0 4050111 18 0 23092007 0 16181515  
5182008 518 0 20080119 0 11804 0 16180518 5182204 3 0 20081814 210707 0 4050614 18130119  
9151418 31 0 20230918 20091406 1931 0 11403 0 19201804 20030808 140718 0 1505 0 15021004  
3201931 0 9142204 19200906 1200913 6 0 3151402 5162018 0 12091104 0 3151419 9142108 202531 0  
3151315 1032013 5191931 0 11403 0 3151413 5032004 4140518 1963 27 27 9663 0 68090605 5180513  
20090111 0 71051512 5201824 0 11403 0 71051512 5201808 2 0 65140111 25190918 37 0 84080918 0  
5140314 13160118 190518 0 4090605 5180513 20090111 0 7051512 5201824 31 0 4090605 5180513

20090111 0 20151614 12150724 31 0 11403 0 7051512 5201808 2 0 1140111 25190918 63 0 7319 0  
9142214 12220518 0 200804 0 211904 0 1505 0 3011202 21122118 0 11403 0 1120704 21800 0 2014 0  
19202103 24 0 16181501 12051318 0 913 0 7051512 5201824 31 0 6150320 19091406 0 1513 0 3211821  
51931 0 19211805 1030518 31 0 11403 0 8090707 5186003 9130513 19091513 111 0 1140111 15072104  
1963 27 27 9763 0 65140111 25200902 111 0 83202103 9051937 0 70090511 418 0 12091104 0 3151315  
120523 0 1140111 25190918 31 0 6211402 20091513 111 0 1140111 25190918 31 0 11403 0 8011812  
15140902 0 1140111 25190918 0 6011211 0 21140404 17 0 20080918 0 3012004 7151824 63 0 84080524  
0 9142214 122204 0 200804 0 4052000 9120503 0 9142204 19200906 1200914 13 0 1505 0 6211402  
20091513 1931 0 20080508 17 0 19160102 51931 0 11403 0 15200804 17 0 18051200 200503 0 13012007  
5130119 9030111 0 5142008 20090518 31 0 16181521 9040913 6 0 0 0 4050515 0 21140404 18192000  
14040913 6 0 1505 0 20080508 17 0 2050800 22091517 0 11403 0 16181515 5182008 51963 27 27 9863 0  
65161611 90503 0 77012007 5130119 9030111 0 83030904 14030518 37 0 84080918 0 9140311  
21040518 0 1161611 90503 0 13012007 5130119 9031931 0 13012007 5130119 9030111 0 16082518  
9031931 0 13012007 5130119 9030111 0 6091400 14030531 0 11403 0 2091512 1200804 13012008  
31931 0 6150320 19091406 0 1513 0 200804 0 1161611 9030119 91513 0 1505 0 13012007 5130119  
9030111 0 13052007 150418 0 224 0 4090605 5180513 19 0 6090511 418 0 19210307 0 118 0 19030904  
140304 0 11403 0 5140708 14050517 9140763 0 84080918 0 1180500 0 1161611 90518 0 20080514  
18090518 0 11403 0 20050307 14091720 518 0 6181512 0 200804 0 16211804 0 16011819 18 0 1505 0  
13012007 5130119 90318 0 2014 0 16180102 20090300 11 0 16181501 12051318 63 27 27 9963 0  
67151315 21200119 9151400 11 0 77012007 5130119 9031937 0 84080918 0 6090511 3 0 3152204 1818 0  
3151315 21200119 9151400 11 0 1120704 2180131 0 3151315 21200119 9151400 11 0 7051512 5201824  
31 0 11403 0 14211304 18090300 11 0 1140111 25190918 63 0 7319 0 4050111 18 0 23092007 0  
13012007 5130119 9030111 0 18051904 1180307 0 913 0 1180500 18 0 20080119 0 18051720 91804 0  
12011806 5601902 11204 0 3151315 21200119 91513 0 11403 0 1120714 18092007 130902 0 16180502  
9190914 1431 0 15062004 13 0 61517 0 19091320 12012008 15141931 0 15162008 13092600 20091513  
1931 0 11403 0 3151315 120523 0 3011202 21120119 9151418 63 27 27 10063 0 68091902 18052004 0  
77012007 5130119 9031937 0 73140311 21040913 6 0 4091902 18052004 0 7051512 5201824 31 0  
3151301 9140119 15180900 11 0 20151614 12150724 31 0 11403 0 4091902 18052004 0 13012007  
5130119 90318 0 9201904 120631 0 20080918 0 6090511 3 0 9142204 19200906 1200518 0 13012007  
5130119 9030111 0 19201820 3202117 518 0 20080119 0 11804 0 6211403 1130513 20011211 24 0  
4091902 18052004 0 18012007 517 0 20080113 0 3151419 9142114 211963 0 7319 0 80118 0 1161611  
9030119 9151418 0 913 0 3151315 21200517 0 19030904 14030531 0 3182515 20150717 1160824 31 0  
11403 0 9140614 18130119 91513 0 20080514 182563 27 27 929163 0 68251400 13090300 11 0 83251919  
51318 0 11403 0 69180714 40902 0 84080514 182537 0 83202103 25091406 0 1180500 18 0 12091104 0  
4251400 13090300 11 0 19251919 5131931 0 3080114 18 0 20080514 182531 0 11403 0 5180714 40902 0  
20080514 182531 0 20080918 0 6090511 3 0 5241611 15180518 0 19251919 51318 0 20080119 0 5221511  
2204 0 15220517 0 20091304 0 1030314 18040913 6 0 2014 0 19160502 9060902 0 18211204 1963 0 7319  
0 5240112 9140518 0 81522 0 20080518 4 0 19251919 51318 0 2050800 220531 0 5221511 220531 0  
11403 0 18051915 151403 0 2014 0 22011808 152118 0 9141620 2018 0 15220517 0 200804 0 12151406 0  
20051812 63 27 27 929263 0 77012007 5130119 9030111 0 76150708 2 0 11403 0 70152113 4012008  
15141937 0 67152204 18091406 0 12150708 331 0 190519 0 20080514 182531 0 11403 0 13150404 11 0  
20080514 182531 0 20080918 0 6090511 3 0 19202103 90518 0 200804 0 6151812 111 0 2011908 18 0  
1505 0 13012007 5130119 9031963 0 7319 0 9142204 19200906 1200518 0 200804 0 16180913 3091611  
518 0 1505 0 13012007 5130119 9030111 0 18050118 15140913 731 0 200804 0 14012020 1804 0 1505 0  
13012007 5130119 9030111 0 15021004 3201931 0 11403 0 200804 0 20080514 18052008 30111 0  
21140404 18160913 14091406 18 0 1505 0 13012007 5130119 9030111 0 20080514 18090518 63 27 27  
929363 0 79162008 13092600 20091513 0 11403 0 67151419 181511 0 84080514 182537 0 69140314  
13160118 19091406 0 12091404 117 0 16181506 18011312 9140731 0 15162008 13092600 20091513 31  
0 3151419 181511 0 20080514 182531 0 11403 0 19251919 51318 0 20080514 182531 0 20080918 0  
6090511 3 0 918 0 3151402 5181404 3 0 23092007 0 6091403 91406 0 200804 0 2051919 0 16151918

9021204 0 19151220 20091513 0 2014 0 0 0 16181501 12051331 0 7092204 13 0 3151418 20180108  
142018 0 11403 0 15021004 3200921 51931 0 11403 0 3151419 18151211 91406 0 200804 0 2050800  
22091517 0 1505 0 4251400 130902 0 19251919 51318 0 913 0 113 0 15162008 130111 0 13011413 51863  
27 27 69010307 0 1505 0 20080518 4 0 6090511 418 0 918 0 22011919 0 11403 0 9142004 18031513  
14050319 18 0 23092007 0 13211219 9161204 0 1180500 18 0 1505 0 13012007 5130119 9031931 0  
9121220 19201800 20091406 0 200804 0 16181505 15211403 0 4051619 7 0 11403 0 2180500 42007 0  
1505 0 13012007 5130119 9030111 0 18051904 1180307 0 11403 0 4052204 12151612 5142063 27 27  
69142112 5180119 92204 0 77012007 5130119 9030111 0 70180112 5231517 10 0 1505 0 84081520  
7082037 27 27 91 0 42 0 68050608 14092008 151418 0 651403 0 79160517 1201517 18 0 41 27 27 91 0  
80181514 5 0 62 0 840804 0 1021904 140304 0 1505 0 4152101 2063 27 27 92 0 83151220 20091513 0 62  
0 8414 0 19151221 4 0 0 0 16181501 120512 0 23092007 152119 0 200804 0 3012118 4 0 1505 0 1141519  
8051863 27 27 93 0 80181501 120512 0 62 0 840804 0 16180518 5140304 0 1505 0 4152101 2063 27 27  
94 0 67151419 52419 0 62 0 79141224 0 151404 0 3151419 52419 0 918 0 5220517 0 16180518 5142063 0  
84080119 0 91931 0 15140304 0 20182104 31 0 1122300 2518 0 20182104 63 27 27 95 0 78211301 517 0  
62 0 64 0 16082518 9030111 0 19201820 3202117 531 0 16180518 51419 0 42104 0 2014 0 0 0 19200119  
4 0 20180113 19092008 151463 27 27 96 0 85140919 0 20180113 19092008 1513 0 62 0 840804 0  
12050118 19 0 5060604 3200921 4 0 3080113 704 0 1505 0 19200119 563 27 27 97 0 67151315 15211403  
0 20180113 19092008 1513 0 62 0 64 0 19051720 5140304 0 1505 0 21140919 0 20180113 19092008  
15141963 27 27 98 0 830519 0 62 0 6513 0 1181800 24 0 1505 0 14211301 5181963 27 27 99 0 770115 0  
62 0 791404 0 78211301 517 0 18051920 122018 0 913 0 1141519 80517 0 78211301 51831 0 224 0 0 0  
4050608 140503 0 85140919 0 1517 0 67151315 15211403 0 20180113 19092008 151463 27 27 9290 0  
67120108 12 0 62 0 64 0 13011663 27 27 9291 0 67151315 12052008 1513 0 62 0 80181514 5 0 1505 0  
67120108 1363 27 27 9292 0 651403 27 27 9293 0 881517 0 58692402 12211908 22051224 0 791858 27  
27 9294 0 781519 27 27 92 0 42 0 83200119 5130513 19 0 41 27 27 91 0 926496 0 61 0 926496 0 62 0  
926497 27 27 92 0 224 0 4050608 14092008 1513 27 27 93 0 92 0 1505 0 926496 0 11403 0 926497 0 91 0  
80118 0 1030808 5220503 0 92649291 27 27 93 0 42 0 83200119 5130513 19 0 41 27 27 91 0 5800 0 59 0  
258 0 918 0 16180912 4 27 27 92 0 8 0 62 0 0 27 27 93 0 8 0 62 0 1 27 27 94 0 83211312 1200914 13 0  
1505 27 27 95 0 5808 0 61 0 9258 27 27 96 0 94 0 95 0 6181512 0 92 0 2014 0 93 27 27 97 0 840804 0  
19172100 1804 0 18151519 0 1505 27 27 98 0 97335800 339358 27 27 99 0 4050308 130111 0 16011819 0  
1505 0 98 27 27 9290 0 69240311 21190921 51224 0 7981 27 27 9291 0 91 0 59 0 58040502 9130111 0  
16011819 0 1505 0 9958 27 27 9292 0 99 0 9290 0 9291 27 27 9293 0 98 0 16122118 0 1517 0 13091420  
18 0 929331 0 918 0 16180912 4 27 27 9294 0 26051814 0 918 0 12051918 0 20080113 0 9290 0 5172100  
11 0 2014 0 9292 0 23080902 7 0 918 0 12051918 0 20080113 0 9290 0 5172100 11 0 2014 0 151404 27 27  
9295 0 58609258 0 918 0 200804 0 19200515 0 22011220 4 0 1505 0 96 27 27 9296 0 5800 0 34 0 258 27  
27 9297 0 26051814 0 918 0 12051918 0 20080113 0 5800 0 59 0 258 0 23080902 7 0 918 0 12051918 0  
20080113 0 9290 0 5172100 11 0 2014 0 9292 27 27 9298 0 91 0 96 0 98 0 9292 0 9297 0 9293 0 9295 0  
9296 27 27 9299 0 69240502 212004 0 9298 0 118 0 200804 0 926496 0 9290 0 926497 27 27 9390 0 9299  
0 15212015 212018 0 58840804 0 14211301 517 0 9331 0 93 0 881517 0 96595150 5158 27 27 40200518  
19 27 39 0 68050608 1404 0 6211402 20091513 18 0 61517 0 22011808 152118 0 15160517 1200914 1418  
27 40505 0 12091919 61060911 5159803 9180502 20151824 63466445 5937 27 0 0 0 0 18052020 1813 0  
15196411 9192003 9185803 9180502 20151824 58 27 27 40505 0 7052060 19251919 5136108 14061557  
5937 27 0 0 0 0 18052020 1813 0 43 27 0 0 0 0 0 0 0 46031620 61160517 3051419 4637 0 16192119  
9126402 16216115 5180304 14205808 14200517 22011262 925931 27 0 0 0 0 0 0 0 46220917 20210111  
61130512 15182545 37 0 16192119 9126421 9182020 1126112 5131517 25585963 61011903 9032057  
5931 27 0 0 0 0 0 0 0 0 46040918 11612118 1070545 37 0 16192119 9126403 9191160 21190106 5584632  
46596460 1190408 3205858 27 0 0 0 0 44 27 27 40505 0 4152313 12150103 61060911 5582117 1231 0  
19012204 61160119 85937 27 0 0 0 0 18051915 15141904 0 62 0 18051720 5192018 64070519 58211811  
58 27 0 0 0 0 23092007 0 15160513 58190121 5611600 200831 0 46230245 58 0 118 0 637 27 0 0 0 0 0 0  
0 6642317 9200557 18051915 15141904 64031513 20051419 58 27 0 0 0 0 18052020 1813 0 6516814  
23141214 1040503 0 44211811 44 0 2014 0 44190121 5611600 20084550 27 27 40505 0 13152204



80252007 1513 0 677672 0 61517 0 87091403 152318 0 67151419 18151250 58 27 0 0 0 0 16180913  
20585183 251604 0 46080511 1645 0 2014 0 190504 0 1220108 12010211 4 0 3151312 1140418 31 0  
46052408 2045 0 2014 0 17210919 31 0 1517 0 16180505 923 0 23092007 0 465045 0 2014 0 5240502  
212004 0 80252007 1513 0 3150404 645158 27 0 0 0 16180913 20585183 251604 0 46132111 200945 0  
2014 0 5142004 17 0 13211219 9120913 4 0 80252007 1513 0 3150404 645158 27 0 0 0 27 0 0 0  
23080911 4 0 84182104 37 27 0 0 0 0 0 0 0 0 21190517 61091415 2119 0 62 0 9141620 20585134 0 5158  
27 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 905 0 21190517 61091415 2119 0 913 0 43460523 9204631 0 46172108  
20464237 27 0 0 0 0 0 0 0 0 0 0 0 0 2180500 10 27 0 0 0 0 0 0 0 0 5120905 0 21190517 61091415 2119 0  
6362 0 46080511 164637 27 0 0 0 0 0 0 0 0 0 0 0 16180913 20585164 22010911 1021204 0 3151312  
1140418 385158 27 0 0 0 0 0 0 0 0 0 0 0 61517 0 31303 0 913 0 67797776 65786882 37 27 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 16180913 20580650 0 0 44031303 455158 27 0 0 0 0 0 0 0 0 0 0 0 16180913 20585188  
1520 0 30113 0 1121914 0 5142004 17 0 80252007 1513 0 3150404 0 16180505 9240503 0 23092007 0  
465045 0 2014 0 5240502 21200563 5158 27 0 0 0 0 0 0 0 0 0 0 16180913 20585183 251604 0  
46132111 200945 0 2014 0 5142004 17 0 13211219 9120913 4 0 80252007 1513 0 3150404 645158 27 0  
0 0 0 0 0 5120905 0 21190517 61091415 2119 0 6362 0 46132111 20094637 27 0 0 0 0 0 0 0 0 0 0  
3150404 0 62 0 13211219 9120913 5610913 16212057 58 27 0 0 0 0 0 0 0 0 0 0 0 5240502 21200560  
4251400 13090360 3150404 58031503 558 27 0 0 0 0 0 0 0 5120905 0 21190517 61091415 21206418  
20011819 19230919 8584649 465937 27 0 0 0 0 0 0 0 0 0 905 0 516350 913 0 21190517 61091415  
2119 0 1517 0 46091315 15182045 0 913 0 21190517 61091415 212037 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 39  
0 72011403 1204 0 1191908 7141304 1419 0 11403 0 5240502 21200914 13 0 19200119 5130513 2018 27  
0 0 0 0 0 0 0 0 0 0 0 0 0 5240502 21200560 4251400 13090360 3150404 58211904 18610913  
16212042 92384258 27 0 0 0 0 0 0 0 0 0 0 5121904 37 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 39 0 72011403  
1204 0 5220111 21012008 1513 0 19200119 5130513 2018 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5220111  
61042513 1130902 61031503 5582118 5186108 14162119 43923841 58 27 0 0 0 0 0 0 0 0 5121904 37 27 0  
0 0 0 0 0 0 0 0 0 0 39 0 69240502 212004 0 16180503 5060913 503 0 3151312 1140418 27 0 0 0 0 0 0 0 0  
0 0 0 0 3151312 1140460 16011819 18 0 62 0 21190517 61091415 21206418 16120919 5858 27 0 0 0 0 0 0  
0 0 0 0 0 3151312 1140460 14011304 0 62 0 3151312 1140460 16011819 19439141 27 0 0 0 0 0 0 0 0 0 0  
0 0 3151312 1140460 1180718 0 62 0 3151312 1140460 16011819 19439237 41 27 0 0 0 0 0 0 0 0 0 0 0  
15212015 2119 0 62 0 5240502 21200560 21190517 61031512 13011403 58031512 13011403 61140112  
531 0 57031512 13011403 61011806 1958 27 0 0 0 0 0 0 0 0 0 0 16180913 20581520 20162119 58 27  
40200518 19 27 27 3120118 18 0 84211808 14076714 13161204 20056520 20151300 20151492 9437 27  
0 0 0 0 40505 0 61610913 9206160 58190511 65937 27 0 0 0 0 0 0 0 19051205 64200115 4 0 62 0 4444 27  
0 0 0 0 0 0 0 19051205 64080500 4611614 19092008 1513 0 62 0 90 27 27 0 0 0 0 40505 0 23180919  
5612000 16059393 58190511 631 0 22011220 55937 27 0 0 0 0 0 0 0 0 19051205 64200115 5431904  
12066407 5010460 16151908 20091513 41 0 62 0 22011220 4 27 0 0 0 0 0 0 0 39 0 71051404 18012004  
0 200804 0 16150913 19 0 2050614 1804 0 13152208 1406 0 200804 0 8050103 27 0 0 0 0 0 0 0  
19051205 64070513 5180119 5611614 9142092 945858 27 0 0 0 0 0 0 0 16180913 20580650 87180919  
200513 0 44220111 210544 0 119 0 16151908 20091513 0 44190511 6640804 1046115 15190919  
9151444 5158 27 0 0 0 0 0 0 19051205 64131521 5610804 1049393 58461808 7082045 31 0 9258 0 0  
39 0 77152204 0 8050103 0 18090707 19 0 1062004 17 0 23180919 91406 0 11403 0 7051404 18012008  
1406 0 16150913 19 27 27 0 0 0 0 40505 0 13152204 61080500 4939457 19051205 31 0 4091804 3200914  
1431 0 19200515 19639258 37 27 0 0 0 0 0 0 0 15120460 16151908 20091513 0 62 0 19051205  
64080500 4611614 19092008 1513 27 0 0 0 0 0 0 0 905 0 4091804 3200914 13 0 6362 0 46120505  
204637 27 0 0 0 0 0 0 0 0 0 19051205 64080500 4611614 19092008 1513 0 6062 0 19200515 18 27  
0 0 0 0 0 0 5120905 0 4091804 3200914 13 0 6362 0 46180906 8204637 27 0 0 0 0 0 0 0 0 0 0  
19051205 64080500 4611614 19092008 1513 0 6262 0 19200515 18 27 0 0 0 0 0 0 16180913 20580650  
77152204 3 0 6181512 0 44151203 61161518 9200914 1444 0 2014 0 44190511 6640804 1046115  
15190919 9151444 5158 27 27 0 0 0 0 40505 0 7051404 18012004 61161508 14209393 58190511 65937  
27 0 0 0 0 0 0 0 0 23 0 62 0 19051205 64080500 4611614 19092008 1513 27 0 0 0 0 0 0 0 24 0 62 0  
19051205 64200115 5640704 20581904 12066407 5010460 16151908 20091513 31 0 9158 27 0 0 0 0 0 0

0 16180913 20580650 71051404 18012004 3 0 16150913 19 0 58442444 31 0 44254558 5158 27 27 40505  
0 18050103 61060911 5939457 6091204 14011304 5937 27 0 0 0 23092007 0 15160513 58060911  
5140112 531 0 46184631 0 5140314 4091406 63462119 6609945 58 0 118 0 6091204 37 27 0 0 0 0 0 0 0 0  
18052020 1813 0 6091204 64180500 45858 27 27 40505 0 16181502 5191960 6091204 61031513  
20051419 93945802 15142004 142031 0 20030158 37 27 0 0 0 3080117 1032004 18611300 15 0 62 0 43  
27 0 0 0 0 0 0 0 46014637 0 9131 0 46024637 0 9231 0 46034637 0 9331 0 46044637 0 9431 0 46054637  
0 9531 0 46064637 0 9631 0 46074637 0 9731 0 46084637 0 9831 0 46094637 0 9931 0 46104637 0 10031  
0 46114637 0 929131 0 46124637 0 929231 0 46134637 0 929331 0 46144637 0 929431 27 0 0 0 0 0 0 0  
46154637 0 929531 0 46164637 0 929631 0 46174637 0 929731 0 46184637 0 929831 0 46194637 0  
929931 0 46204637 0 930031 0 46214637 0 939131 0 46224637 0 939231 0 46234637 0 939331 0  
46244637 0 939431 0 46254637 0 939531 0 46264637 0 939631 27 0 0 0 0 0 0 45 0 4637 0 939731 0  
46301445 37 0 939831 0 46302045 37 0 939931 0 46303045 37 0 940031 0 46304645 37 0 949131 0  
46324637 0 949231 0 46334637 0 949331 0 46344637 0 949431 0 46354637 0 949531 0 46364637 0  
949631 0 46374637 0 949731 0 46384637 0 949831 27 0 0 0 0 0 0 0 0 46394637 0 949931 0 46404637 0  
950031 0 46414637 0 959131 0 46424637 0 959231 0 46434637 0 959331 0 46444637 0 959431 0  
46454637 0 959531 0 46514637 0 959631 0 45 0 474637 0 959731 0 46484637 0 959831 0 45 0 494637 0  
959931 0 46504637 0 960031 27 0 0 0 0 0 0 45 0 524637 0 969131 0 46534637 0 969231 0 46544637 0  
969331 0 46554637 0 969431 0 46564637 0 969531 0 46574637 0 969631 0 46584637 0 969731 0  
46594637 0 969831 0 46604637 0 969931 0 46614637 0 970031 0 46624637 0 979131 0 46634637 0  
979231 27 0 0 0 0 0 0 0 46644637 0 979331 0 46654637 0 979431 0 46664637 0 979531 0 46674637 0  
979631 0 46684637 0 979731 0 46694637 0 979831 0 46704637 0 979931 0 46714637 0 980031 0  
46724637 0 989131 0 46734637 0 989231 0 46744637 0 989331 0 46754637 0 989431 27 0 0 0 0 0 0 0  
46764637 0 989531 0 46774637 0 989631 0 46784637 0 989731 0 46794637 0 989831 0 46804637 0  
989931 0 46814637 0 990031 0 46824637 0 999131 0 46834637 0 999231 0 46844637 0 999331 0  
46854637 0 999431 0 46864637 0 999531 0 46874637 0 999631 27 0 0 0 0 0 0 0 46884637 0 999731 0  
46894637 0 999831 0 46904637 0 999931 0 46914637 0 1000031 0 46924637 0 1009131 0 46934637 0  
1009231 0 46944637 0 1009331 0 46954637 0 1009431 0 46964637 0 1009531 0 46974637 0 1009631 0  
46984637 0 1009731 0 46994637 0 1009831 0 47004637 0 1009931 0 45 0 4637 0 10099 27 0 0 0 44 27 0  
0 0 0 61517 0 3080117 0 913 0 3151419 5142037 27 0 0 0 0 0 0 0 22011220 4 0 62 0 3080117 1032004  
18611300 16640704 20580307 11831 0 609258 27 0 0 0 0 0 0 0 20030163 23180919 5612000 16059393  
58220111 210558 27 27 40505 0 13010913 93945858 37 27 0 0 0 6091204 14011304 0 62 0 9141620  
20585168 14200517 0 200804 0 6091204 14011304 0 1505 0 200804 0 64202419 0 6091204 37 0 5158 27  
0 0 0 0 3151419 51419 0 62 0 18050103 61060911 5939457 6091204 14011304 58 27 0 0 0 0 200300 0 62  
0 84211808 14076714 13161204 20056520 20151300 20151492 945858 27 0 0 0 16181502 5191960  
6091204 61031513 20051419 93945802 15142004 142031 0 20030158 27 0 0 0 16121519 61031514  
18040913 1200518 93945819 3016419 1160558 27 27 40505 0 16121519 61031514 18040913 1200518  
93945819 1160558 37 27 0 0 0 0 24612200 121931 0 25612200 1218 0 62 0 12091919 58200115 5641104  
25195858 5931 0 12091919 58200115 5642200 12210518 585958 27 0 0 0 60906 0 62 0 7156469  
9072117 5580400 20016306 15648302 1202004 18582462 24612200 121931 0 25632560 22011218 31 0  
13150404 63461300 18110517 19621208 14051945 31 0 14011304 63466714 15180408 14012004  
19465958 27 0 0 0 6090763 21160400 20056111 1251520 20582008 20120562 46731419 5180102  
20092204 0 80121519 0 1505 0 71051404 18012004 3 0 67151517 4091400 20051945 31 27 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 24012408 19612008 20120562 4687 0 65240918 4631 27 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 0 25012408 19612008 20120562 4688 0 65240918 4658 27 0 0 0 6090763 19081522 5858  
27 0 0 0 0 39 0 79162008 15140111 122531 0 251520 0 30113 0 19012204 0 200804 0 16121519 0 118 0  
113 0 72847775 0 6091204 27 0 0 0 6090763 23180919 5610819 13125845 16121519 64082012 124658  
27 0 0 0 16180913 20585179 121519 0 19012204 3 0 118 0 46161214 20640819 13124663 5158 27 27  
3120118 18 0 84211808 14076714 13161204 20056520 20151300 20151437 27 0 0 0 40505 0 61610913  
9206160 58190511 65937 27 0 0 0 0 0 0 19051205 64200115 4 0 62 0 4444 27 0 0 0 0 0 0 0 19051205  
64080500 4611614 19092008 1513 0 62 0 90 27 27 0 0 0 0 40505 0 23180919 5612000 16055818 5120631

0 22011220 55937 27 0 0 0 0 0 0 0 0 19051205 64200115 5431904 12066407 5010460 16151908 20091513  
41 0 62 0 22011220 4 27 0 0 0 0 0 0 0 0 16180913 20580650 87180919 200513 0 44220111 210544 0 119 0  
16151908 20091513 0 44190511 6640804 1046115 15190919 9151444 5158 27 27 0 0 0 40505 0  
13152204 61080500 4581904 120631 0 4091804 3200914 1431 0 19200515 195937 27 0 0 0 0 0 0 0  
15120460 16151908 20091513 0 62 0 19051205 64080500 4611614 19092008 1513 27 0 0 0 0 0 0 0 905  
0 4091804 3200914 13 0 6362 0 46120505 204637 27 0 0 0 0 0 0 0 0 0 0 0 0 19051205 64080500 4611614  
19092008 1513 0 6062 0 19200515 18 27 0 0 0 0 0 0 0 0 0 0 0 0 19051205 64080500 4611614  
46180906 8204637 27 0 0 0 0 0 0 0 0 0 0 0 19051205 64080500 4611614 19092008 1513 0 6262 0  
19200515 18 27 0 0 0 0 0 0 0 0 16180913 20580650 77152204 3 0 6181512 0 44151203 61161518 9200914  
1444 0 2014 0 44190511 6640804 1046115 15190919 9151444 5158 27 27 0 0 0 40505 0 7051404  
18012004 61161508 14205818 5120658 37 27 0 0 0 0 0 0 0 0 23 0 62 0 19051205 64080500 4611614  
19092008 1513 27 0 0 0 0 0 0 0 0 24 0 62 0 19051205 64200115 5640704 20581904 12066407 5010460  
16151908 20091513 31 0 9158 27 0 0 0 0 0 0 0 0 16180913 20580650 71051404 18012004 3 0 16150913  
19 0 58442444 31 0 44254558 5158 27 0 0 0 0 0 0 0 0 18052020 1813 0 582431 0 2558 27 27 40505 0  
13010913 93935858 37 27 0 0 0 0 200300 0 62 0 84211808 14076714 13161204 20056520 20151300  
20151457 58 27 0 0 0 0 16150913 2018 0 62 0 4341 27 27 0 0 0 0 16180913 20585186 5120314 1304 0  
2014 0 200804 0 84211808 1406 0 67151315 12052004 0 67151517 4091400 2004 0 80150913 19 0  
83091320 12012014 185158 27 0 0 0 0 16180913 20585172 14192017 21032008 15141937 5158 27 0 0 0  
16180913 205850 0 0 23180919 4 0 43220111 210541 0 59 0 87180919 4 0 0 0 22011220 4 0 119 0 200804  
0 3211817 51419 0 20011604 0 16151908 20091513 645158 27 0 0 0 0 16180913 205850 0 0 13152204 0  
12050619 0 43192004 161941 0 59 0 77152204 0 200804 0 8050103 0 12050619 0 224 0 0 0 19160502  
9060904 3 0 14211301 517 0 1505 0 19200515 19645158 27 0 0 0 0 16180913 205850 0 0 13152204 0  
18090707 19 0 43192004 161941 0 59 0 77152204 0 200804 0 8050103 0 18090707 19 0 224 0 0 0  
19160502 9060904 3 0 14211301 517 0 1505 0 19200515 19645158 27 0 0 0 0 16180913 205850 0 0  
7051404 18012004 0 59 0 71051404 18012004 0 0 0 16150913 19 0 2011904 3 0 1513 0 200804 0 3211817  
51419 0 20011604 0 11403 0 8050103 0 16151908 20091513 645158 27 0 0 0 0 16180913 205850 0 0  
5240919 0 59 0 69240919 0 200804 0 16181506 18011363 5158 27 0 0 0 0 16180913 205858 27 27 0 0 0  
23080911 4 0 84182104 37 27 0 0 0 0 0 0 0 0 3151312 11403 0 62 0 9141620 20585168 14200517 0  
3151312 1140437 0 51596418 20180915 58596411 15230517 5858 27 0 0 0 0 0 0 0 0 16011819 18 0 62 0  
3151312 1140463 19161208 205858 27 0 0 0 0 0 0 0 1032008 1513 0 62 0 16011819 19439141 27 27 0 0  
0 0 0 0 0 905 0 1032008 1513 0 6362 0 51231808 200550 0 11403 0 12051457 16011819 1958 0 34 0  
9237 27 0 0 0 0 0 0 0 0 0 0 0 22011220 4 0 62 0 6121500 20581600 18201942 924258 27 0 0 0 0 0 0 0 0  
0 0 0 20030163 23180919 5612000 16055821 1122104 58 27 0 0 0 0 0 0 0 0 5120905 0 1032008 1513 0  
913 0 43511314 22055131 0 51131521 55141 0 11403 0 12051457 16011819 1958 0 6362 0 9437 27 0 0 0  
0 0 0 0 0 0 0 0 4091804 3200914 13 0 62 0 16011819 19439241 27 0 0 0 0 0 0 0 0 0 0 19200515 18 0  
62 0 9142057 16011819 19439341 58 27 0 0 0 0 0 0 0 0 0 0 20030163 13152204 61080500 4580408  
18050319 9151431 0 19200515 1958 27 0 0 0 0 0 0 0 0 5120905 0 1032008 1513 0 6362 0 51070513  
5180119 55137 27 0 0 0 0 0 0 0 0 0 0 16150913 20196400 16160513 4582002 1640704 14051800  
20056115 15091419 585958 27 0 0 0 0 0 0 0 0 5120905 0 1032008 1513 0 6362 0 51052408 205137 27 0 0  
0 0 0 0 0 0 0 0 0 16180913 20585168 24092008 14076463 645158 27 0 0 0 0 0 0 0 0 0 0 2180500 10  
27 0 0 0 0 0 0 0 0 5121904 37 27 0 0 0 0 0 0 0 0 0 0 16180913 20585184 14111414 2313 0 3151312  
1140463 0 80120500 1904 0 201824 0 1070108 14645158 27 27 0 0 0 0 16180913 20585170 5140517  
1200503 0 67151517 4091400 20051937 5131 0 16150913 201958 27 0 0 0 0 16121519 61031514  
18040913 1200518 93935815 15091419 1958 27 27 40505 0 16121519 61031514 18040913 1200518  
93935815 15091419 195937 27 0 0 0 0 24612200 121931 0 25612200 1218 0 62 0 26091657 57161508  
14201958 0 905 0 16150913 2018 0 5121904 0 58434231 0 434258 27 0 0 0 0 60906 0 62 0 7156469  
9072117 5580400 20016306 15648302 1202004 18582462 24612200 121931 0 25632560 22011218 31 0  
13150404 63461300 18110517 19621208 14051945 31 0 14011304 63466714 15180408 14012004  
19465958 27 0 0 0 0 6090763 21160400 20056111 1251520 20582008 20120562 46731419 5180102  
20092204 0 80121519 0 1505 0 71051404 18012004 3 0 67151517 4091400 20051945 31 27 0 0 0 0 0 0 0 0



913 0 18011406 5589131 0 12051457 16180912 5195931 0 945937 27 0 0 0 0 0 0 0 0 3082113 10 0 62 0  
16180912 5194308 38096293 41 27 0 0 0 0 0 0 0 905 0 12051457 3082113 1158 0 33 0 9437 27 0 0 0 0  
0 0 0 0 0 0 3082113 10 0 6262 0 439241 0 56 0 5893 0 59 0 12051457 3082113 115958 27 0 0 0 0 0 0  
3151517 4196400 16160513 4580307 21141158 27 0 0 0 0 18052020 1813 0 3151517 418 27 27 39 0  
65140912 1200914 13 0 19052020 15 27 40505 0 1140912 1200592 92580958 37 27 0 0 0 0 1246402  
12050117 5858 27 0 0 0 0 1246418 3012019 5185856 26091657 57031514 18041942 38096291 425931 0  
3634617 4631 0 13011810 5186345 154658 27 0 0 0 0 1246418 5206123 12091357 9131 0 13012457  
3151517 41931 0 11052562 12011301 400 0 2437 0 24439141 59439141 0 61 0 929158 27 0 0 0 0 1246418  
5206124 12091357 9131 0 13012457 3151517 41931 0 11052562 12011301 400 0 2437 0 24439241  
59439241 0 61 0 929158 27 0 0 0 0 1246418 5206125 12091357 9131 0 13012457 3151517 41931 0  
11052562 12011301 400 0 2437 0 24439341 59439341 0 61 0 929158 27 27 40404039 40404039 40404039  
40404039 404039 27 27 27 40505 0 9140404 24612014 61180701 93005802 15121517 61091403 5245937  
27 0 0 0 0 51515166 15142204 1819 0 0 0 3151214 17 0 9140404 23 0 2014 0 113 0 827165 0 20211611  
5645150 50 27 0 0 0 0 180503 0 62 0 58031511 15186108 14040523 0 3534 0 929758 0 55 0 939695 0 0 39  
0 83080905 19 0 18090707 19 0 224 0 9296 0 2092018 0 11403 0 13011910 0 23092007 0 939695 0 2014 0  
70519 0 200804 0 180503 0 3151315 15140513 19 27 0 0 0 0 7180504 13 0 62 0 58031511 15186108  
14040523 0 3534 0 9958 0 55 0 939695 0 0 39 0 83080905 19 0 18090707 19 0 224 0 98 0 2092018 0  
11403 0 13011910 0 23092007 0 939695 0 2014 0 70519 0 200804 0 7180504 13 0 3151315 15140513 19  
27 0 0 0 0 2122104 0 62 0 3151214 18610913 40523 0 55 0 939695 0 0 39 0 77011910 0 23092007 0  
939695 0 2014 0 70519 0 200804 0 2122104 0 3151315 15140513 19 27 0 0 0 0 18052020 1813 0  
58180503 31 0 7180504 1431 0 2122104 58 27 27 40505 0 3011202 21120119 5610314 14060906  
21180119 9151460 9140404 24930057 19172100 18056102 15121517 61091403 52431 0 19251301  
15126108 14040523 31 0 6151419 61031511 15186108 14040523 5937 27 0 0 0 0 78857760 83897765  
797682 0 62 0 10097 27 0 0 0 0 78857760 70797883 61677975 798282 0 62 0 93969756 5793 27 0 0 0 0  
9140404 23 0 62 0 58191720 1180560 3151214 18610913 40523 0 56 0 78857760 83897765 797682 0 56 0  
78857760 70797883 61677975 79828358 0 61 0 58192512 2151260 9140404 23 0 56 0 78857760  
70797883 61677975 79828358 0 61 0 6151419 61031511 15186108 14040523 27 0 0 0 0 18052020 1813 0  
9140404 23 27 27 40505 0 19012204 61031513 6090720 18012008 15146119 15610608 12059299  
58060911 5611400 130531 0 7180903 61190925 531 0 19172100 18056118 9260531 0 6151419 61190925  
531 0 3151405 9072117 1200914 14195937 27 0 0 0 0 23092007 0 15160513 58060911 5611400 130531 0  
46234658 0 118 0 6091204 37 27 0 0 0 0 0 0 6091204 64231808 20055805 51711808 3 0 83092604 37  
0 44071808 4611908 26054523 44071808 4611908 26054529 145158 27 0 0 0 0 0 0 0 6091204 64231808  
20055805 51832101 60191720 11804 0 80092404 11 0 76051406 200837 0 44191720 1180560 19092604  
45301450 58 27 0 0 0 0 0 0 0 6091204 64231808 20055805 51701513 19 0 83092604 37 0 44061513  
20611908 26054529 145158 27 0 0 0 0 0 0 6091204 64231808 20055850 67151405 9072117 1200914  
14193829 145158 27 0 0 0 0 0 0 0 0 61517 0 3151405 906 0 913 0 3151405 9072117  
1200914 141937 27 0 0 0 0 0 0 0 0 0 0 19172100 18056102 15121517 61091403 52431 0 19251301  
15126108 14040523 31 0 6151419 61031511 15186108 14040523 0 62 0 3151405 906 27 0 0 0 0 0 0 0 0  
0 0 9140404 23 0 62 0 3011202 21120119 5610314 14060906 21180119 9151460 9140404 24930057  
19172100 18056102 15121517 61091403 52431 0 19251301 15126108 14040523 31 0 6151419 61031511  
15186108 14040523 58 27 0 0 0 0 0 0 0 0 0 0 39 0 67151421 51819 0 9140408 30518 0 2014 0 827165  
27 0 0 0 0 0 0 0 0 0 0 0 0 19172100 18056102 15121517 61180701 0 62 0 9140404 24612014 61180701  
93005818 17210117 5610314 12151860 9140404 2458 27 0 0 0 0 0 0 0 0 0 0 6151419 61031511  
15186117 701 0 62 0 9140404 24612014 61180701 93005805 15142060 3151214 18610913 4052458 27 0  
0 0 0 0 0 0 0 0 0 0 6091204 64231808 20055805 51731403 52437 0 44091403 5244531 0 83172100 1804  
0 67151214 17 0 73140404 2437 0 44191720 1180560 3151214 18610913 4052444 0 58827165 37 0  
44191720 1180560 3151214 18611806 2455931 0 83251301 151237 0 44192512 2151260 9140404  
244531 0 70151419 0 67151214 17 0 73140404 2437 0 44061513 20610314 12151860 9140404 2444 0  
58827165 37 0 44061513 20610314 12151860 18070244 59301450 58 27 27 40505 0 9196115  
18091304 92995813 21135937 27 0 0 0 51515166 8050310 0 905 0 0 0 14211301 517 0 918 0 16180912

563 0 65191920 130518 0 142112 0 918 0 0 0 16151908 20092204 0 9142004 7051863 515150 27 0 0 0 0  
905 0 142112 0 33 0 9337 27 0 0 0 0 0 0 0 18052020 1813 0 70011218 4 27 0 0 0 61517 0 8 0 913 0  
18011406 5589331 0 9142057 13012007 64191717 20581420 135958 0 61 0 925937 27 0 0 0 0 0 0 0 0 905  
0 142112 0 53 0 8 0 6362 0 9137 27 0 0 0 0 0 0 0 0 0 0 18052020 1813 0 70011218 4 27 0 0 0  
18052020 1813 0 84182104 27 27 40505 0 13010913 92995858 37 27 0 0 0 39 0 73141620 19 0 0 0  
11403 0 1 27 0 0 0 0 0 0 62 0 9142057 9141620 20585168 14200517 0 0 0 5800 0 34 0 25937 0 515958 27 0  
0 0 0 1 0 62 0 9142057 9141620 20585168 14200517 0 1 0 5801 0 33 0 15937 0 515958 27 27 0 0 0 0 39 0  
86011208 4012004 0 3151403 9200914 1418 0 61517 0 0 0 11403 0 1 27 0 0 0 0 905 0 141519 0 5800 0 34  
0 1 0 11403 0 90 0 33 0 5800 0 59 0 258 0 3462 0 92935937 27 0 0 0 0 0 0 0 16180913 20585183 804 0  
3151403 9200914 1418 0 90 0 33 0 5800 0 59 0 258 0 3462 0 9292 0 11403 0 0 0 34 0 1 0 11804 0 141519 0  
13052063 5158 27 0 0 0 0 0 0 0 18052020 1813 27 0 0 0 27 0 0 0 0 39 0 67011202 21120119 4 0 1631 0  
200804 0 192112 0 6181512 0 8 0 62 0 0 0 2014 0 8 0 62 0 1 0 1505 0 5808 0 61 0 9258 27 0 0 0 0 15 0 62 0  
19211357 8 0 61 0 91 0 61517 0 8 0 913 0 18011406 5580231 0 0 0 61 0 925958 27 0 0 0 0 27 0 0 0 0 39 0  
67011202 21120119 4 0 25 27 0 0 0 0 25 0 62 0 13012007 64191717 205815 0 32 0 5800 0 32 0 935958 27  
0 0 0 0 27 0 0 0 0 39 0 67011202 21120119 4 0 5161908 121513 0 58040502 9130111 0 16011819 0 1505 0  
2658 27 0 0 0 0 5161908 12151491 0 62 0 25 0 59 0 9142057 2658 27 0 0 0 0 5161908 12151492 0 62 0 91  
0 59 0 5161908 12151491 27 0 0 0 0 5161908 12151493 0 62 0 91 27 0 0 0 0 27 0 0 0 0 39 0 80180515  
11804 0 200804 0 16151918 9021204 0 25 0 22011220 518 0 2014 0 3080502 10 0 61517 0 16180912  
1120919 24 27 0 0 0 0 26612200 12210518 0 62 0 43 27 0 0 0 0 0 0 0 5125 0 61 0 5161908 12151492  
5137 0 25 0 61 0 5161908 12151492 31 27 0 0 0 0 0 0 0 5125 0 59 0 5161908 12151491 5137 0 25 0 59 0  
5161908 12151491 31 27 0 0 0 0 0 0 0 5125 0 59 0 5161908 12151492 5137 0 25 0 59 0 5161908  
12151492 31 27 0 0 0 0 0 0 0 5125 0 61 0 5161908 12151491 5137 0 25 0 61 0 5161908 12151491 31 27 0  
0 0 0 0 0 0 5125 0 61 0 5161908 12151493 5137 0 25 0 61 0 5161908 12151493 31 27 0 0 0 0 0 0 0 5125  
0 59 0 5161908 12151493 5137 0 25 0 59 0 5161908 12151493 27 0 0 0 0 44 27 0 0 0 0 27 0 0 0 0 39 0  
710519 0 6091204 0 14011304 0 6181512 0 21190517 0 11403 0 23180919 4 0 200804 0 18051920 122018  
27 0 0 0 0 6091204 61140112 4 0 62 0 9141620 20585168 14200517 0 200804 0 14011304 0 1505 0  
200804 0 6091204 0 2014 0 19012204 0 18051920 12201937 0 5158 27 0 0 0 0 23092007 0 15160513  
58060911 5611400 130531 0 51235158 0 118 0 6091204 37 27 0 0 0 0 0 0 0 6091204 64231808  
20055805 51731415 2119 0 22011220 51937 0 0 0 62 0 44014531 0 1 0 62 0 44024529 145158 27 0 0 0 0 0  
0 0 0 6091204 64231808 20055805 51670111 3211200 200503 0 1637 0 44164529 145158 27 0 0 0 0 0 0 0  
0 6091204 64231808 20055805 51670111 3211200 200503 0 2637 0 44264529 145158 27 0 0 0 0 0 0 0  
6091204 64231808 20055805 51670111 3211200 200503 0 5161908 12151491 37 0 44051618 9121513  
92453013 5158 27 0 0 0 0 0 0 0 6091204 64231808 20055805 51670111 3211200 200503 0 5161908  
12151492 37 0 44051618 9121513 93453013 5158 27 0 0 0 0 0 0 0 6091204 64231808 20055805  
51051618 9121513 9437 0 44051618 9121513 94453013 5158 27 0 0 0 0 0 0 0 61517 0 12010204 1231 0  
22011220 4 0 913 0 26612200 12210518 64092004 13195858 37 27 0 0 0 0 0 0 0 0 0 0 905 0 9142057  
22011220 558 0 6362 0 22011220 537 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16180912 5611919 1202118 0 62 0  
9196115 18091304 92995808 14205821 1122104 5958 27 0 0 0 0 0 0 0 0 0 0 5121904 37 27 0 0 0 0 0  
0 0 0 0 0 0 0 0 16180912 5611919 1202118 0 62 0 51783364 0 58141519 0 113 0 9142004 7051858 50  
27 0 0 0 0 0 0 0 0 0 6091204 64231808 20055805 517318 0 44120101 51244 0 62 0 44220111 210544  
0 16180912 535 0 44161808 13056118 20012020 19453013 5158 27 27 40505 0 1161604 14046118  
20180913 7612014 61060911 5935803 9180502 20151824 31 0 6091204 14011304 31 0 21190517  
61192017 9140758 37 27 0 0 0 0 51515164 16160513 418 0 200804 0 21190517 0 19201808 1406 0 2014 0  
0 0 6091204 0 905 0 919 0 4150518 0 141519 0 1121804 10424 0 5240918 19 0 913 0 200804 0 6091204  
64515150 27 0 0 0 39 0 67180500 2004 0 6211211 0 6091204 0 16012007 27 0 0 0 0 6091204 61160119  
7 0 62 0 15196415 1200863 10150913 58040917 5032014 182531 0 6091204 14011304 58 27 0 0 0 0 27 0  
0 0 39 0 67080502 10 0 905 0 6091204 0 5240918 2018 0 11403 0 18050103 0 92018 0 3151419 5142018  
0 905 0 919 0 4150518 27 0 0 0 9065814 19641600 20086404 24091919 19580608 12056115 1200858 0  
11403 0 1205457 21190517 61192017 9140758 0 34 0 955937 27 0 0 0 0 0 0 0 23092007 0 15160513  
58060911 5611600 200831 0 46184631 0 5140314 4091406 63462119 6609945 58 0 118 0 6091204 37 27



14211301 51818 0 62 0 43091419 582458 0 61517 0 23 0 913 0 12091404 64192017 9165858 64191611  
9205858 41 27 0 0 0 0 0 0 2091400 18256118 20180913 718 0 62 0 43020118 5612460 20156101 9140117  
25581420 1331 0 2011904 58 0 61517 0 142112 0 913 0 14211301 5181941 27 0 0 0 0 0 6611520  
20642317 9200557 50 0 51641014 9145801 9140117 25611919 18091406 1958 0 61 0 51301450 58 27 27  
40505 0 7051404 18012004 61011403 61190121 5611460 2092060 3151301 9140119 9151418 58060911  
5611600 200831 0 145937 27 0 0 0 0 23092007 0 15160513 58060911 5611600 200831 0 51235158 0 118  
0 6091204 37 27 0 0 0 0 0 0 0 61517 0 8 0 913 0 18011406 5589356 57145937 0 0 39 0 935513 0 3151301  
9140119 9151418 27 0 0 0 0 0 0 0 0 0 0 3151301 9140119 91513 0 62 0 2091457 9594392 38426425  
6091211 581458 27 0 0 0 0 0 0 0 0 0 0 6091204 64231808 20055802 15130208 14012008 1513 0 61 0  
51301450 58 27 27 40505 0 4050314 13160911 5610400 20015858 37 27 0 0 0 0 51515167 5031512  
16091204 18 0 4012000 0 21190913 6 0 0 0 4251400 13090300 121224 0 12150103 503 0 4050314  
13160911 4 0 6211402 20091513 64515150 27 0 0 0 0 14011304 0 62 0 9141620 20585168 14200517 0  
200804 0 14011304 0 1505 0 200804 0 4012000 0 19201820 3202117 4 0 2014 0 4050314 13160911 537 0  
5158 27 0 0 0 0 6091204 14011304 0 62 0 6514413 1130544 61121506 9036415 2550 27 0 0 0 905 0  
141519 0 15196415 1200863 5240918 20195805 9120513 1130558 37 27 0 0 0 0 0 0 0 16180913  
20580650 76150708 2 0 19031808 1619 0 61517 0 44140112 544 0 141519 0 6152113 463 0 80120500  
1904 0 5141920 1804 0 200804 0 4012000 0 19201820 3202117 4 0 918 0 4050608 140503 0 11403 0  
3151315 9120503 0 6091818 20645158 27 0 0 0 0 0 0 0 18052020 1813 27 0 0 0 27 0 0 0 3151315  
9120503 61091415 2119 0 62 0 9141620 20585168 14200517 0 3151315 9120503 0 4012000 0 19201808  
140737 0 5158 27 0 0 0 0 20182537 27 0 0 0 0 0 0 0 4050314 13160911 5046103 12000 0 62 0 5240502  
21200560 4251400 13090360 19031808 16205813 1130531 0 6510404 3151315 9120560 44140112  
5455131 0 3151315 9120503 61091415 212058 27 0 0 0 0 0 0 16180913 20585167 5031512 16091204  
3 0 68012000 385131 0 10191513 64042112 16195803 5031512 16091204 4610400 200131 0 9140404  
14206394 5958 27 0 0 0 0 5240304 1619 0 69240304 16200914 13 0 118 0 537 27 0 0 0 0 0 0 16180913  
20585164 13 0 5181814 17 0 15030320 18180503 385131 0 558 27 27 40505 0 5240502 21200560  
4251400 13090360 19031808 16205813 1130531 0 6211402 20091513 61140112 531 0 4012000 5937 27  
0 0 0 51515167 25140112 9030111 1224 0 9131614 1819 0 11403 0 5240502 212004 0 0 0 6211402  
20091513 0 6181512 0 0 0 7051404 18012004 3 0 19031808 16206450 5150 27 0 0 0 6091204 14011304  
0 62 0 6514413 1130544 61121506 9036415 2550 27 0 0 0 19160502 0 62 0 9131614 18201208 2642119  
9126418 16050360 6181512 61060911 5611214 3012008 15145813 1130531 0 6091204 14011304 58 27 0  
0 0 0 13150420 1204 0 62 0 9131614 18201208 2642119 9126412 15042111 5610617 15136118 16050357  
19160502 58 27 0 0 0 19160502 64121500 4051863 5240502 61131503 21120557 13150420 120558 27 0  
0 0 0 6211402 0 62 0 7052000 20201857 13150420 120531 0 6211402 20091513 61140112 558 27 0 0 0  
18052020 1813 0 6211402 58040119 158 27 0 0 0 27 40505 0 16151620 12012004 61040119 1581400  
13055937 27 0 0 0 51515179 18151315 19 0 200804 0 21190517 0 2014 0 5142004 17 0 4012000 0  
61517 0 200804 0 4050608 140503 0 19201820 3202117 4 0 11403 0 19012204 0 2014 0 0 0 20052419 0  
6091204 64515150 27 0 0 0 6091204 14011304 0 62 0 6514413 1130544 64101914 1450 27 0 0 0 0 905 0  
141519 0 15196415 1200863 5240918 20195805 9120513 1130558 37 27 0 0 0 0 0 0 16180913  
20580650 68012000 0 19201820 3202117 4 0 4050608 14092008 1513 0 61517 0 44140112 544 0 141519  
0 6152113 463 0 80120500 1904 0 4050608 1404 0 200804 0 4012000 0 19201820 3202117 4 0 6091818  
20645158 27 0 0 0 0 0 0 0 18052020 1813 27 27 0 0 0 0 20182537 27 0 0 0 0 0 0 0 23092007 0  
15160513 58060911 5140112 531 0 46184658 0 118 0 6091204 37 27 0 0 0 0 0 0 0 0 0 0 0 4012000  
61192017 21032020 1804 0 62 0 10191513 64121500 4580608 120558 27 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0  
4012000 0 62 0 4444 27 0 0 0 0 0 0 0 61517 0 6090511 3 0 913 0 4012000 61192017 21032020 18054345  
6090511 4194641 37 27 0 0 0 0 0 0 0 0 0 0 0 4012000 43060904 120441 0 62 0 9141620 20580650  
69142004 17 0 22011220 4 0 61517 0 44060904 12044537 0 5158 27 27 0 0 0 0 0 0 0 0 4012000 61060911  
5140112 4 0 62 0 6514413 1130544 61040119 1642023 2050 27 0 0 0 0 0 0 0 23092007 0 15160513  
58040119 1610608 12051400 130531 0 46014658 0 118 0 6091204 37 27 0 0 0 0 0 0 0 0 0 0 0 6091204  
64231808 20055845 0 47 0 46641014 9145818 20185803 1200142 6090511 44258 0 61517 0 6090511 3 0  
913 0 4012000 61192017 21032020 18054345 6090511 4194641 58 0 61 0 46301445 58 27 0 0 0 0 0 0 0

16180913 20580650 68012000 0 19012204 3 0 2014 0 44040119 1610608 12051400 13054563 5158 27 0 0  
0 0 27 0 0 0 0 5240304 1619 0 70091204 78152069 15211403 69181814 1837 27 0 0 0 0 0 0 0 0 16180913  
20585168 18181517 0 12150103 91406 0 4012000 0 19201820 3202117 4 0 6091204 63 0 80120500 1904  
0 3080502 10 0 200804 0 6091204 0 19251919 5136450 58 27 27 40505 0 3152113 20010211 5195858  
37 27 0 0 0 0 23080911 4 0 84182104 37 27 0 0 0 0 0 0 0 16180913 20585129 14770108 13 0 77051420  
385158 27 0 0 0 0 0 0 0 0 16180913 20585191 63 0 68050608 1404 0 68012000 0 83201820 3202117  
55158 27 0 0 0 0 0 0 0 0 16180913 20585192 63 0 80151620 12012004 0 68012000 5158 27 0 0 0 0 0 0 0  
16180913 20585193 63 0 67151315 91204 0 68012000 5158 27 0 0 0 0 0 0 0 16180913 20585194 63 0  
68050314 13160911 4 0 68012000 5158 27 0 0 0 0 0 0 0 16180913 20585195 63 0 69240919 5158 27 0 0  
0 0 0 0 0 3081508 304 0 62 0 9141620 20585168 14200517 0 3081508 30537 0 5158 27 27 0 0 0 0 0 0 0  
905 0 3081508 304 0 6362 0 51925137 27 0 0 0 0 0 0 0 0 0 0 4050608 14056103 1200160 19201820  
3202117 55858 27 0 0 0 0 0 0 0 0 5120905 0 3081508 304 0 6362 0 51935137 27 0 0 0 0 0 0 0 0 0 0  
14011304 0 62 0 9141620 20585168 14200517 0 200804 0 14011304 0 1505 0 200804 0 4012000 0  
19201820 3202117 4 0 2014 0 16151620 12012004 37 0 5158 27 0 0 0 0 0 0 0 0 0 0 16151620  
12012004 61040119 1581400 130558 27 0 0 0 0 0 0 0 0 5120905 0 3081508 304 0 6362 0 51945137 27 0 0  
0 0 0 0 0 0 0 0 0 0 14011304 0 62 0 9141620 20585168 14200517 0 200804 0 14011304 0 1505 0 200804 0  
4012000 0 19201820 3202117 4 0 2014 0 3151315 9120537 0 5158 27 0 0 0 0 0 0 0 0 0 0 0 3151315  
9120560 4012000 58140112 558 27 0 0 0 0 0 0 5120905 0 3081508 304 0 6362 0 51955137 27 0 0 0 0  
0 0 0 0 0 0 4050314 13160911 5610400 20015858 27 0 0 0 0 0 0 0 5120905 0 3081508 304 0 6362 0  
51965137 27 0 0 0 0 0 0 0 0 0 0 2180500 10 27 0 0 0 0 0 0 0 5121904 37 27 0 0 0 0 0 0 0 0 0 0  
16180913 20585172 14220111 903 0 3081508 30563 0 80120500 1904 0 19051204 319 0 1070108  
14645158 27 27 39 0 65191920 13091406 0 200804 0 5240502 21200560 4251400 13090360 19031808  
1619 0 6211402 20091513 0 918 0 1121804 10424 0 4050608 140503 0 11403 0 9131614 18200503 0 118  
0 4091902 21191904 463 27 27 40505 0 4050608 14056103 1200160 19201820 3202117 5585937 27 0 0  
0 20051315 12012004 61030814 90304 0 62 0 9141620 20585182 5120502 19 0 20051315 12012004 0  
58020118 90331 0 4052000 9120503 5937 0 5158 27 0 0 0 905 0 20051315 12012004 61030814 90304 0  
141519 0 913 0 20051315 12012004 1937 27 0 0 0 0 0 0 16180913 20585172 14220111 903 0  
20051315 12012004 63 0 82052020 18140913 6 0 2014 0 13010913 0 13051420 645158 27 0 0 0 0 0 0  
18052020 1813 27 27 0 0 0 14011304 0 62 0 9141620 20585168 14200517 0 14011304 0 61517 0 200804  
0 4012000 0 19201820 3202117 537 0 5158 27 0 0 0 4012000 61192017 21032020 1804 0 62 0 44461400  
13054637 0 14011304 31 0 46060904 12041945 37 0 20051315 12012004 19432004 13161200 20056102  
8150902 54244 27 0 0 0 6091204 14011304 0 62 0 6514413 1130544 64101914 1450 27 0 0 0  
23092007 0 15160513 58060911 5140112 531 0 46234658 0 118 0 6091204 37 27 0 0 0 0 0 0 0 10191513  
64042112 16580400 20016118 20182102 20211804 31 0 6091204 58 27 0 0 0 19031808 16206102  
15142004 1419 0 62 0 7051404 18012004 61190317 9162018 58140112 531 0 20051315 12012004  
19432004 13161200 20056102 8150902 54258 27 0 0 0 19012204 61190317 9162057 14011304 31 0  
19031808 16206102 15142004 142058 27 0 0 0 0 16180913 20580650 68012000 0 19201820 3202117 4 0  
44140112 544 0 4050608 140503 0 23092007 0 20051315 12012004 0 44200512 16120119 5610307  
15090304 44 0 11403 0 19012204 3 0 118 0 44060911 5140112 5456450 58 27 27 40505 0 7051404  
18012004 61190317 9162018 58140112 531 0 6090511 4195937 27 0 0 0 3151315 9120560 6211402 0  
62 0 6510404 5 0 3151315 9120560 44140112 5455803 1200158 38301450 27 0 0 0 3151315 9120560  
6211402 0 6262 0 50 0 0 0 18052020 1813 0 46484663 10150913 58192017 58040119 1430608 5120441  
58 0 61517 0 6090511 3 0 913 0 4012000 59301450 27 0 0 0 27 0 0 0 0 4050314 13160911 5610620 1402  
0 62 0 6510404 5 0 4050314 13160911 5614413 1130544 58040119 1593829 1450 27 0 0 0 4050314  
13160911 5610620 1402 0 6262 0 50 0 0 0 6090511 418 0 62 0 50 0 61 0 19201857 6090511 41958 0 61  
0 51301450 27 0 0 0 0 4050314 13160911 5610620 1402 0 6262 0 50 0 0 0 18052020 1813 0 4090319  
58260915 58060904 12041931 0 4012000 64191611 9205845 48465958 59301450 27 0 0 0 27 0 0 0  
18052020 1813 0 3151315 9120560 6211402 0 61 0 51301450 0 61 0 4050314 13160911 5610620 1402 27  
27 40505 0 19012204 61190317 9162057 14011304 31 0 3151419 5142058 37 27 0 0 0 0 6091204  
14011304 0 62 0 6514413 1130544 61121506 9036415 2550 27 0 0 0 23092007 0 15160513 58060911

5140112 531 0 46234658 0 118 0 6091204 37 27 0 0 0 0 0 0 0 6091204 64231808 20055802 15142004  
142058 27 0 0 0 0 16180913 20580650 67151315 9120532 68050314 13160911 4 0 12150708 2 0  
19012204 3 0 2014 0 44060911 5140112 5456450 58 27 0 0 0 0 27 40505 0 3151315 9120560 4012000  
58140112 55937 0 0 0 0 27 0 0 0 51515166 15131608 120518 0 4012000 0 21190913 6 0 0 0 4251400  
13090300 121224 0 12150103 503 0 3151315 91204 0 6211402 20091513 64515150 27 0 0 0 0 6091204  
14011304 0 62 0 6514413 1130544 61121506 9036415 2550 27 0 0 0 0 905 0 141519 0 15196415 1200863  
5240918 20195805 9120513 1130558 37 27 0 0 0 0 0 0 0 16180913 20580650 76150708 2 0 19031808  
1619 0 61517 0 44140112 544 0 141519 0 6152113 463 0 80120500 1904 0 5141920 1804 0 200804 0  
4012000 0 19201820 3202117 4 0 918 0 4050608 140503 0 11403 0 3151315 9120503 0 6091818  
20645158 27 0 0 0 0 0 0 0 18052020 1813 27 27 0 0 0 0 16180913 20585168 14200517 0 4012000 0 2014  
0 3151315 91204 0 58061517 130119 0 118 0 74837977 31 0 5640763 31 0 44460903 46384691 93944631  
0 46140112 5463845 74150813 46455937 5158 27 0 0 0 0 16180913 20585177 15200537 0 69141920 1804  
0 25152117 0 74837977 0 918 0 3151817 50320111 24 0 6151812 1202004 3 0 61517 0 25152117 0  
3151312 11403 0 12091404 0 5142208 18151412 5142063 5158 27 0 0 0 0 4012000 61091415 2119 0 62 0  
9141620 205858 27 0 0 0 0 20182537 27 0 0 0 0 0 0 0 39 0 83201808 15 0 152119 0 21142300 14200503  
0 3080117 1032004 1818 0 905 0 14050304 19190117 24 0 58191604 3090608 2 0 2014 0 25152117 0  
5142208 18151412 5142058 27 0 0 0 0 0 0 0 4012000 61091415 2119 0 62 0 4012000 61091415  
21206418 20180915 5858 27 0 0 0 0 0 0 905 0 4012000 61091415 21206418 20011819 19230919  
8585145 5158 0 11403 0 4012000 61091415 21206404 14041922 9200857 51465158 37 27 0 0 0 0 0 0 0  
0 0 0 0 4012000 61091415 2119 0 62 0 4012000 61091415 21204391 38609241 0 0 39 0 82051314 2204 0  
19211817 15211403 91406 0 19091406 1204 0 17211519 518 0 61517 0 85140923 33760913 2123 0  
19080511 1218 27 0 0 0 0 0 0 0 4012000 0 62 0 10191513 64121500 4195803 1200160 9141620 2058 0 0  
39 0 83010604 1224 0 16011818 4 0 200804 0 74837977 0 9141620 19 27 0 0 0 0 0 0 0 3151315 9120503  
61040119 0 62 0 5240502 21200560 4251400 13090360 19031808 16205813 1130531 0 6510314  
13160911 5614413 1130544 5131 0 4012000 58 27 0 0 0 0 0 0 16180913 20585166 15131608 120503 0  
68012000 385131 0 3151315 9120503 61040119 158 27 0 0 0 0 5240304 1619 0 10191513 64748378  
78680502 15040568 18181517 37 27 0 0 0 0 0 0 16180913 20585172 14220111 903 0 74837977 0  
9141620 2063 0 80120500 1904 0 3080502 10 0 200804 0 6151812 119 0 11403 0 201824 0 1070108  
14645158 27 0 0 0 0 5240304 1619 0 69240304 16200914 13 0 118 0 537 27 0 0 0 0 0 0 0 16180913  
20585164 13 0 5181814 17 0 15030320 18180503 385131 0 558 27 27 39 0 68091804 3201224 0 9142214  
11091406 0 200804 0 13010913 0 13051420 0 6211402 20091513 0 23080513 0 200804 0 19031808 1619  
0 18211418 27 40031520 14200101 12051957 58 27 40404039 40404039 40404039 40404039 40404039  
39 27 27 40505 0 13010913 91585937 27 0 0 0 0 16180913 20585129 14301486 5120314 130563 0  
67081514 1904 0 25152117 0 15160517 1200914 1437 0 5158 27 0 0 0 39 0 85190517 0 15162008 1513  
0 2014 0 7051404 18012004 0 13011615 9140718 0 1517 0 141519 27 0 0 0 0 7051404 18012004  
61130115 16091406 18 0 62 0 9141620 20585167 14 0 251520 0 23011419 0 2014 0 7051404 18012004 0  
13011615 9140718 35 0 91 0 43061517 0 25051941 31 0 90 0 43061517 0 14154237 0 5158 27 0 0 0 0 905  
0 7051404 18012004 61130115 16091406 18 0 6362 0 51925137 27 0 0 0 0 0 0 76120113 7210106  
5611314 4051257 58 27 27 0 0 0 0 39 0 68050608 1404 0 9141620 19 0 11403 0 15212015 2119 0 4091804  
3201517 90518 27 0 0 0 0 91491 0 62 0 51121200 14072100 7057714 4330913 16212018 50 27 0 0 0 0  
15212091 0 62 0 51121200 14072100 7057714 4331520 20060911 550 27 0 0 0 0 15212092 0 62 0  
51121200 14072100 7057714 4331520 200250 27 27 27 0 0 0 0 27 0 0 0 0 16181502 518 0 62 0 9141620  
20585167 14 0 251520 0 23011419 0 2014 0 16181502 51918 0 6091204 1935 0 91 0 43061517 0  
25051941 31 0 90 0 43061517 0 14154237 0 5158 27 0 0 0 0 9065815 18150304 18 0 6362 0 46924658 37  
27 0 0 0 0 0 0 0 39 0 82050103 0 13011615 9140718 0 15141224 0 905 0 20080524 0 11804 0 14050304  
19190117 24 27 0 0 0 0 0 0 0 13011615 9140718 0 62 0 18050103 61130115 16091406 195858 27 0 0 0  
0 0 0 39 0 710519 0 21190517 0 9141620 19 0 61517 0 200804 0 18011406 4 0 1505 0 6091204 18 0  
2014 0 16181502 51918 27 0 0 0 0 0 0 0 19200117 19 0 62 0 9142057 9141620 20585168 14200517 0  
200804 0 19200117 19 0 1505 0 200804 0 18011406 4 0 1505 0 6091204 18 0 2014 0 18050103 0  
58056406 6431 0 95 0 61517 0 96642023 205937 0 515958 27 0 0 0 0 0 0 0 51403 0 62 0 9142057

9141620 20585168 14200517 0 200804 0 51403 0 1505 0 200804 0 18011406 4 0 1505 0 6091204 18 0  
2014 0 18050103 0 58056406 6431 0 9295 0 61517 0 92966419 24205937 0 515958 27 27 0 0 0 0 0 0 0  
39 0 80181502 51918 0 6091204 18 0 23092007 913 0 200804 0 19160502 9060904 3 0 18011406 4 27 0  
0 0 0 0 0 16181502 5191960 6091204 19580913 9231 0 15212091 31 0 19200117 2031 0 5140431 0  
13011615 9140718 31 0 15212092 58 27 27 0 0 0 39 0 79162008 15140111 122531 0 7051404 18012004  
0 113 0 9130106 4 0 6181512 0 20052419 0 9141620 19 27 0 0 0 17210518 20091513 4 0 62 0 9141620  
20585170 5140517 12004 0 20052419 0 9141620 19 0 2014 0 9130106 4 0 6091204 35 0 91 0 43061517 0  
25051941 31 0 90 0 43061517 0 14154237 0 5158 27 0 0 0 905 0 17210518 20091513 4 0 6362 0  
51925137 27 0 0 0 0 0 0 0 0 4012000 0 62 0 15160513 61060911 5915858 27 0 0 0 0 0 0 0 905 0 4012000  
37 27 0 0 0 0 0 0 0 0 0 0 0 16181502 5191960 9130106 5610704 14051800 20091513 91580400 200158  
27 27 40505 0 16181502 5191960 9130106 5610704 14051800 20091513 91580400 20015937 27 0 0 0 0  
51515170 5140517 1200518 0 113 0 9130106 4 0 6181512 0 16181521 9040503 0 4012000 64515150 27 0  
0 0 0 905 0 141519 0 4012000 37 27 0 0 0 0 0 0 0 16180913 20585177 14 0 4012000 0 2014 0 16181502  
51918 0 61517 0 9130106 4 0 7051404 18012008 15146450 58 27 0 0 0 0 0 0 0 18052020 1813 27 0 0 0 0  
16180913 20585167 12000 0 18050304 9220503 0 61517 0 9130106 4 0 7051404 18012008 15143850 31 0  
4012000 58 27 0 0 0 3151214 18610400 2000 0 62 0 4012000 61201560 3151214 18915803 1200158 27  
0 0 0 905 0 141519 0 3151214 18610400 200137 27 0 0 0 0 0 0 0 16180913 20585177 14 0 3151214 17  
0 4012000 0 7051404 18012004 4645158 27 0 0 0 0 0 0 0 18052020 1813 27 0 0 0 9130106 4 0 62 0  
3180500 20056108 13010704 91580314 12151860 4012000 58 27 0 0 0 9130106 5641907 15235858 0 0  
39 0 79162008 15140111 1224 0 4091915 120124 0 200804 0 9130106 4 27 0 0 0 19012204 61151619  
91513 0 62 0 9141620 20585167 14 0 251520 0 23011419 0 2014 0 19012204 0 20080918 0 9130106 4 0  
118 0 0 0 80687035 0 58250518 33141558 37 0 5158 27 0 0 0 905 0 19012204 61151619 9151463  
12152304 185858 0 6362 0 46250518 4637 27 0 0 0 0 0 0 39 0 65191920 13091406 0 46091300 70545  
0 918 0 0 0 807375 0 73130106 4 0 15021004 319 0 25152145 2204 0 3180500 200503 0 1517 0 13011408  
16211200 200503 27 0 0 0 0 0 0 6091204 61160119 7 0 62 0 9141620 20585168 14200517 0 200804 0  
16012007 0 2014 0 19012204 0 200804 0 806869 0 6091204 0 58091402 12210408 1406 0 6091204  
14011304 5937 0 5158 27 0 0 0 0 0 0 0 19012204 61091300 7056100 19611603 6915808 13010704 31 0  
6091204 61160119 858 27 0 0 0 0 0 0 0 27 40505 0 3180500 20056108 13010704 91580314 12151860  
4012000 5937 27 0 0 0 515150 0 67180500 2004 0 113 0 9130106 4 0 6181512 0 200804 0 3151214 17 0  
4012000 0 2011904 3 0 1513 0 21190517 0 9141620 19 0 4091304 14190914 1418 0 11403 0 16092404 11  
0 19092604 63 0 515150 27 0 0 0 39 0 68091304 14190914 1418 0 2011904 3 0 1513 0 200804 0  
12051406 2007 0 1505 0 3151214 18610400 2000 0 2014 0 3180500 2004 0 0 0 19172100 1804 0 9130106  
4 27 0 0 0 0 14211360 16092404 12196122 9042007 0 62 0 9142057 12051457 3151214 18610400 200158  
0 5756 0 91649658 0 0 39 0 65191920 13091406 0 0 0 18152106 81224 0 19172100 1804 0 9130106 4 0  
61517 0 19091315 12090308 2024 27 0 0 0 0 14211360 16092404 12196107 5090707 19 0 62 0 14211360  
16092404 12196122 9042007 0 905 0 14211360 16092404 12196122 9042007 0 5756 0 92 0 6362 0  
12051457 3151214 18610400 200158 0 5121904 0 14211360 16092404 12196122 9042007 0 61 0 91 27 0  
0 0 0 27 0 0 0 16092404 12611908 4056111 5140719 7 0 62 0 9290 0 0 39 0 69010307 0 3151214 17 0  
2121502 10 0 23091211 0 204 0 92912491 90 0 16092404 1218 27 27 0 0 0 9130760 23090419 7 0 62 0  
16092404 12611908 4056111 5140719 7 0 56 0 14211360 16092404 12196122 9042007 27 0 0 0  
9130760 8050906 819 0 62 0 16092404 12611908 4056111 5140719 7 0 56 0 14211360 16092404  
12196107 5090707 19 27 27 0 0 0 39 0 67180500 2004 0 0 0 140522 0 9130106 4 0 23092007 0  
23080919 4 0 2010310 7181520 1403 27 0 0 0 0 9130106 4 0 62 0 73130106 5641404 23584681 71664631  
0 58091306 61230903 200831 0 9130760 8050906 8205931 0 46230808 20054658 27 0 0 0 4180122 0 62  
0 73130106 5681800 23646817 1235808 13010704 58 27 27 0 0 0 39 0 68180122 0 5010307 0 2121502  
10 27 0 0 0 0 9140404 23 0 62 0 90 27 0 0 0 61517 0 8 0 913 0 18011406 5581420 13611608 24051218  
61080508 7082058 37 27 0 0 0 0 0 0 61517 0 9 0 913 0 18011406 5581420 13611608 24051218  
61230903 20085937 27 0 0 0 0 0 0 0 0 0 0 0 905 0 9140404 23 0 33 0 12051457 3151214 18610400  
20015937 27 0 0 0 0 0 0 0 0 0 0 0 0 0 20151660 12050619 0 62 0 5809 0 56 0 16092404 12611908  
4056111 5140719 831 0 8 0 56 0 16092404 12611908 4056111 5140719 858 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0



31 0 5240918 20611510 63841820 558 27 27 0 0 0 0 61517 0 12051406 2007 0 913 0 18011406 5581919  
1182060 12051406 200831 0 5140460 12051406 2007 0 61 0 925937 27 0 0 0 0 0 0 0 23092007 0  
15160513 58065143 15212015 21206103 9180502 20151824 45334411 5140719 8456419 24205131 0  
51235158 0 118 0 6091204 37 27 0 0 0 0 0 0 0 0 0 0 16180913 20580650 71051404 18012008 1406 0  
44120513 720844 60030800 18010319 517 0 3151301 9140119 9151418 64646450 58 27 0 0 0 0 0 0 0 0  
0 0 0 14211360 3151301 9140119 9151418 0 62 0 1302007 64161522 58120513 58030800 18010319  
5181958 31 0 12051406 200858 27 0 0 0 0 0 0 0 0 0 0 61517 0 8 0 913 0 18011406 5580913 20581420  
13610314 1320913 1200914 14195958 37 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
5150 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
45301450 58 27 27 0 0 0 0 16180913 20585166 15130208 14012008 151418 0 7051404 18012004 3 0  
19210302 5191905 21121224 645158 27 27 39 0 69240112 161204 0 1505 0 81522 0 2014 0 3011211 0  
200804 0 6211402 20091513 0 58211402 15131304 1419 0 200804 0 6151211 15230913 6 0 12091404 0  
2014 0 211904 0 919 0 913 0 113 0 1032020 111 0 19031808 162058 27 39 0 76120113 7210106 5611314  
4051257 58 27 27 27 40505 0 18050103 61130115 16091406 19585937 27 0 0 0 0 51515181 5010418 0  
13011615 9140718 0 6181512 0 6091204 18 0 14011304 3 0 92642023 19 0 2014 0 95642023 19 0 11403 0  
19201517 518 0 20080512 0 913 0 0 0 12091919 0 1505 0 4090319 9151400 18090518 64515150 27 0 0 0  
0 13011615 9140718 0 62 0 434444 0 61517 0 60 0 913 0 18011406 5589558 41 27 0 0 0 61517 0 8 0 913  
0 18011406 5589231 0 965937 27 0 0 0 0 0 0 0 20182537 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
58065111 12011406 21010704 77150432 13011615 9140718 33440944 64202419 5131 0 51185158 0 118  
0 6091204 37 27 0 0 0 0 0 0 0 0 0 0 0 61517 0 12091404 0 913 0 6091204 37 27 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 16011819 18 0 62 0 12091404 64192017 9165858 64191611 9205858 27 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 905 0 12051457 16011819 1958 0 6362 0 9337 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 9046118 201831 0 22011220 5611919 17 0 62 0 16011819 18 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 13011615 9140718 43096091 42432200 12210560 19201841 0 62 0 9046118 2017 27 0 0 0 0 0 0  
0 0 5240304 1619 0 70091204 78152069 15211403 69181814 1837 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
20580650 69181814 17 0 15160513 91406 0 6091204 0 13011615 9140718 33440944 64202419 5158 27 0  
0 0 0 18052020 1813 0 13011615 9140718 27 27 40505 0 16181502 5191960 6091204 19580913 16212060  
4091804 3201517 2531 0 15212015 21206103 9180502 20151824 31 0 19200117 2031 0 5140431 0  
13011615 9140718 31 0 15212003 9189358 37 27 0 0 0 0 51515179 18150304 19190518 0 6091204 18 0  
23092007 913 0 0 0 7092204 13 0 18011406 4 0 21190913 6 0 16180503 5060913 503 0 13011615  
9140718 0 11403 0 7051404 18012004 18 0 15212015 2119 0 6091204 19645150 50 27 0 0 0 0 13012460  
6091204 61142112 20517 0 62 0 13012457 43091419 58151963 16012007 64191611 9200523 20581518  
64160119 8640200 19051400 13055805 59594390 4258 0 61517 0 5 0 913 0 7121501 64071214 2580650  
44152119 16212060 4091804 3201517 25453356 64202419 51594231 0 4050600 21122062 9158 27 0 0  
0 61517 0 8 0 913 0 18011406 5581919 1182031 0 51403 0 61 0 925937 27 0 0 0 0 0 0 0 0 9141620  
20610608 12056115 12007 0 62 0 6514408 14162119 61040917 5032014 18254532 44094563 20242050  
27 0 0 0 0 0 0 0 20182537 27 0 0 0 0 0 0 0 0 0 0 16181502 5191960 12011806 5610608 12055808  
14162119 61060911 5611600 200831 0 15212015 21206103 9180502 20151824 31 0 13011615 9140718  
31 0 13012460 6091204 61142112 2051858 27 0 0 0 0 0 0 0 0 0 0 16181502 5191960 12011806  
5610608 12059357 9141620 20610608 12056115 1200831 0 15212003 9189331 0 13011615 9140718 31 0  
13012460 6091204 61142112 2051858 27 0 0 0 0 0 0 0 0 0 0 13012460 6091204 61142112 205170  
6262 0 91 27 0 0 0 0 0 0 0 0 5240304 1619 0 70091204 78152069 15211403 69181814 1837 27 0 0 0 0 0 0  
0 0 0 0 0 16180913 20580650 67152111 3 0 141519 0 15160513 0 200804 0 6091204 0 44091415  
21206105 9120560 16012007 455158 27 27 40505 0 16181502 5191960 12011806 5610608 12055808  
14162119 61060911 5611600 200831 0 15212015 21206103 9180502 20151824 31 0 13011615 9140718  
31 0 6091204 61142112 2051858 37 27 0 0 0 0 51515168 6060902 9051419 1224 0 16181502 5191904 18









5131619 24 0 6091204 645158 27 0 0 0 0 0 0 0 0 12091404 18 0 62 0 4341 27 27 0 0 0 0 23080911 4 0  
84182104 37 27 0 0 0 0 0 0 0 0 16180913 20585129 149263 0 650403 0 20052419 5158 27 0 0 0 0 0 0 0  
16180913 20585192 63 0 86090522 0 20052419 5158 27 0 0 0 0 0 0 0 0 16180913 20585193 63 0  
85160400 2004 0 0 0 12091404 5158 27 0 0 0 0 0 0 0 0 16180913 20585194 63 0 68051204 2004 0 0 0  
12091404 5158 27 0 0 0 0 0 0 0 16180913 20585195 63 0 83012204 0 3080113 7051950 58 27 0 0 0 0 0 0  
0 0 16180913 20585196 63 0 69240919 0 23092007 152119 0 19012208 14075158 27 0 0 0 0 0 0 0  
3081508 304 0 62 0 9141620 20585166 8151518 4 0 113 0 15162008 151437 0 5158 27 27 0 0 0 0 0 0 0  
905 0 3081508 304 0 6362 0 46924637 27 0 0 0 0 0 0 0 0 0 0 0 0 20052419 62 0 9141620 20585168  
14200517 0 25152117 0 20052419 37 0 5158 27 0 0 0 0 0 0 0 0 0 0 0 0 12091404 19640115 16051403  
58200523 2058 27 0 0 0 0 0 0 0 0 0 0 0 0 16180913 20585183 52419 0 1040404 3 0 19210302 5191905  
21121224 645158 27 0 0 0 0 0 0 0 5120905 0 3081508 304 0 6362 0 46934637 27 0 0 0 0 0 0 0 0 0 0  
905 0 12091404 1937 27 0 0 0 0 0 0 0 0 0 0 0 0 16180913 20585129 14672117 18051419 0  
84052419 385158 27 0 0 0 0 0 0 0 0 0 0 0 0 61517 0 931 0 12091404 0 913 0 5142112 5180119  
5581208 14051931 0 925937 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16180913 20580650 44094563 0  
44120913 5455158 27 0 0 0 0 0 0 0 0 0 0 0 0 5121904 37 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 16180913  
20585183 804 0 20052419 0 918 0 5131619 25645158 27 0 0 0 0 0 0 0 0 5120905 0 3081508 304 0 6362 0  
46944637 27 0 0 0 0 0 0 0 0 0 0 0 0 12091404 61142112 0 62 0 9142057 9141620 20585168 14200517 0  
200804 0 12091404 0 14211301 517 0 2014 0 21160400 200537 0 515958 0 59 0 91 27 0 0 0 0 0 0 0 0 0 0 0  
0 905 0 90 0 3462 0 12091404 61142112 0 33 0 12051457 12091404 195937 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
0 14052360 20052419 0 62 0 9141620 20585168 14200517 0 200804 0 140522 0 20052419 37 0 5158 27 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 12091404 19431208 14056113 211341 0 62 0 14052360 20052419 27 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 16180913 20585175 91404 0 21160400 200503 0 19210302 5191905 21121224  
645158 27 0 0 0 0 0 0 0 0 0 0 5121904 37 27 0 0 0 0 0 0 0 0 0 0 0 0 0 16180913 20585172  
14220111 903 0 12091404 0 14211301 5186450 58 27 0 0 0 0 0 0 0 0 5120905 0 3081508 304 0 6362 0  
46954637 27 0 0 0 0 0 0 0 0 0 0 12091404 61142112 0 62 0 9142057 9141620 20585168 14200517 0  
200804 0 12091404 0 14211301 517 0 2014 0 4051204 200537 0 515958 0 59 0 91 27 0 0 0 0 0 0 0 0 0 0  
0 905 0 90 0 3462 0 12091404 61142112 0 33 0 12051457 12091404 195937 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
40511 0 12091404 19431208 14056113 211341 27 0 0 0 0 0 0 0 0 0 0 0 0 0 16180913 20585175  
91404 0 4051204 200503 0 19210302 5191905 21121224 645158 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5121904 37 27  
0 0 0 0 0 0 0 0 0 0 0 0 16180913 20585172 14220111 903 0 12091404 0 14211301 5186450 58 27 0  
0 0 0 0 0 0 5120905 0 3081508 304 0 6362 0 46964637 27 0 0 0 0 0 0 0 0 0 0 23092007 0 15160513  
58060911 5611600 200831 0 46234658 0 118 0 6091204 37 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 61517 0  
12091404 0 913 0 12091404 1937 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6091204 64231808 20055805  
51441208 14054529 145158 0 0 39 0 650403 0 14052311 91404 0 3080117 1032004 1818 0 23080513 0  
23180919 91406 27 0 0 0 0 0 0 0 0 0 16180913 20585166 8011406 518 0 19012204 3 0 19210302  
5191905 21121224 645158 27 0 0 0 0 0 0 0 0 0 2180500 10 0 39 0 69240919 0 1062004 17 0  
19012208 1406 27 0 0 0 0 0 0 0 0 5120905 0 3081508 304 0 6362 0 46974637 27 0 0 0 0 0 0 0 0 0 0  
16180913 20585168 24092008 1406 0 23092007 152119 0 19012208 1406 0 3080113 7051963 5158 27 0  
0 0 0 0 0 0 0 0 0 0 2180500 10 27 0 0 0 0 0 0 0 0 5121904 37 27 0 0 0 0 0 0 0 0 0 0 16180913 20585172  
14220111 903 0 3081508 30563 0 80120500 1904 0 201824 0 1070108 14645158 27 27 27 40505 0  
9130760 7051404 18012014 18585937 27 0 0 0 0 16180913 20585117 1140414 135858 0 37 0 5131 0  
18011403 15136417 1140414 13585958 27 0 0 0 0 13011919 517 0 62 0 84115858 27 0 0 0 0 13011919  
5186400 20201808 2212004 19584659 6211211 19031804 5144631 0 84182104 58 27 0 0 0 0 4000 0 62 0  
939690 27 0 0 0 0 4001 0 62 0 939190 27 0 0 0 0 16180913 20585186 5120314 13053013 5158 27 0 0 0 0  
16180913 20585183 91637 0 19090404 0 23090419 7 0 19081520 1203 0 204 0 0 6010319 1517 0 1505 0  
200804 0 9130106 4 0 23090419 831 0 200804 0 19011304 0 7150518 0 61517 0 200804 0 9130106 4 0  
8050906 819 0 913 0 18051200 20091513 0 2014 0 200804 0 19090404 0 8050906 819 0 1505 0 5010307 0  
16092404 12645158 27 0 0 0 0 3080113 70591 0 62 0 9141620 20585168 14200517 0 19090404 0  
23090419 7 0 1505 0 9130106 4 0 2121502 1137 0 0 5158 27 0 0 0 0 3080113 704 0 62 0 9142057 3080113  
7059258 27 0 0 0 0 3080113 70592 0 62 0 9141620 20585168 14200517 0 19090404 0 8050906 819 0 1505

0 9130106 4 0 2121502 1137 0 0 5158 27 0 0 0 0 3080113 7056107 0 62 0 9142057 3080113 7059358 27 0  
0 0 0 4000 0 62 0 92999990 27 0 0 0 0 4001 0 62 0 92919690 0 27 0 0 0 0 191 0 62 0 9141620 20585168  
14200517 0 23090419 7 0 1505 0 9130106 537 0 5158 27 0 0 0 0 0 62 0 9142057 19258 27 0 0 0 291 0  
62 0 9141620 19 0 58516913 200517 0 8050906 819 0 1505 0 9130106 537 0 0 5158 27 0 0 0 1 0 62 0  
9142057 29258 27 0 0 0 0 16091460 1691 0 62 0 1330307 1140704 27 0 0 0 0 16091460 1692 0 62 0  
2330307 1140704 6107 27 0 0 0 0 22 0 62 0 67011421 1195812 1192004 1831 0 23090419 862 0 131 0  
8050906 82062 0 258 27 27 0 0 0 0 4002 0 62 0 43511620 18161204 5131 0 51071804 5145131 0  
51071511 45131 0 51180503 5131 0 51250511 12152350 31 0 51151800 14070550 31 0 51160913 115131  
0 51021814 23145131 0 51032500 145131 0 51120912 55131 0 51200500 125131 0 51130106 5142000  
5141 27 0 0 0 0 2 0 62 0 4341 27 0 0 0 0 23080911 4 0 84182104 37 27 0 0 0 0 0 0 0 1161660 51403 0 62  
0 9141620 20585168 14200517 0 3151214 2117 0 720523 0 23092007 0 39 0 1517 0 3151214 2117 0  
14011304 37 0 5158 27 0 0 0 0 0 0 0 3640115 16051403 58011615 61051403 58 27 0 0 0 0 0 0 0 0 27 0 0  
0 0 0 0 0 3151419 960 0 62 0 9141620 20585168 14200517 0 91 0 2014 0 10403 0 1141519 80517 0  
3151214 211831 0 90 0 2014 0 13152204 0 151437 0 5158 27 0 0 0 0 0 0 0 905 0 3151419 960 0 5062 0  
46924637 0 0 39 0 66180500 1118 0 200804 0 12151515 0 905 0 9141620 19 0 918 0 141519 0 469245 27  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 2180500 10 27 0 0 0 2402 0 62 0 90 27 0 0 0 0 26051814 23 0 62 0 90 27 0 0 0 0  
26051814 24 0 62 0 90 27 0 0 0 15 0 62 0 91 27 0 0 0 18011406 5610614 17 0 62 0 9142057 58013302  
8011406 5595757 2330307 1140704 61085958 27 0 0 0 14011304 0 62 0 91 27 0 0 0 0 3051211 18 0 62 0  
58580132 33030800 14070558 57580232 33030800 14070560 85958 27 0 0 0 21161604 17 0 62 0  
3051211 18 0 59 0 91 27 0 0 0 0 14021860 3151301 0 62 0 13012007 64161522 58120513 58035931  
3051211 1958 27 0 0 0 0 6091204 18 0 62 0 9142057 14021860 3151301 58 27 0 0 0 0 40161808 14205811  
5145802 5958 27 0 0 0 0 18152313 0 62 0 90 27 0 0 0 0 9130760 6146115 18050608 23 0 62 0 9141620  
20585168 14200517 0 73130106 4 0 6091204 14011304 0 16180505 923 0 58151308 19 0 200804 0  
6091204 0 5242004 14190914 13 0 14011304 5937 0 5158 27 0 0 0 0 6091204 61031520 1419 0 62 0 90 0  
0 0 27 0 0 0 0 61517 0 19 0 913 0 18011406 5589131 6091204 195937 27 0 0 0 0 0 0 0 6091204  
61031520 1419 0 62 0 6091204 61031520 1419 0 61 0 91 27 0 0 0 0 0 0 0 0 40090657 6091204 61031520  
1419 0 6362 0 92939258 37 27 0 0 0 0 0 0 0 0 39 0 0 0 0 2180500 10 27 0 0 0 0 0 0 0 0 19230919 307 0 62 0  
91 27 0 0 0 0 0 0 0 1922 0 62 0 90 27 0 0 0 0 0 0 0 0 40061517 0 23 0 913 0 18011406 5581800 14070560  
6151858 37 27 0 0 0 0 0 0 0 0 0 40061517 0 23 0 913 0 18011406 4 0 58913202 5121218 5937 27 0 0 0 0 0  
0 0 23 0 62 0 90 27 0 0 0 0 0 0 0 31511 0 62 0 3051211 18 0 59 0 91 27 0 0 0 0 0 0 0 0 40161808  
14205817 15231458 27 0 0 0 0 0 0 0 0 23080911 55823 0 33 0 3051211 195937 27 0 0 0 0 0 0 0 0 0 0 0 0 27  
0 0 0 0 0 0 0 0 0 0 0 0 4005 0 62 0 15160513 58465418 64161945 0 53 0 14011304 31 0 46230245 58 27 0 0  
0 0 0 0 0 0 0 0 0 0 0 40066402 12151904 27 0 0 0 0 0 0 0 0 0 0 9065818 23092002 7 0 6362 0 925937 27 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 181522 0 62 0 91 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 14241204 619 0 62 0 90 27 0  
0 0 0 0 0 0 0 0 0 0 0 0 0 0 14241808 70819 0 62 0 3080113 704 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 14251204  
619 0 62 0 90 27 0 0 0 0 0 0 0 0 0 0 0 0 14251808 70819 0 62 0 3080113 7056107 27 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 26051814 23 0 62 0 90 27 0 0 0 0 0 0 0 0 0 0 0 0 0 26051814 24 0 62 0 90 27 0 0 0 0 0 0  
0 0 0 0 0 3611204 14072007 0 62 0 12051457 358 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 0 0 19230919 307 0 62 0 90  
27 0 0 0 0 0 0 0 0 0 0 0 40180113 0 62 0 18011403 15136417 1140408 14205890 32036111 5140719 7 0  
59 0 9258 27 27 0 0 0 0 0 0 0 0 0 0 18040921 0 62 0 13012007 64161522 58120513 58035931 3151258  
27 0 0 0 0 0 0 0 0 0 0 0 3051211 0 62 0 58181522 14331803 92258 0 53 0 58120513 58035958 27 27 0 0 0  
0 0 0 0 0 0 0 0 0 3051211 503 0 62 0 9142057 3051211 58 27 0 0 0 0 0 0 0 0 0 0 0 16180913 20580304  
12120503 58 27 27 0 0 0 0 0 0 0 0 0 0 40236402 18050119 5611804 3200113 7120557 26051814 2431  
0 14251204 62031 0 14241808 7082031 14251808 7082031 0 6091211 0 62 0 3430304 12120503 4231 0  
15212011 91404 0 62 0 51021200 3115131 0 23090419 7 0 62 0 9158 27 0 0 0 0 0 0 0 0 0 0 0 23640317  
5012004 61180502 20011406 12055825 5181523 31 0 14251204 62031 0 14241808 7082031 14251808  
7082031 0 6091211 0 62 0 3430304 12120503 4231 0 15212011 91404 0 62 0 51021200 3115131 0  
23090419 7 0 62 0 9158 27 0 0 0 0 0 0 0 0 0 0 0 9065823 0 3462 0 3051211 195937 27 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 31511 0 62 0 31511 0 59 0 91 27 27 0 0 0 0 0 0 0 0 0 0 0 0 23640717 9045817 1522 0 62 0  
26051814 2531 0 3151220 1313 0 62 0 26051814 23 0 61 0 3080113 70558 27 0 0 0 0 0 0 0 0 0 0 0 0



17 27 27 27 0 0 0 0 21218 0 62 0 9692 27 0 0 0 0 19031804 513 0 62 0 16250700 13056403 9191611  
1256418 5206112 15040557 58190317 5051460 23090419 831 0 19031804 5146107 5090707 205958 27 0  
0 0 0 40190317 50513 0 62 0 16250700 13056403 9191611 1256418 5206112 15040557 58190317  
5051460 23090419 831 0 19031804 5146107 5090707 205931 0 16250700 13056481 69837389 65667668  
58 27 0 0 0 0 40404039 40404039 40404039 40404039 4039 27 27 0 0 0 0 40671804 12004 0 67151312  
1140418 27 0 0 0 0 2211960 0 62 0 939695 27 0 0 0 0 1131520 142060 0 62 0 58022118 60 0 61 0 9258 0  
56 0 58022118 60 0 61 0 9258 27 0 0 0 0 12091404 61142112 2051860 0 62 0 90 27 0 0 0 0 0 0 0 0 0 0 0 0  
0 0 0 27 0 0 0 0 3180500 20056105 0 62 0 15160513 58516714 13130113 4618404 13161200 20056419  
24205131 0 51235158 27 0 0 0 0 23080911 5581208 14056113 21130204 1860 0 33 0 1131520 14206158  
37 27 0 0 0 0 0 0 0 4008 0 62 0 12150791 91581208 14056113 21130204 1858 27 0 0 0 0 0 0 0 0 3180500  
20056105 64231808 20055850 16180913 20583050 43691315 2024 0 67151312 11403 0 83121519 41 0  
58670800 140704 0 21190913 6 0 0 0 20052419 60050408 201517 0 2014 0 85160400 2004 0 20080918 0  
19121519 0 913 0 67151312 1140460 84051315 12012004 64202419 305158 0 40180512 5130204  
18091406 0 2014 0 18051400 1304 0 67151312 1140460 84051315 12012004 64202419 30145158 27 0 0 0  
0 0 0 0 0 12091404 61142112 2051860 0 62 0 12091404 61142112 2051860 0 61 0 91 27 27 0 0 0  
3180500 20056105 64031214 19055858 27 27 0 0 0 0 40761500 4671512 13011403 18 27 0 0 0 0 240913 0  
62 0 90 27 0 0 0 0 6091213 112 0 62 0 51671512 13011403 61840512 16120119 5642023 2050 27 0 0 0  
23092007 0 15160513 58060911 14011331 0 46184658 0 118 0 6091204 37 27 0 0 0 0 0 0 0 3151312  
1140418 0 62 0 6091204 64180500 4120913 5195858 27 0 0 0 0 0 0 0 0 240913 0 62 0 240913 0 61 0 91 27  
27 27 0 0 0 0 19200117 0 62 0 439141 27 27 0 0 0 0 61517 0 931 0 3151312 11403 0 913 0 5142112  
5180119 5580314 13130113 4195937 27 0 0 0 0 0 0 0 3151312 1140418 430941 0 62 0 3151312 1140463  
19201808 165858 27 0 0 0 0 0 0 0 40031303 0 62 0 3151312 1140463 19161208 20585131 5158 27 0 0 0  
0 0 0 0 0 19200117 64011615 5140457 3151312 1140458 27 0 0 0 0 40511 0 19200117 439141 27 27 0 0 0  
0 40404039 40404039 40404039 40404039 40404039 27 27 0 0 0 0 40404039 40404039 40404039 40404039  
4039 27 27 0 0 0 0 40761500 4671514 18040913 1200518 27 0 0 0 0 24091492 0 62 0 90 27 0 0 0  
23092007 0 15160513 58466714 15180408 14012004 19618024 20081513 64202419 4631 0 46184658 0  
118 0 6091204 9337 27 0 0 0 0 0 0 3151312 1140418 92 0 62 0 6091204 93641804 1041208 14051957  
58 27 0 0 0 0 0 0 0 24091492 0 62 0 24091492 0 61 0 91 27 27 27 0 0 0 0 19200117 1923 0 62 0 439141  
27 0 0 0 0 19200117 1924 0 62 0 439141 27 27 0 0 0 0 61517 0 99331 0 3151312 1140492 0 913 0 5142112  
5180119 5580314 13130113 4199358 37 27 0 0 0 0 0 0 0 3151312 1140418 93430992 41 0 62 0 3151312  
1140492 64192017 9165858 27 0 0 0 0 0 0 0 3130492 0 62 0 3151312 1140492 64191611 9205850  
325158 27 0 0 0 0 0 0 0 19200117 19246400 16160513 4580913 20580312 4934391 425958 27 0 0 0 0 0  
0 0 0 22011824 18 0 62 0 9142057 3130492 43934263 19201808 16585958 27 0 0 0 0 0 0 0 19200117  
19256400 16160513 4582200 18251958 27 0 0 0 0 0 0 0 27 0 0 0 0 40511 0 19200117 19244390 41 27 0  
0 0 0 40511 0 19200117 19254390 41 27 27 0 0 0 0 515150 27 0 0 0 0 61517 0 99431 0 3151312 1140424 0  
913 0 19200117 192537 27 0 0 0 0 0 0 0 140522 0 62 0 19200117 19254308 94426418 20180915 5858 27  
0 0 0 0 0 0 0 19200117 19254308 9441 0 62 0 9142057 14052358 27 0 0 0 0 515150 27 27 0 0 0  
16180913 20581919 1181923 58 27 0 0 0 0 16180913 20581919 1181924 58 27 0 0 0 0 40404039  
40404039 40404039 40404039 40404039 27 27 27 0 0 0 0 39 0 76150103 0 9130106 518 27 0 0 0 0 2121502  
11610912 1070518 0 62 0 4341 27 0 0 0 0 61517 0 8 0 913 0 18011406 5589395 975937 27 0 0 0 0 0 0  
2121502 11610912 1070518 64011615 5140457 16250700 13056408 13010704 64121500 4580650  
2121502 11440961 92456415 14075158 58 27 27 27 0 0 0 0 39 0 77012604 0 12012514 2119 27 0 0 0  
13012604 61120124 152119 0 62 0 42 27 0 0 0 0 0 0 0 439131 0 9231 0 9331 0 944231 27 0 0 0 0 0 0 0  
439531 0 9631 0 9731 0 984231 27 0 0 0 0 0 0 0 439931 0 10031 0 929131 0 92924231 27 0 0 0 0 0 0 0  
43929331 0 929431 0 929531 0 929641 27 0 0 0 0 41 27 27 27 0 0 0 0 40702113 3200914 13 0 2014 0  
3080502 10 0 905 0 0 0 16151908 20091513 0 918 0 9141908 404 0 200804 0 13012604 27 0 0 0 0 40505 0  
9196108 14190903 5611300 26055823 31 0 2531 0 13012604 61230903 200831 0 13012604 61080508  
7082031 0 2121958 37 27 0 0 0 0 0 0 0 18052020 1813 0 90 0 3462 0 23 0 33 0 13012604 61230903 2007  
0 56 0 21218 0 11403 0 90 0 3462 0 24 0 33 0 13012604 61080508 70819 0 56 0 21218 27 27 0 0 0 0 39 0  
76150103 0 16120124 517 0 9130106 518 27 0 0 0 0 16120124 5186108 13010704 0 62 0 16250700

13056408 13010704 64121500 4585115 12012504 18641613 75158 27 0 0 0 16120124 5186108  
13010704 92 0 62 0 16250700 13056408 13010704 64121500 4585115 12012504 18936415 14075158 27  
27 0 0 0 0 40190300 12050460 2121502 11610912 1070518 0 62 0 43162506 1130563 20180113 19061517  
13641902 1120557 9130731 0 58091419 58091306 64070519 61230903 20085858 0 56 0 19030111  
5610600 3201517 5931 0 9142057 9130763 7052060 8050906 8205858 0 56 0 19030111 5610600  
3201517 595958 0 61517 0 91306 0 913 0 2121502 11610912 1070518 41 27 0 0 0 0 40190300 12050460  
16120124 5186108 13010704 0 62 0 16250700 13056419 18011418 6151812 64190300 12055815  
12012504 18610912 1070531 0 58091419 58161200 25051860 9130106 5640704 20612308 4200857 58 0  
56 0 19030111 5610600 3201517 5931 0 9142057 16120124 5186108 13010704 64070519 61080508  
7082057 58 0 56 0 19030111 5610600 3201517 595958 27 27 27 0 0 0 39 0 830519 0 16120124 517 0  
16151908 20091513 27 0 0 0 0 16120124 5186123 0 62 0 90 27 0 0 0 0 16120124 5186124 0 62 0 90 27 0  
0 0 3151460 16181506 180112 0 62 0 9142057 9141620 20585166 15142008 14210535 0 91 0 43250518  
4231 0 90 0 43141541 37 0 515958 27 0 0 0 0 9065802 15146115 18150717 112 0 6362 0 925937 27 0 0 0  
0 0 0 0 0 16180913 20585178 10 0 64646450 58 27 0 0 0 0 9065802 15146115 18150717 112 0 6362 0  
915937 27 0 0 0 0 0 0 0 0 16180913 20585168 24092008 1406 0 64646450 58 27 0 0 0 0 0 0 0 0 0 5240919  
5858 27 0 0 0 0 9065802 15146115 18150717 112 0 5062 0 90 0 11403 0 3151460 16181506 180112 0  
5062 0 925937 27 0 0 0 0 0 0 16180913 20585168 18181517 5158 27 0 0 0 0 0 0 0 5240919 5858 27 0  
0 0 0 39 0 830519 0 9140919 90111 0 9141620 19 0 20251604 27 0 0 0 0 20251604 8 0 62 0 511150 27 0 0  
0 0 16180913 20585129 14301472 13 0 76150300 11 0 67151312 11403 0 77150404 30143013 5158 27 0 0  
0 0 16180913 20585182 23092002 7 0 67151312 11403 0 77012017 923 0 43194231 3013 0 13152118 4 0  
43134231 30141104 25021500 1803 0 1181814 2318 0 43114231 30146859 800103 0 58880214 23 0  
791404 0 67151419 18151211 51858 0 43044229 14801804 1918 0 43838064 67696664 8241 0 2014 0  
3011211 0 0 0 3151312 11403 0 6181512 0 67151312 1140418 61840512 16120119 5642023 19 0  
58671512 13011403 18 0 30113 0 204 0 3080113 70503 0 21190913 6 0 0 0 20052419 60050408 20151858  
0 0 646463 0 5158 27 0 0 0 0 16180913 20585169 1517 0 13150404 0 460445 0 64 0 58192101 60130119  
180923 0 19200117 19 0 18050708 151458 0 84080513 0 64 0 58192101 60130119 180923 0 51403 0  
18050708 15145963 0 80180518 18 0 19200117 19 0 61517 0 8051215 0 58830502 211804 0 73142004  
18140519 0 65030304 1918 0 82051720 9180503 59645158 27 0 0 0 39 0 19151304 0 9141620 19 0  
22011808 1021204 27 0 0 0 0 9141620 20612200 18090101 1204 0 62 0 90 27 0 0 0 0 39 0 10403 0 200804  
0 3211919 1512 0 5220513 19 0 2014 0 200804 0 5220513 19 0 17210520 4 27 0 0 0 0 16250700 13056404  
22051419 64161518 20581624 7011304 64052204 14206468 22051419 58778960 67858383 79776168  
86697883 31 0 44510913 16212060 22011808 1021204 5137 0 9141620 20612200 18090101 12054558 58  
27 0 0 0 0 40052002 7 0 12151515 27 0 0 0 0 5200307 0 62 0 90 27 27 0 0 0 0 40130125 4 0 23090419 7 27  
0 0 0 0 13012604 61230903 2007 0 62 0 9296 27 0 0 0 0 13012604 61080508 70819 0 62 0 9296 27 0 0 0  
39 0 77010913 0 7011304 0 12151515 27 0 0 0 0 18211413 91406 0 62 0 84182104 27 0 0 0 0 40140522 0  
19031804 513 0 19092604 0 19052019 9140718 27 0 0 0 0 40401902 18050513 61230903 200831 0  
19031804 5146107 5090707 19 0 62 0 16250700 13056403 9191611 1256406 5206118 21180600 3055858  
64070519 61190925 55858 27 0 0 0 0 19030111 5610600 3201517 6123 0 62 0 19031804 5146122  
9042007 0 32 0 13012604 61230903 2007 27 0 0 0 0 40190300 12056105 1032014 186124 0 62 0  
19031804 5146107 5090707 19 0 32 0 13012604 61080508 70819 27 0 0 0 0 40190300 12056105 1032014  
17 0 62 0 13091457 19030111 5610600 3201517 612431 0 19030111 5610600 3201517 612558 27 0 0 0  
19030111 5610600 3201517 0 62 0 19030111 5610600 3201517 6123 27 27 0 0 0 0 21218 0 62 0 9142057  
21218 0 56 0 91649658 27 27 0 0 0 0 19030111 5046101 12150310 61091300 70518 0 62 0 4341 27 27 0 0  
0 0 39 0 66050614 1804 0 200804 0 13010913 0 12151515 27 0 0 0 0 9196118 5120502 20091406  
61192101 13012017 923 0 62 0 70011218 4 27 0 0 0 0 19210212 1201808 24611919 1182060 161518 0 62  
0 78151404 27 0 0 0 0 19051204 3200503 61031512 13011403 0 62 0 78151404 27 0 0 0 0 5240502  
21200560 3151312 1140460 16181512 1619 0 62 0 70011218 4 27 27 0 0 0 0 61517 0 1408 0 913 0  
18011406 5589395 975937 27 0 0 0 0 0 0 0 196108 0 62 0 16250700 13056419 18011418 6151812  
64190300 12055801 12150310 61091300 7051942 14094231 0 58021218 31 0 2121958 58 27 0 0 0 0 0 0  
0 2121502 11610912 1070518 43140941 0 62 0 196108 27 27 27 0 0 0 0 23080911 4 0 18211413













67151912 15121506 24 0 58851904 1021204 0 23092007 0 9131 0 91 0 11403 0 93595158 27 0 0 0 0  
16180913 20585168 14200517 0 9537 0 83051720 5142008 111 0 76011406 21010704 0 71051404  
18012008 1513 0 58870117 14091406 31 0 251520 0 13211919 0 6091219 517 0 152119 0 200804 0  
16181505 1140563 0 65121914 0 0 0 21190517 0 64202419 0 6091204 0 13211919 0 5240918 19 0  
23092007 913 0 200804 0 23151810 91406 0 4091804 3201517 24 0 20080119 0 12091919 18 0 58151404  
0 3080117 1032004 17 0 160517 0 12091404 58 0 85140902 150404 0 3080117 1032004 181937 0 5158 27  
0 0 0 0 16180913 20585168 14200517 0 9637 0 84052419 0 69040919 15185158 27 0 0 0 0 16180913  
20585168 14200517 0 9737 0 71051404 18012004 0 67151312 11403 0 77012017 923 0 80122106 913 0  
61517 0 935158 27 0 0 0 16180913 20585168 14200517 0 9837 0 69240919 0 80181506 18011350 58 27  
0 0 0 0 16180913 20585168 14200517 0 9937 0 9467 0 676567 0 676576 0 69140708 14055158 27 0 0 0 0  
16180913 20585168 14200517 0 10037 0 81210113 202112 0 67091802 210919 0 81091910 919 0  
87151810 66051402 85158 27 0 0 0 0 16180913 20585168 14200517 0 929137 0 67120118 19090300 11 0  
67091802 210919 0 83030804 13681800 22 0 87151810 66051402 85158 27 0 0 0 0 16180913 20585168  
14200517 0 929237 0 67151912 15121506 9030111 0 83091320 12012008 1513 0 224 0 84052419 5158 27  
0 0 0 0 16180913 20585168 14200517 0 929337 0 76011806 4 0 76011406 21010704 0 77150404 125158  
27 0 0 0 0 16180913 20585168 14200517 0 929437 0 68012000 0 83031808 1619 0 224 0 67211919 1512 0  
83201820 3202117 5195158 27 0 0 0 0 16180913 20585168 14200517 0 929537 0 71051404 18012004 0  
66091400 1824 0 85140916 2104 0 83201808 140718 0 2014 0 0 0 3211919 15136419 2419 0 6091204  
5158 27 0 0 0 0 16180913 20585168 14200517 0 929737 0 15212001 0 6091204 0 2014 0 2091400  
18251520 19 0 6091204 0 58692419 5141908 1513 0 1505 0 92935950 58 27 0 0 0 0 16180913 20585168  
14200517 0 929837 0 84052419 0 68012000 2011904 5158 27 0 0 0 0 16180913 20585168 14200517 0  
929937 0 80181523 9130111 0 80180912 4 0 78211301 5181950 58 27 0 0 0 0 16180913 20585168  
14200517 0 930037 0 67151405 9072117 1200914 1418 0 61517 0 77150404 0 935158 27 0 0 0 0  
16180913 20585168 14200517 0 939237 0 80181523 9130111 0 80180912 518 0 65140912 1201517 5158  
27 0 0 0 0 16180913 20585168 14200517 0 939337 0 77011420 111 0 84211808 1406 0 67151315  
12052004 0 70091408 2004 0 83200119 4 0 77010307 9140550 58 27 0 0 0 0 16180913 20585168  
14200517 0 939437 0 65212014 13012008 2 0 84211808 1406 0 67151315 12052004 0 70091408 2004 0  
83200119 4 0 77010307 9140550 58 27 0 0 0 0 16180913 20585168 14200517 0 939537 0 80252007 1513  
0 69140101 120503 0 67151312 11403 0 76091404 0 73142004 18060102 55158 27 0 0 0 0 16180913  
20585168 14200517 0 939637 0 65212014 13012008 2 0 67151419 181511 0 69140708 14050517  
9140750 58 27 0 0 0 0 16180913 20585168 14200517 0 939737 0 65212014 13012004 3 0 68051908 713 0  
11403 0 84050307 14151214 7255158 27 0 0 0 0 16180913 20585168 14200517 0 939837 0 67151418  
20182102 19 0 25152117 0 152313 0 76091406 21091919 902 0 76011406 21010704 5158 27 0 0 0 0  
16180913 20585168 14200517 0 939937 0 67151421 51819 0 84052419 0 69140314 4091406 0 58701814  
12 0 929358 0 2014 0 0 0 67151421 5180704 1419 0 73130106 55158 27 0 0 0 0 16180913 20585168  
14200517 0 940037 0 77012007 5130119 90318 0 76010250 58 27 0 0 0 0 16180913 20585168 14200517 0  
949137 0 77211219 9607600 14072100 704 0 67080117 1032004 17 0 77011615 5185158 27 0 0 0 0 0 0  
0 27 23080911 5588417 2105937 27 0 0 0 0 27 0 0 0 0 5142017 1140304 62 0 9142057 9141620  
20585129 14301429 20691419 517 0 25152117 0 13150404 0 1505 0 15160517 1200914 13 0 589295 0  
61517 0 1220108 12010211 4 0 15162008 15141958 37 0 515958 27 0 0 0 0 9065804 14201800 140304 0  
6362 0 915937 27 0 0 0 0 0 0 0 9130760 7051404 18012014 185858 27 0 0 0 0 9065804 14201800  
140304 0 6362 0 925937 27 0 0 0 0 0 0 0 3151315 9120560 9130106 5610608 12055858 27 0 0 0 0  
9065804 14201800 140304 0 6362 0 935937 27 0 0 0 0 0 0 0 16181506 18011312 9140760 5140708  
14055858 27 0 0 0 0 9065804 14201800 140304 0 6362 0 945937 27 0 0 0 0 0 0 0 3211817 5142060  
31511 0 62 0 90 27 0 0 0 0 0 0 3211817 5142060 181522 0 62 0 90 27 0 0 0 0 0 0 0 0 18051908  
26050460 9130106 518 0 62 0 4341 0 0 39 0 71121501 111 0 22011808 1021204 0 2014 0 19201517 4 0  
18051908 260503 0 9130106 518 27 27 0 0 0 0 0 0 0 39 0 82051919 0 1505 0 25152117 0 3150404  
646463 27 0 0 0 0 0 0 0 39 0 71121501 111 0 22011808 1021204 0 61517 0 200804 0 9130106 4 0  
19051204 3200914 13 0 6180112 4 27 0 0 0 0 0 0 0 9130106 5611904 12050319 9151460 6180112 4 0 62  
0 78151404 27 0 0 0 0 0 0 0 0 19051204 3200503 61091300 704 0 62 0 78151404 0 0 39 0 71121501 111 0



































20152408 3151214 7255131 27000000000000000000 51691421 9181513 13051419 1110 3080512  
9192017 255131 27000000000000000000 51691421 9181513 13051419 1110 13090317 15020914  
12150724 5131 27000000000000000000 51690314 20152408 3151214 7255131 270000000000  
00000000 51711804 5130 3080512 9192017 255131 27000000000000000000 51831600 3040  
16082518 9031950 31 27000000000000000000 51831600 3040 23050119 8051850 31 27000000  
000000000000 51651919 18151919 1200918 20090318 5131 2700000000000000000000000000  
51671512 16212000 20091513 1110 6122108 304251400 13090318 5131 27000000000000000000  
51770119 8051300 20090300 110 15162008 13092600 20091513 5131 27000000000000000000  
51791604 18012008 1514180 18051904 1180307 5131 27000000000000000000 51722112 1130  
7051404 20090318 5131 27000000000000000000 51702113 3200914 1401110 7051414 13090318  
5131 27000000000000000000 51771511 5032111 1170 7051404 20090318 5131 270000000000  
00000000 51670113 305170 7051404 20090318 5131 27000000000000000000 51801924  
3080900 20180902 0 7051404 20090318 5131 27000000000000000000 51801515 21120119 91513  
0 7051414 13090318 5131 27000000000000000000 51660914 5140708 14050517 9140750 31 270  
0000000000000000 51660914 13012004 18090111 195131 27000000000000000000  
51660914 13050307 1201814 14090318 5131 27000000000000000000 51670117 4091521 1190320  
1201170 5140708 14050517 9140750 31 27000000000000000000 51780520 1801110 5140708  
14050517 9140750 31 27000000000000000000 51820507 1020911 9200119 91513 0 5140708  
14050517 9140750 31 27000000000000000000 51710513 5200902 0 5140708 14050517 9140750  
31 27000000000000000000 51691421 9181513 13051419 1110 5140708 14050517 9140750 31 27  
0000000000000000 51870119 5170 18051914 21180304 180 5140708 14050517 9140750 31 27  
0000000000000000 51832017 21032020 1801110 5140708 14050517 9140750 31 2700000000  
0000000000 51821501 15200902 18 0 5140708 14050517 9140750 31 27000000000000000000  
51812100 14202112 0 9140614 18130119 91513 0 20080514 18255131 27000000000000000000  
51812100 14202112 0 19091320 12012008 15145131 27000000000000000000 51812100 14202112  
0 19051418 9140750 31 27000000000000000000 51710514 7180115 8090300 110 9140614  
18130119 91513 0 19251919 513180 58717382 595131 27000000000000000000 51851801 1130  
16120113 14091406 5131 27000000000000000000 51820513 5230101 1204 0 5140517 7240  
19251919 5131950 31 27000000000000000000 51831511 1170 30512110 20050307 14151214  
7255131 27000000000000000000 51870913 3 0 5140517 7240 18051904 1180307 5131 2700000  
000000000000 51691404 180724 0 16151208 3240 11403 0 5031513 15130902 195131 2700000  
000000000000 51671512 16212000 20091513 1110 14052117 15190308 5140304 5131 27000000  
000000000000 51780520 18150208 15121506 255131 27000000000000000000 51671506  
14092008 2204 0 14052117 15190308 5140304 5131 27000000000000000000 51832518 20051318  
0 14052117 15190308 5140304 5131 27000000000000000000 51722112 1146017 15021519 0  
9142004 18010319 9151450 31 27000000000000000000 51692214 12212008 15140117 240  
16192502 8151214 7255131 27000000000000000000 51831502 901110 14052022 151810 0  
1140111 25190918 50 27000000000000 41 27 27 27 27000000000000 0 40505 0 19172100  
18056115 18091419 61201515 9036108 14040902 5195817 61220111 210531 0 7612200 12210531 0  
2612200 12210531 0 6091204 5937 27000000000000000000 0 9140404 2491 0 620 18612200  
122104 0 53 0 12051457 9140404 24619258 27000000000000000000 0 9140404 2492 0 62 0 7612200  
122104 0 53 0 12051457 9140404 24619258 27000000000000000000 0 9140404 2493 0 62 0 2612200  
122104 0 53 0 12051457 9140404 24619258 27000000000000000000 0 464645 270000000000  
000000 6091204 64231808 20055805 51302029 20302043 20151608 3194308 14040523 92424529  
145158 2700000000000000 6091204 64231808 20055805 51302029 20302043 20151608  
3194308 14040523 93424529 145158 2700000000000000 6091204 64231808 20055805  
51302029 20302043 20151608 3194308 14040523 94424529 145158 2700000000000000  
464645 2700000000000000 6091204 64231808 20055805 51302029 20302043 9140404  
24619242 9140404 24924244 30145158 2700000000000000 6091204 64231808 20055805  
51302029 20302043 9140404 24619242 9140404 24934244 30145158 2700000000000000































2014 0 200804 0 6091204 645158 27 0 0 0 0 9065804 14201800 140304 0 6362 0 92935937 27 0 0 0 0 0  
0 0 13010913 915858 27 0 0 0 0 9065804 14201800 140304 0 6362 0 92945937 27 0 0 0 0 0 0 0 0 3152113  
20010211 5195858 27 0 0 0 0 9065804 14201800 140304 0 6362 0 92955937 27 0 0 0 0 0 0 0 0 39 0  
83160502 90624 0 200804 0 6091204 0 16012007 0 61517 0 200804 0 93976001 919 0 3151301 9140119  
9151418 27 0 0 0 0 0 0 0 14011304 61052419 0 62 0 9141620 20585168 14200517 0 6091204 14011304  
0 23092007 0 64202419 0 5242004 14190914 1437 0 5158 27 0 0 0 0 0 0 0 6091204 61160119 8619396  
61020919 0 62 0 51121200 14072100 7057714 4330208 14011824 60031513 6090718 3350 0 61 0  
14011304 61052419 27 0 0 0 0 0 0 0 1460 0 62 0 9142057 9141620 20585168 14200517 0 20919 0  
12051406 2007 0 160517 0 2091400 1824 0 19201808 140737 0 515958 27 27 0 0 0 0 0 0 0 39 0  
71051404 18012004 0 11403 0 19012204 0 200804 0 93976001 919 0 3151301 9140119 9151418 27 0 0 0  
0 0 0 0 0 7051404 18012004 61011403 61190121 5611460 2092060 3151301 9140119 9151418 58060911  
5611600 20086192 97610208 2031 0 146158 27 0 0 0 0 9065804 14201800 140304 0 6362 0 92965937 27  
0 0 0 0 0 0 0 0 8051215 61161808 14205858 27 0 0 0 0 9065804 14201800 140304 0 6362 0 92975937 27  
0 0 0 27 0 0 0 0 0 0 0 0 2011904 0 62 0 9142057 9141620 20585168 14200517 0 200804 0 9142004 70517  
0 2011904 0 1505 0 15180906 913 0 6091204 0 58020519 23050513 0 92 0 11403 0 92975937 0 515958 27  
27 0 0 0 0 0 0 0 9141620 20610608 1204 0 62 0 9141620 20585168 14200517 0 15180906 913 0 6091204  
0 16012007 37 0 5158 27 0 0 0 0 0 0 0 15212015 21206105 91204 0 62 0 9141620 20585168 14200517 0  
15212015 2119 0 6091204 0 16012007 37 0 5158 27 27 0 0 0 0 0 0 0 20182537 27 0 0 0 0 0 0 0 0 0 0 0  
3151421 5182060 6091204 58091415 21206105 9120531 0 15212015 21206105 9120531 0 2011904 58 27  
0 0 0 0 0 0 0 0 0 0 0 16180913 20580650 83210302 5191905 21121224 0 3151421 5182004 3 0 44091415  
21206105 9120544 0 2014 0 44152119 16212060 6091204 455158 27 0 0 0 0 0 0 0 5240304 1619 0  
70091204 78152069 15211403 69181814 1837 27 0 0 0 0 0 0 0 0 0 0 0 16180913 20580650 69181814  
1837 0 70091204 0 141519 0 6152113 463 0 80120500 1904 0 3080502 10 0 200804 0 16012007 18 0  
61517 0 44091415 21206105 9120544 0 11403 0 44152119 16212060 6091204 455158 27 0 0 0 0 27 0 0 0  
0 9065804 14201800 140304 0 6362 0 92985937 27 0 0 0 0 0 0 0 23080911 55895 0 34 0 935937 27 0 0 0  
0 0 0 0 0 0 0 0 9142024 0 62 0 9142057 9141620 20585190 0 61517 0 13010913 61130513 2131 0 91 0  
2014 0 3151419 9142104 37 0 515958 27 0 0 0 0 0 0 0 0 0 0 0 9065808 142024 0 6362 0 915937 27 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 2180500 10 27 0 0 0 0 0 0 0 0 0 0 0 0 9065808 142024 0 6362 0 925937 27 0 0 0 0  
0 0 0 0 0 0 0 0 0 0 0 13010913 935858 27 0 0 0 0 9065804 14201800 140304 0 6362 0 92995937 27 0 0 0  
0 0 0 0 23080911 55891 0 33 0 935937 27 0 0 0 0 0 0 0 0 0 0 0 9142019 0 62 0 9142057 9141620  
20585190 0 61517 0 13010913 61130513 2131 0 91 0 2014 0 3151419 9142104 37 0 515958 27 0 0 0 0 0  
0 0 0 0 0 9065808 142019 0 6362 0 915937 27 0 0 0 0 0 0 0 0 0 0 0 0 2180500 10 27 0 0 0 0 0 0 0  
0 0 0 0 9065808 142019 0 6362 0 925937 27 0 0 0 0 0 0 0 0 0 0 0 0 13010913 92995858 27 0 0 0  
9065804 14201800 140304 0 6362 0 93005937 27 0 0 0 0 0 0 0 16180913 20585150 51301429 14840804  
0 6151211 15230913 6 0 918 0 113 0 5240112 161204 0 13011615 91406 0 61517 0 200804 0 6151419 0  
9140404 24383013 27 0 0 0 0 0 0 0 90 0 0 27 0 0 0 0 0 0 0 0 0 91 0 1 27 0 0 0 0 0 0 0 0 92 0 2 27 0 0 0 0 0  
0 0 93 0 3 27 0 0 0 0 0 0 0 94 0 4 27 0 0 0 0 0 0 0 0 95 0 5 27 0 0 0 0 0 0 0 0 96 0 6 27 0 0 0 0 0 0 0 97 0 7  
27 0 0 0 0 0 0 0 98 0 8 27 0 0 0 0 0 0 0 0 99 0 9 27 0 0 0 0 0 0 0 0 9290 0 10 27 0 0 0 0 0 0 0 0 9291 0 11 27  
0 0 0 0 0 0 0 9292 0 12 27 0 0 0 0 0 0 0 0 9293 0 13 27 0 0 0 0 0 0 0 0 9294 0 14 27 0 0 0 0 0 0 0 0 9295 0  
15 27 0 0 0 0 0 0 0 9296 0 16 27 0 0 0 0 0 0 0 0 9297 0 17 27 0 0 0 0 0 0 0 0 9298 0 18 27 0 0 0 0 0 0 0  
9299 0 19 27 0 0 0 0 0 0 0 0 9390 0 20 27 0 0 0 0 0 0 0 0 9391 0 21 27 0 0 0 0 0 0 0 0 9392 0 22 27 0 0 0 0  
0 0 0 9393 0 23 27 0 0 0 0 0 0 0 0 9394 0 24 27 0 0 0 0 0 0 0 9395 0 25 27 0 0 0 0 0 0 0 0 9396 0 45 27 0 0  
0 0 0 0 0 9397 0 31 27 0 0 0 0 0 0 0 0 9398 0 32 27 0 0 0 0 0 0 0 9399 0 33 27 0 0 0 0 0 0 0 0 9490 0 34  
27 0 0 0 0 0 0 0 9491 0 35 27 0 0 0 0 0 0 0 0 9492 0 36 27 0 0 0 0 0 0 0 9493 0 37 27 0 0 0 0 0 0 0 0 9494  
0 38 27 0 0 0 0 0 0 0 9495 0 39 27 0 0 0 0 0 0 0 9496 0 40 27 0 0 0 0 0 0 0 9497 0 41 27 0 0 0 0 0 0 0  
9498 0 42 27 0 0 0 0 0 0 0 9499 0 43 27 0 0 0 0 0 0 0 9590 0 44 27 0 0 0 0 0 0 0 9591 0 45 27 0 0 0 0  
0 0 0 9592 0 0 46 27 0 0 0 0 0 0 0 9593 0 47 27 0 0 0 0 0 0 0 9594 0 48 27 0 0 0 0 0 0 0 9595 0 49 27  
0 0 0 0 0 0 0 9596 0 50 27 0 0 0 0 0 0 0 9597 0 0 51 27 0 0 0 0 0 0 0 9598 0 52 27 0 0 0 0 0 0 0 9599 0  
53 27 0 0 0 0 0 0 0 9690 0 54 27 0 0 0 0 0 0 0 9691 0 55 27 0 0 0 0 0 0 0 9692 0 56 27 0 0 0 0 0 0 0  
9693 0 57 27 0 0 0 0 0 0 0 9694 0 58 27 0 0 0 0 0 0 0 9695 0 59 27 0 0 0 0 0 0 0 9696 0 60 27 0 0 0 0



















































1406 0 200804 0 9130106 518 0 7051404 18012004 3 0 224 0 200804 0 73130106 4 0 71051404 18012014  
1863 27 27 404039 0 9463 0 67151912 15121506 24 27 57576804 19031808 16200914 14575737 0  
84151511 18 0 11403 0 6211402 20091513 1120919 90518 0 61517 0 3151912 15121506 9030111 0  
18051904 1180307 0 11403 0 19091320 12012008 15141963 0 670113 0 204 0 21190503 0 913 0 3151409  
21140319 91513 0 23092007 0 79162008 151418 0 913 0 9231 0 11403 0 9363 27 27 404039 0 9563 0  
83051720 5142008 111 0 76011406 21010704 0 71051404 18012008 1513 27 57576804 19031808  
16200914 14575737 0 71051404 18012004 18 0 19051720 5140304 18 0 1505 0 12011406 21010704 63 0  
57578700 18140913 7575737 0 84080918 0 20151511 0 13211919 0 6091219 517 0 152119 0 16181505  
11404 0 3151419 5142063 0 64 0 21190517 0 0 64202419 0 6091204 0 13211919 0 5240918 19 0  
23092007 913 0 200804 0 23151810 91406 0 4091804 3201517 2531 0 12091919 91406 0 85140902  
150404 0 3080117 1032004 1818 0 58151404 0 160517 0 12091404 5963 27 27 404039 0 9663 0 84052419  
0 69040919 1517 27 57576804 19031808 16200914 14575737 0 64 0 2210911 20600913 0 20052419 0  
5040919 1517 0 61517 0 23180919 91406 0 11403 0 5040919 91406 0 20052419 0 6091204 1963 27 27  
404039 0 9763 0 71051404 18012004 0 67151312 11403 0 77012017 923 0 80122106 913 0 61517 0  
200804 0 80181506 18011312 91406 0 69140708 1404 27 57576804 19031808 16200914 14575737 0  
67180500 200518 0 3151312 11403 0 13012017 923 0 16122106 91418 0 2014 0 5242004 1403 0 200804 0  
3011600 2091208 20090518 0 1505 0 200804 0 80181506 18011312 91406 0 69140708 140563 27 27  
404039 0 9863 0 69240919 0 80181506 180112 27 57576804 19031808 16200914 14575737 0 69240919  
18 0 200804 0 65130111 502 0 1161611 9030119 9151463 27 27 404039 0 9963 0 9467 0 676567 0 676576  
0 69140708 1404 27 57576804 19031808 16200914 14575737 0 64 0 9467 0 67656832 676576 0 5140708  
1404 0 61517 0 3151315 21200517 60010903 503 0 4051908 713 0 11403 0 13011420 6010319 21180913  
763 27 27 404039 0 10063 0 81210113 202112 0 67091802 210919 0 81091910 919 0 87151810 66051402  
7 27 57576804 19031808 16200914 14575737 0 64 0 23151810 2051402 7 0 61517 0 3180500 20091406 0  
11403 0 19091320 12012008 1406 0 17210113 202112 0 3091802 21092018 0 21190913 6 0 81091910  
92063 27 27 404039 0 929163 0 67120118 19090300 11 0 67091802 210919 0 83030804 13681800 22 0  
87151810 66051402 7 27 57576804 19031808 16200914 14575737 0 64 0 23151810 2051402 7 0 61517 0  
4051908 7140913 6 0 11403 0 19091320 12012008 1406 0 3120118 19090300 11 0 3091802 21092018 0  
21190913 6 0 83030804 13681800 2363 27 27 404039 0 929263 0 67151912 15121506 9030111 0  
83091320 12012008 1513 0 224 0 84052419 27 57576804 19031808 16200914 14575737 0 83091320  
12012004 18 0 3151912 15121506 9030111 0 16080513 15130513 0 0 21190913 6 0 20052419 60020118  
503 0 9141620 2018 0 11403 0 15212015 21201963 27 27 404039 0 929363 0 76011806 4 0 76011406  
21010704 0 77150404 11 27 57576804 19031808 16200914 14575737 0 73142004 7180119 518 0 0 0  
12011806 4 0 12011406 21010704 0 13150404 11 0 61517 0 22011808 152118 0 12011406 21010704 0  
16181502 5191908 1406 0 20011910 1963 27 27 404039 0 929463 0 68012000 0 83031808 1619 0 224 0  
67211919 1512 0 83201820 3202117 518 27 57576804 19031808 16200914 14575737 0 71051404  
18012004 18 0 4012000 0 19031808 162018 0 21190913 6 0 3211919 1512 0 4012000 0 19201820  
3202117 51963 27 27 404039 0 929563 0 71051404 18012004 0 66091400 1824 0 85140916 2104 0  
83201808 140718 0 2014 0 0 3211919 15136419 2419 0 70091204 27 57576804 19031808 16200914  
14575737 0 67180500 200518 0 21140916 2104 0 2091400 1824 0 19201808 140718 0 11403 0 19012204  
18 0 20080512 0 2014 0 0 3211919 1512 0 0 64202419 0 0 6091204 63 27 27 404039 0 929763 0  
67151421 51819 0 64152119 1 0 70091204 0 2014 0 66091400 18257920 19 0 70091204 0 58692419  
5141908 1513 0 1505 0 79162008 1513 0 929358 27 57576804 19031808 16200914 14575737 0 67151421  
5182018 0 0 64152119 1 0 0 6091204 18 0 2014 0 0 2091400 18251520 19 0 0 6091204 1931 0 5242004  
14040913 6 0 200804 0 6211402 20091513 1120919 24 0 1505 0 200804 0 76011806 4 0 76011406  
21010704 0 77150404 1263 27 27 404039 0 929863 0 84052419 0 68012000 2011904 27 57576804  
19031808 16200914 14575737 0 77011400 70518 0 11403 0 13011408 16211200 200518 0 20052419 0  
4012000 2011904 1963 27 27 404039 0 929963 0 80181523 9130111 0 80180912 4 0 78211301 51818 27  
57576804 19031808 16200914 14575737 0 67011202 2112019 518 0 11403 0 1140111 25260518 0  
16181523 9130111 0 16180912 4 0 14211301 5181963 27 27 404039 0 930063 0 67151405 9072117  
1200914 1418 0 61517 0 77150404 0 92 27 57576804 19031808 16200914 14575737 0 80181521 9040518

0 3151405 9072117 1200914 13 0 19052019 9140718 0 61517 0 200804 0 80181506 18011312 91406 0  
69140708 1404 0 58791619 91513 0 935963 27 27 404039 0 939263 0 80181523 9130111 0 80180912 518  
0 65140912 1201517 27 57576804 19031808 16200914 14575737 0 65140912 1200518 0 200804 0  
2050800 22091517 0 11403 0 4091919 18090220 20091513 0 1505 0 16181523 9130111 0 16180912 4 0  
14211301 5181963 27 27 404039 0 939363 0 77011420 111 0 84211808 1406 0 67151315 12052004 0  
70091408 2004 0 83200119 4 0 77010307 91404 27 57576804 19031808 16200914 14575737 0 65121214  
2318 0 200804 0 21190517 0 2014 0 13011420 1121224 0 3151418 20182102 19 0 0 0 84211808 14076002  
15131611 52004 0 6091408 2004 0 19200119 4 0 13010307 9140563 27 27 404039 0 939463 0 65212014  
13012008 2 0 84211808 1406 0 67151315 12052004 0 70091408 2004 0 83200119 4 0 77010307 91404 27  
57576804 19031808 16200914 14575737 0 65212014 13012008 3011211 24 0 7051404 18012004 18 0 0  
84211808 14076002 15131611 52004 0 6091408 2004 0 19200119 4 0 13010307 9140563 27 27 404039 0  
939563 0 80252007 1513 0 69140101 120503 0 67151312 11403 0 76091404 0 73142004 18060102 4 27  
57576804 19031808 16200914 14575737 0 80181521 9040518 0 0 0 3151312 11403 0 12091404 0  
9142004 18060102 4 0 23092007 0 80252007 1513 0 3011600 2091208 20090518 63 27 27 404039 0  
939663 0 65212014 13012008 2 0 67151419 181511 0 69140708 14050517 91406 27 57576804 19031808  
16200914 14575737 0 84151511 18 0 11403 0 6211402 20091513 1120919 90518 0 61517 0 1212014  
13012008 2 0 3151419 181511 0 5140708 14050517 9140763 27 27 404039 0 939763 0 65212014  
13012004 3 0 68051908 713 0 11403 0 84050307 14151214 724 27 57576804 19031808 16200914  
14575737 0 70010308 12092000 200518 0 1212014 13012004 3 0 4051908 713 0 11403 0 20050307  
14151214 7090300 11 0 4052204 12151612 51419 0 20011910 1963 27 27 404039 0 939863 0 67151418  
20182102 19 0 89152117 0 792313 0 76091406 21091919 902 0 76011406 21010704 27 57576804  
19031808 16200914 14575737 0 69140101 120518 0 21190517 18 0 2014 0 3151418 20182102 19 0  
20080508 17 0 152313 0 12091406 21091919 902 0 12011406 21010704 1963 27 27 404039 0 939963 0  
67151421 51819 0 84052419 0 69140314 4091406 0 2014 0 0 0 67151421 5180704 1419 0 73130106 4 0  
58701814 12 0 79162008 1513 0 929358 27 57576804 19031808 16200914 14575737 0 67151421  
5182018 0 20052419 0 5140314 4091406 31 0 4051808 220503 0 6181512 0 200804 0 76011806 4 0  
76011406 21010704 0 77150404 1231 0 9142014 0 0 0 3151421 5180704 1419 0 9130106 563 27 27  
404039 0 940063 0 77012007 5130119 90318 0 760101 27 57576804 19031808 16200914 14575737 0 64 0  
120101 0 5142208 18151412 51419 0 61517 0 3151403 21032008 1406 0 13012007 5130119 9030111 0  
5241604 18091304 142018 0 11403 0 22091920 1120925 1200914 141963 27 27 4039 0 73141919  
1121200 20091513 27 9263 0 57576711 151404 0 200804 0 18051614 19092014 18255756 37 27 0 0 0 0 0  
0 1907 27 0 0 0 70919 0 3121513 4 0 8202015 19383332 7092007 21026402 15133367 79777364  
88697167 69330315 25640708 19 27 9363 0 57577800 22090700 2004 0 2014 0 200804 0 4091804  
3201517 24 0 913 0 20051812 9140111 0 1517 0 3151418 15120563 5756 27 9463 0 57571608 15 0  
9141919 11211 0 6017 0 18051720 9180512 5142018 64202419 5756 27 27 4039 0 76090304 141904 27  
84080918 0 16181509 50319 0 918 0 12090304 14190503 0 21140404 17 0 200804 0 777383 0 76090304  
141904 0 59 0 190504 0 200804 0 43767366 69788368 42587672 67697882 6958 0 6091204 0 61517 0  
4052000 9121963 27 27 19052020 16641624 0 47 27 27 6181512 0 19052020 16201514 1218 0 9131614  
1819 0 19052020 1631 0 6091403 61160102 11010704 18 27 27 19052020 1657 27 0 0 0 0 14011304  
63516512 1120502 5131 27 0 0 0 22051818 9151462 51916491 64915131 27 0 0 0 0 16010310 1070518  
63060913 4611600 3110106 5195858 31 27 0 0 0 0 0 9141919 1121260 18051720 9180518 6342 27 0 0 0 0  
0 0 0 51020500 21200905 21121914 21169550 31 27 0 0 0 0 0 0 0 51110524 2150117 45131 27 0 0 0 0  
0 0 0 51130119 16121519 12090250 31 27 0 0 0 0 0 0 0 51142112 16255131 27 0 0 0 0 0 0 0 51162506  
1130550 31 27 0 0 0 0 0 0 0 51800911 12152350 31 27 0 0 0 0 0 0 0 51161214 20122550 31 27 0 0 0 0  
0 0 0 51161920 20091250 31 27 0 0 0 0 0 0 0 51162500 21201506 21095131 27 0 0 0 0 0 0 0 51802580  
20965131 27 0 0 0 0 0 0 0 51170918 11092050 31 27 0 0 0 0 0 0 0 51180516 21051919 195131 27 0 0 0  
0 0 0 0 51190307 5130417 1235131 27 0 0 0 0 0 0 0 51190307 5042111 55131 27 0 0 0 0 4231 27 0 0 0  
0 5142017 25611614 9142018 6343 27 0 0 0 0 0 0 0 46031513 19151204 61190317 9162018 4637 0 42  
27 0 0 0 0 0 0 0 0 0 0 0 46172100 14202112 61011808 20081304 20090362 67381300 9144631 27 0 0 0 0  
0 0 0 0 4231 27 0 0 0 0 4531 27 0 0 0 0 1212007 15186350 771863 0 68151308 140902 0 65120523

1140404 17 0 67151515 5185131 27 0 0 0 1212007 15186104 13010911 63510400 3070404 64031514  
16051838 7130108 12640314 135131 27 0 0 0 4051902 18091619 9151462 5164 0 80252007 1513 0  
1161611 9030119 91513 0 61517 0 7051404 18012008 1406 0 17210113 202112 0 3091802 21092018 0  
11403 0 16051805 15181308 1406 0 1180919 8130519 902 0 7051512 5201824 0 21190913 6 0 19200119  
4 0 19080115 51931 0 11403 0 13210307 0 13151804 645131 27 0 0 0 12090304 14190562 51777383  
5131 27 0 0 0 11052522 15180418 63511720 1142020 12 0 3091802 21092018 0 1180919 8130519 902 0  
7051512 5201824 0 9130106 4 0 7051404 18012008 1513 0 3151912 15121506 24 0 12011406 21010704 0  
7051404 18012008 15145131 27 0 0 0 21181262 51082019 16193832 33070919 8210263 3151332  
4151308 1240506 4053302 16255131 27 58 27 27 18051720 9180512 5142018 64202419 0 47 27 27  
2050120 20090620 12191520 1694 27 11052501 15011803 27 13012015 12152011 901 27 14211315 24  
27 16250700 1304 27 80091211 1522 27 16121519 1224 27 16192119 911 27 16250120 20150720 8 27  
80258119 95 27 17091910 919 27 18051720 5192018 27 19030804 13041800 22 27 19030804 4211204 27  
27 67151517 4091400 20051960 80252007 15146419 2419 0 47 27 27 92329131 90 27 93329131 91 27  
94329131 92 27 95329131 93 27 96329131 94 27 97329131 95 27 98329131 96 27 99329131 97 27  
100329131 98 27 92913290 3299 27 92923290 329290 27 92933290 329291 27 92943290 329292 27  
92953290 329293 27 92963290 329294 27 92973290 329295 27 92983291 3290 27 92993291 3291 27  
93003291 3292 27 93913291 3293 27 93923291 3294 27 93933291 3295 27 93943291 3296 27 93953291  
3297 27 93963291 3298 27 93973291 3299 27 93983291 329290 27 93993291 329291 27 94003291  
329292 27 94913291 329293 27 94923291 329294 27 94933291 329295 27 94943292 3290 27 94953292  
3291 27 94963292 3292 27 94973292 3293 27 94983292 3294 27 94993292 3295 27 95003292 3296 27  
95913292 3297 27 95923292 3298 27 95933292 3299 27 95943292 329290 27 95953292 329291 27  
95963292 329292 27 95973292 329293 27 95983292 329294 27 95993292 329295 27 96003293 3290 27  
96913293 3291 27 96923293 3292 27 96933293 3293 27 96943293 3294 27 96953293 3295 27 96963293  
3296 27 96973293 3297 27 96983293 3298 27 96993293 3299 27 97003293 329290 27 97913293 329291  
27 97923293 329292 27 97933293 329293 27 97943293 329294 27 97953293 329295 27 97963294 3290  
27 97973294 3291 27 97983294 3292 27 97993294 3293 27 98003294 3294 27 98913294 3295 27  
98923294 3296 27 98933294 3297 27 98943294 3298 27 98953294 3299 27 98963294 329290 27  
98973294 329291 27 98983294 329292 27 98993294 329293 27 99003294 329294 27 99913294 329295  
27 99923295 3290 27 99933295 3291 27 99943295 3292 27 99953295 3293 27 99963295 3294 27  
99973295 3295 27 99983295 3296 27 99993295 3297 27 100003295 3298 27 100913295 3299 27  
100923295 329290 27 100933295 329291 27 100943295 329292 27 100953295 329293 27 100963295  
329294 27 100973295 329295 27 100983296 3290 27 100993296 3291 27 101003296 3292 27 92919131  
973293 27 92919231 973294 27 92919331 973295 27 92919431 973296 27 92919531 973297 27  
92919631 973298 27 92919731 973299 27 92919831 97329290 27 92919931 97329291 27 92920031  
97329292 27 92929131 97329293 27 92929231 97329294 27 92929331 97329295 27 92929431 983290 27  
92929531 983291 27 92929631 983292 27 92929731 983293 27 92929831 983294 27 92929931 983295  
27 92930031 983296 27 92939131 983297 27 92939231 983298 27 92939331 983299 27 92939431  
98329290 27 92939531 98329291 27 92939631 98329292 27 92939731 98329293 27 92939831 98329294  
27 92939931 98329295 27 92940031 993290 27 92949131 993291 27 92949231 993292 27 92949331  
993293 27 92949431 993294 27 92949531 993295 27 92949631 993296 27 92949731 993297 27  
92949831 993298 27 92949931 993299 27 92950031 99329290 27 92959131 99329291 27 92959231  
99329292 27 92959331 99329293 27 92959431 99329294 27 92959531 99329295 27 92959631 1003290  
27 92959731 1003291 27 92959831 1003292 27 92959931 1003293 27 92960031 1003294 27 92969131  
1003295 27 92969231 1003296 27 92969331 1003297 27 92969431 1003298 27 92969531 1003299 27  
92969631 100329290 27 92969731 100329291 27 92969831 100329292 27 92969931 100329293 27  
92970031 100329294 27 92979131 100329295 27 92979231 92913290 27 92979331 92913291 27  
92979431 92913292 27 92979531 92913293 27 92979631 92913294 27 92979731 92913295 27 92979831  
92913296 27 92979931 92913297 27 92980031 92913298 27 92989131 92913299 27 92989231 92913291  
90 27 92989331 92913291 91 27 92989431 92913291 92 27 92989531 92913291 93 27 92989631  
92913291 94 27 92989731 92913291 95 27 92989831 92923290 27 92989931 92923291 27 92990031

92923292 27 92999131 92923293 27 92999231 92923294 27 92999331 92923295 27 92999431 92923296  
27 92999531 92923297 27 92999631 92923298 27 92999731 92923299 27 92999831 92923291 90 27  
92999931 92923291 91 27 93000031 92923291 92 27 93009131 92923291 93 27 93009231 92923291 94  
27 93009331 92923291 95 27 93009431 92933290 27 93009531 92933291 27 93009631 92933292 27  
93009731 92933293 27 93009831 92933294 27 93009931 92933295 27 93010031 92933296 27 93919131  
92933297 27 93919231 92933298 27 93919331 92933299 27 93919431 92933291 90 27 93919531  
92933291 91 27 93919631 92933291 92 27 93919731 92933291 93 27 93919831 92933291 94 27  
93919931 92933291 95 27 93920031 92943290 27 93929131 92943291 27 93929231 92943292 27  
93929331 92943293 27 93929431 92943294 27 93929531 92943295 27 93929631 92943296 27 93929731  
92943297 27 93929831 92943298 27 93929931 92943299 27 93930031 92943291 90 27 93939131  
92943291 91 27 93939231 92943291 92 27 93939331 92943291 93 27 93939431 92943291 94 27  
93939531 92943291 95 27 93939631 92953290 27 93939731 92953291 27 93939831 92953292 27  
93939931 92953293 27 93940031 92953294 27 93949131 92953295 27 93949231 92953296 27 93949331  
92953297 27 93949431 92953298 27 93949531 92953299 27 93949631 92953291 90 27 93949731  
92953291 91 27 93949831 92953291 92 27 93949931 92953291 93 27 93950031 92953291 94 27  
93959131 92953291 95 27 93959231 92963290 27 93959331 92963291 27 93959431 92963292 27  
93959531 92963293 27 93959631 92963294 27 93959731 92963295 27 93959831 92963296 27 93959931  
92963297 27 93960031 92963298 27 93969131 92963299 27 93969231 92963291 90 27 93969331  
92963291 91 27 93969431 92963291 92 27 93969531 92963291 93 27 93969631 92963291 94 27  
93969731 92963291 95 27 27 13010913 3161659 15212015 21206415 24 0 47 27 27 9131614 1819 0  
10191513 27 6181512 0 18051614 18201200 2641208 2641600 7051908 260518 0 9131614 1819 0  
12052019 517 27 6181512 0 18051614 18201200 2641208 2641919 25120518 0 9131614 1819 0 7052082  
1131611 5832024 12058307 5052031 0 80011800 7180115 8832024 1204 27 6181512 0 18051614  
18201200 2641611 1202515 2118 0 9131614 1819 0 83091315 12056814 3840512 16120119 531 0  
80011800 7180115 831 0 83160102 51831 0 80010704 66180500 10 27 27 40505 0 18050103 61101914  
14610608 12055805 9120513 1130558 37 27 0 0 0 0 0 23092007 0 15160513 58060911 5140112 531 0  
46184658 0 118 0 6091204 37 27 0 0 0 0 0 0 0 4012000 0 62 0 10191513 64121500 4580608 120558 27 0  
0 0 0 18052020 1813 0 4012000 27 27 40505 0 3180500 20056115 4065803 1200131 0 15212015  
21206105 9120513 1130558 37 27 0 0 0 0 41502 0 62 0 83091315 12056814 3840512 16120119 5581520  
20162119 61060911 5140112 531 0 16010704 19092604 63120519 20051858 27 0 0 0 0 19202511 518 0  
62 0 7052082 1131611 5832024 12058307 5052057 58 27 0 0 0 0 3211919 15136118 20251204 18 0 62 0  
43 27 0 0 0 0 0 0 0 51200919 12055137 0 80011800 7180115 8832024 120557 27 0 0 0 0 0 0 0 0 0 0 0 0  
46840919 12054631 27 0 0 0 0 0 0 0 0 0 0 0 16011804 14206318 20251204 19434683 9201204 464231  
27 0 0 0 0 0 0 0 0 0 0 0 0 6151419 83092604 63939531 27 0 0 0 0 0 0 0 0 0 0 0 0 0 19160102 5650619  
5186391 92 27 0 0 0 0 0 0 0 0 0 5931 27 0 0 0 0 0 0 0 0 0 51021503 255137 0 80011800 7180115 8832024  
120557 27 0 0 0 0 0 0 0 0 0 0 0 46661503 25840523 204631 27 0 0 0 0 0 0 0 0 0 0 0 0 0 16011804 14206318  
20251204 19434665 15042583 5242045 4231 27 0 0 0 0 0 0 0 0 0 0 6151419 83092604 63929331 27 0  
0 0 0 0 0 0 0 0 0 0 0 19160102 5650619 5186391 9331 27 0 0 0 0 0 0 0 0 0 0 0 12050103 9140762 9295 27  
0 0 0 0 0 0 0 0 0 0 0 5931 27 0 0 0 0 0 0 0 0 0 51192101 20092011 55137 0 80011800 7180115 8832024 120557 27  
0 0 0 0 0 0 0 0 0 0 0 46832101 20092011 54631 27 0 0 0 0 0 0 0 0 0 0 0 0 0 16011804 14206318 20251204  
19434683 9201204 464231 27 0 0 0 0 0 0 0 0 0 0 6151419 83092604 63929331 27 0 0 0 0 0 0 0 0 0 0 0 0 0  
19160102 5650619 5186396 27 0 0 0 0 0 0 0 0 0 0 0 58 27 0 0 0 0 44 27 27 0 0 0 0 19201517 24 0 62 0 4341 27  
27 0 0 0 0 61517 0 19201820 3202117 5610903 31 0 4052000 91218 0 913 0 4012000 64092004 13195858  
37 27 0 0 0 0 0 0 0 0 0 19201517 25640115 16051403 58800117 1071800 16085818 20182102 20211804  
61090431 0 3211919 15136118 20251204 19434619 9201204 46425958 27 0 0 0 0 0 0 0 0 19201517  
25640115 16051403 58831600 3051857 9231 0 92935958 27 0 0 0 0 0 0 0 0 19201517 25640115 16051403  
58800117 1071800 16085805 51840912 5192000 131637 0 44040519 1091218 43462008 13051919  
1131645 42455131 0 3211919 15136118 20251204 19434618 21022008 20120545 425958 27 0 0 0 0 0 0 0  
0 19201517 25640115 16051403 58831600 3051857 9231 0 92935958 27 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0  
16181515 5182008 518 0 62 0 4052000 9121942 51161814 16051819 9051950 41 27 0 0 0 0 0 0 0 0 0 61517



3151817 50319 0 16012007 0 2014 0 25152117 0 22031610 6 0 9141919 1121200 20091513 58 27 0 56 27  
0 56 0 83200515 0 9537 0 822113 0 200804 0 80181506 180112 27 0 56 0 59 0 65062004 17 0 19210302  
5191905 2111 0 3151315 9120119 9151431 0 182113 0 200804 0 5240502 21200101 120537 27 0 56 0 0 0  
64307700 20080512 1200902 9151463 52404 27 0 56 27 0 56 0 84181520 2120518 8151519 9140737 27 0  
56 0 59 0 69141920 1804 0 11211 0 16012007 18 0 11804 0 3151817 50319 0 11403 0 16150913 19 0 2014  
0 200804 0 1161617 15161808 12004 0 4091804 3201517 9051963 27 0 56 0 59 0 69141920 1804 0 11211  
0 4051604 14040513 3090518 0 11804 0 9141919 1121204 3 0 11403 0 1030304 19190901 120563 27 0 56  
0 59 0 67080502 10 0 61517 0 11424 0 3151315 9120119 91513 0 5181814 1818 0 11403 0 18051914  
122204 0 13091918 91406 0 9140311 210404 0 16012007 18 0 1517 0 12090217 1180904 1963 27 0 56 27  
0 56 0 69240112 16120537 27 0 56 0 34 0 13110408 17 0 3151405 906 27 0 56 0 34 0 64307700 20080512  
1200902 9151463 52404 27 0 56 0 27 0 56 0 891520 19081520 1203 0 141522 0 204 0 1021204 0 2014 0  
9142004 18010319 0 23092007 0 200804 0 13051420 0 11403 0 16051805 151812 0 22011808 152118 0  
15160517 1200914 1418 0 118 0 4051902 18090204 3 0 913 0 200804 0 16181506 18011363 27 0 56 27 0  
56 0 701517 0 6211819 80517 0 8051215 0 11403 0 4150320 13051419 1200914 1431 0 18050604 17 0  
2014 0 200804 0 15060608 3090111 0 4150320 13051419 1200914 13 0 1505 0 200804 0 74837977 0  
61517 0 77150404 1813 0 676261 0 12090217 11824 0 11403 0 22031610 763 27 0 5732 27 40091402  
12210404 0 34091518 20180500 1334 27 40091402 12210404 0 34192017 9140734 27 40091402  
12210404 0 34190519 34 27 40091402 12210404 0 34211414 18040517 5046112 11634 27 40091402  
12210404 0 34130512 15182534 27 40091402 12210404 0 34030817 15141534 27 40091402 12210404 0  
34191919 18050112 34 27 40091402 12210404 0 51683832 22031610 7133321 3161106 33091418  
20011211 5043323 97956022 9140414 23193308 14031220 4053313 12150812 1141432 10191513  
64081615 50 27 40091402 12210404 0 34060911 5192518 20051334 27 27 21190913 6 0 10191513 0 62 0  
14121507 13011413 38381018 151436 27 27 3332 0 64 0 6211402 20091513 0 2014 0 70519 0 200804 0  
3211817 51419 0 20091304 19200112 15 0 913 0 200804 0 4051908 180503 0 6151812 119 27 19200437  
38192017 91406 0 7052066 21181804 14208408 13051919 1131657 58 0 43 27 0 0 0 0 1212014 0 141522  
0 62 0 19200437 38030817 15141537 38192518 20051360 3121502 11383813 15235858 36 27 0 0 0 0  
1212014 0 9146119 9130506 19 0 62 0 19200437 38030817 15141537 38192518 20051360 3121502  
11383819 15612008 13056119 58141522 5936 27 27 0 0 0 0 19200437 38192017 9140718 20180500 12 0  
191936 27 0 0 0 0 1918 0 3433 0 19200437 38162119 61200912 5581919 4383806 13200912 5585608  
14612008 13056119 5931 0 51548959 54136053 4845471 38547737 54839050 5936 27 0 0 0 0 18052020  
1813 0 19196418 20185858 36 27 44 27 27 3332 0 66011904 0 3120118 18 0 61517 0 8082 0 58801814  
16051819 24 0 83052058 27 3120118 18 0 80181515 5182024 830519 0 43 27 16210211 90337 27 0 0 0 0  
22091819 210111 0 22150903 0 4091915 12012557 58 0 3151418 19 0 62 0 9136 27 0 0 0 0 22091819  
210111 0 10191513 0 20156109 19151457 58 0 3151418 19 0 62 0 9136 27 0 0 0 0 22091819 210111 0  
22150903 0 4051204 20056115 18151604 18202557 3151418 19 0 19200437 38192017 9140755 0  
11052558 0 62 0 9136 27 0 0 0 0 22091819 210111 0 41801814 16051819 25830519 5858 0 62 0 4050600  
21122036 27 4536 27 27 3332 0 6513 0 5240112 161204 0 1505 0 0 21190517 60040505 9140503 0  
16181515 5182024 0 190519 0 21190913 6 0 74837977 27 3120118 18 0 68251400 13090379 18151604  
18202582 519 0 37 0 16210211 902 0 80181515 5182024 830519 0 43 27 16180921 1200537 27 0 0 0 0  
10191513 0 16181515 5182008 51936 27 16210211 90337 27 0 0 0 0 68251400 13090379 18151604  
18202582 5205802 15141919 0 10191513 55 0 16181515 1958 0 37 0 16181515 5182008 5195815  
18151618 58 0 4444 27 27 0 0 0 0 22150903 0 4091915 12012557 58 0 3151418 19 0 15220517 18090404  
0 43 27 0 0 0 0 0 0 19200437 38031520 19 0 3433 0 16181515 5182008 5196403 21131657 9558 0  
3433 0 19200437 38051403 1236 27 0 0 0 44 27 27 0 0 0 0 10191513 0 20156109 19151457 58 0  
3151418 19 0 15220517 18090404 0 43 27 0 0 0 0 0 0 0 18052020 1813 0 16181515 5182008 51936 27 0  
0 0 44 27 27 0 0 0 0 22150903 0 4051204 20056115 18151604 18202557 3151418 19 0 19200437  
38192017 9140755 0 11052558 0 15220517 18090404 0 43 27 0 0 0 0 0 0 16181515 5182008 5196404  
18011904 58110524 5936 27 0 0 0 0 44 27 4536 27 27 3332 0 67151515 517 0 3120118 18 0 2014 0  
5140300 16192111 12004 0 200804 0 19201820 3202117 4 27 3120118 18 0 67151515 517 0 43 27

16180921 1200537 27 0 0 0 19200437 38192017 91406 0 90436 27 0 0 0 0 19200437 38211408  
17210560 16201833 80181515 5182024 83052034 0 16181515 5182024 83052036 27 0 0 0 0 19200437  
38192017 91406 0 20091304 19200112 1636 27 27 0 0 0 0 19200119 902 0 19200437 38190519 34192003  
38381919 18091406 34 0 9048304 2036 27 27 16210211 90337 27 0 0 0 0 67151515 5185818 20043837  
19201808 1406 0 90431 0 19200437 38211408 17210560 16201833 80181515 5182024 83052034 0  
161958 27 0 0 0 0 0 0 0 37 0 9045818 20043837 13152204 58090458 5931 0 16181515 5182024  
83052057 19200437 38131521 5581618 595931 0 20091304 19200112 16580704 20672117 18051419  
84091304 19200112 16585958 0 43 27 0 0 0 0 0 0 0 905 0 58090482 5206405 9140457 20080918  
60350903 58 0 5062 0 9048304 20640513 4585958 0 43 27 0 0 0 0 0 0 0 0 0 0 0 20081814 22 0 19200437  
38091421 1120903 61011806 21130513 20585172 67 0 1121804 10424 0 5240918 20196450 5936 27 0 0  
0 0 0 0 0 44 27 0 0 0 0 0 0 0 9048304 20640913 19051819 58200808 19603508 45936 27 0 0 0 0 44 27 27  
0 0 0 0 3151418 19 0 19200437 38192017 9140755 0 7052072 45858 0 3151418 19 0 43 0 18052020 1813  
0 90436 0 44 27 0 0 0 0 3151418 19 0 19200437 38192017 9140755 0 7052083 9130518 20011315 5858 0  
3151418 19 0 43 0 18052020 1813 0 20091304 19200112 1636 0 44 27 0 0 0 0 3151418 19 0 80181515  
5182024 83052056 0 7052079 18151604 18202582 5205858 0 3151418 19 0 43 0 18052020 1813 0  
16181515 5182024 83052063 7052057 5936 0 44 27 27 0 0 0 0 22150903 0 4091915 12012557 58 0  
3151418 19 0 43 27 0 0 0 0 0 0 0 19200437 38031520 19 0 3433 0 51736837 0 50 0 3433 0 903 0 3433 0  
19200437 38051403 1236 27 0 0 0 0 0 0 0 19200437 38031520 19 0 3433 0 51840912 5192000 131637 0  
50 0 3433 0 20091304 19200112 15 0 3433 0 19200437 38051403 1236 27 0 0 0 0 0 0 0 19200437  
38031520 19 0 3433 0 51801814 16051819 9051937 0 5136 27 0 0 0 0 0 0 0 16181515 5182024  
83052059 35040918 16120124 585936 27 0 0 0 0 44 27 27 0 0 0 19200119 902 0 22150903 0 18051904  
20730482 5205858 0 43 27 0 0 0 0 0 0 9048304 20640311 5011857 5936 27 0 0 0 0 44 27 27 0 0 0  
19200119 902 0 22150903 0 21160400 20057303 83052057 3151418 19 0 19200437 38190519 34192003  
38381919 18091406 3555 0 14052372 4830519 58 0 43 27 0 0 0 0 0 0 0 9048304 19 0 62 0 14052372  
4830519 36 27 0 0 0 0 44 27 27 0 0 0 0 22150903 0 19012204 58031513 1919 0 19200437 38192017  
9140755 0 6091204 14011304 58 0 3151418 19 0 43 27 0 0 0 0 0 0 0 10191513 0 1036 27 0 0 0 0 0 0 0  
19200437 38090618 20180500 12 0 9147008 12055805 9120513 1130558 36 27 0 0 0 0 0 0 0 0 905 0  
58091469 9120563 9196114 16051457 5958 0 43 27 0 0 0 0 0 0 0 0 0 0 9147008 1204 0 3534 0 1036 27  
0 0 0 0 0 0 0 0 0 0 9147008 12056402 12151904 585936 27 0 0 0 0 0 0 0 0 44 27 0 0 0 0 0 0 0 27 0 0 0  
0 0 0 0 0 10430903 42435119 9130518 20011315 5141 0 62 0 20091304 19200112 1636 27 0 0 0 0 0 0 0  
10430903 42435115 18151604 18200904 195141 0 62 0 16181515 5182024 83052059 35201560  
10191513 585936 27 27 0 0 0 0 0 0 0 19200437 38150618 20180500 12 0 15212069 9120557 6091204  
14011304 5936 27 0 0 0 0 0 0 905 0 58152119 70091204 64091960 15160513 585958 0 43 27 0 0 0 0 0  
0 0 0 0 0 0 15212069 91204 0 3433 0 10640420 13165894 5936 27 0 0 0 0 0 0 0 44 27 0 0 0 0 44 27 27 0  
0 0 0 19200119 902 0 19200437 38211414 18040517 5046112 1163418 20043837 19201808 140731 0  
19200437 38211408 17210560 16201833 67151515 5183534 0 12150103 65121257 3151418 19 0  
19200437 38192017 9140755 0 6091204 14011304 58 0 43 27 0 0 0 0 0 0 0 19200437 38090618  
20180500 12 0 6091204 58060911 5140112 55936 27 0 0 0 0 0 0 0 905 0 58500608 12056408 19611515  
5145858 58 0 43 27 0 0 0 0 0 0 0 0 0 0 20081814 22 0 19200437 38182113 20091304 61051817  
15185850 67152111 3 0 141519 0 15160513 0 6091204 0 61517 0 18050103 9140750 5936 27 0 0 0 0 0 0  
0 44 27 0 0 0 0 0 0 0 10191513 0 1036 27 0 0 0 0 0 0 0 6091204 0 3534 0 1036 27 27 0 0 0 0 0 0  
19200437 38211414 18040517 5046112 1163418 20043837 19201808 140731 0 19200437 38211408  
17210560 16201833 67151515 5183534 0 3151515 5187700 1636 27 27 0 0 0 0 0 0 61517 0 58012119  
1555 0 43090431 0 4012000 41 0 37 0 10640919 5131957 5958 0 43 27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
38192017 91406 0 20091304 19200112 15 0 62 0 4012000 43512008 13051919 1131650 4236 27 0 0 0 0  
0 0 0 0 0 0 10191513 0 16181515 5182008 518 0 62 0 4012000 43511617 15160517 20090518 514236 27  
0 0 0 0 0 0 0 0 0 0 0 27 0 0 0 0 0 0 0 0 0 0 1212014 0 16181515 5182024 83051931 0 62 0 19200437  
38130110 5612113 9172104 34682513 1130902 80181515 5182024 83052034 58161814 16051819  
9051958 36 27 0 0 0 0 0 0 0 0 0 0 0 1212014 0 3151515 517 0 62 0 19200437 38130110 5612113  
9172104 34671514 16051834 58090431 0 19200437 38131521 5581617 15160517 20258304 20595936 27

0 0 0 0 0 0 0 0 0 0 0 0 3151515 5186034 20091304 19200112 15 0 62 0 20091304 19200112 1636 0 3332 0  
69141920 1804 0 200804 0 20091304 19200112 15 0 918 0 16180518 5182204 3 27 0 0 0 0 0 0 0 0 0 0 0 0  
27 0 0 0 0 0 0 0 0 0 0 3151515 5187700 16430903 41 0 62 0 19200437 38131521 5580314 15160517  
5936 27 0 0 0 0 0 0 0 44 27 27 0 0 0 0 0 0 0 0 18052020 1813 0 3151515 5187700 1636 27 0 0 0 0 44 27  
27 0 0 0 0 22150903 0 21160400 20058017 15160517 20258304 20580314 141919 0 10191513 55 0  
14052379 18151618 58 0 43 27 0 0 0 0 0 0 0 16181515 5182024 830519 0 62 0 19200437 38130110  
5612113 9172104 34682513 1130902 80181515 5182024 83052034 58140522 80181515 195936 27 0 0 0  
0 44 27 27 0 0 0 0 22150903 0 4051204 20058017 15160517 20255802 15141919 0 19200437 38192017  
9140755 0 11052558 0 43 27 0 0 0 0 0 0 0 16181515 5182024 83052059 35040511 5200560 16181515  
5182024 58110524 5936 27 0 0 0 0 44 27 27 0 0 0 0 22150903 0 3080113 7057303 58031513 1919 0  
19200437 38192017 9140755 0 14052372 458 0 43 27 0 0 0 0 0 0 0 905 0 58090482 5206405 9140457  
14052372 458 0 5062 0 9048304 20640513 4585958 0 43 27 0 0 0 0 0 0 0 0 0 0 0 0 0 20081814 22 0  
19200437 38091421 1120903 61011806 21130513 20585172 67 0 1121804 10424 0 5240918 20196450  
5936 27 0 0 0 0 0 0 44 27 0 0 0 0 0 0 9048304 20640517 1190557 9045936 27 0 0 0 0 0 0 0 0 903 0  
62 0 14052372 436 27 0 0 0 0 0 0 9048304 20640913 19051819 58090458 36 27 0 0 0 0 44 27 4536 27  
27 19200437 38190519 34192003 38381919 18091406 34 0 67151515 5183837 9048304 2036 27 27  
22150903 0 4091915 12012576 5142157 58 0 43 27 0 0 0 0 19200437 38031520 19 0 3433 0 51770513  
21383013 5136 27 0 0 0 0 19200437 38031520 19 0 3433 0 519263 0 67180500 2004 0 140522 0 19201820  
3202117 5301450 36 27 0 0 0 0 19200437 38031520 19 0 3433 0 519363 0 68091915 120124 0 19201820  
3202117 5301450 36 27 0 0 0 0 19200437 38031520 19 0 3433 0 519463 0 85160400 2004 0 19201820  
3202117 4 0 16181515 5182008 5193013 5136 27 0 0 0 0 19200437 38031520 19 0 3433 0 519563 0  
68051204 2004 0 19201820 3202117 5301450 36 27 0 0 0 0 19200437 38031520 19 0 3433 0 519663 0  
67080113 704 0 19201820 3202117 4 0 73683013 5136 27 0 0 0 0 19200437 38031520 19 0 3433 0 519763  
0 83012204 0 19201820 3202117 4 0 2014 0 6091204 30145136 27 0 0 0 0 19200437 38031520 19 0 3433  
0 519863 0 76150103 0 19201820 3202117 4 0 6181512 0 6091204 30145136 27 0 0 0 0 19200437  
38031520 19 0 3433 0 519963 0 68051204 2004 0 0 0 16181515 5182024 0 6181512 0 19201820 3202117  
5301450 36 27 0 0 0 0 19200437 38031520 19 0 3433 0 520063 0 72051215 0 4150320 13051419 1200914  
14301450 36 27 0 0 0 0 19200437 38031520 19 0 3433 0 51929163 0 69240919 30145136 27 44 27 27  
22150903 0 8051215 68150320 13051419 1200914 145858 0 43 27 0 0 0 0 19200437 38031520 19 0 3433  
0 51720511 15 0 68150320 13051419 1200914 14383013 5136 27 0 0 0 0 19200437 38031520 19 0 3433 0  
519263 0 67180500 2004 0 140522 0 19201820 3202117 537 0 65121214 2318 0 251520 0 2014 0 3180500  
2004 0 0 0 140522 0 13012007 5130119 9030111 0 19201820 3202117 4 0 224 0 5142004 18091406 0  
16181515 5182008 518 0 9142004 18010319 9220511 25643013 5136 27 0 0 0 0 19200437 38031520 19 0  
3433 0 519363 0 68091915 120124 0 19201820 3202117 537 0 68091915 12012518 0 200804 0 4052000  
91218 0 1505 0 0 0 19051204 3200503 0 19201820 3202117 5643013 5136 27 0 0 0 0 19200437 38031520  
19 0 3433 0 519463 0 85160400 2004 0 19201820 3202117 4 0 16181515 5182008 51937 0 65121214 2318  
0 251520 0 2014 0 21160400 2004 0 200804 0 16181515 5182008 518 0 1505 0 113 0 5240918 20091406 0  
19201820 3202117 5643013 5136 27 0 0 0 19200437 38031520 19 0 3433 0 519563 0 68051204 2004 0  
19201820 3202117 537 0 68051204 200518 0 113 0 5240918 20091406 0 19201820 3202117 4 0 6181512  
0 200804 0 19251919 5136429 145136 27 0 0 0 19200437 38031520 19 0 3433 0 519663 0 67080113 704  
0 19201820 3202117 4 0 736837 0 67080113 70518 0 200804 0 7367 0 1505 0 113 0 5240918 20091406 0  
19201820 3202117 531 0 5141920 18091406 0 1414 0 4211611 9030119 5196429 145136 27 0 0 0 0  
19200437 38031520 19 0 3433 0 519763 0 83012204 0 19201820 3202117 4 0 2014 0 6091204 37 0  
83012204 18 0 200804 0 4052000 91218 0 1505 0 0 19051204 3200503 0 19201820 3202117 4 0 2014 0  
0 74837977 0 6091204 64301450 36 27 0 0 0 0 19200437 38031520 19 0 3433 0 519863 0 76150103 0  
19201820 3202117 4 0 6181512 0 6091204 37 0 76150103 18 0 0 0 19201820 3202117 4 0 6181512 0 0 0  
74837977 0 6091204 0 11403 0 1040418 0 919 0 2014 0 200804 0 19251919 5136429 145136 27 0 0 0 0  
19200437 38031520 19 0 3433 0 519963 0 68051204 2004 0 0 0 16181515 5182024 0 6181512 0 19201820  
3202117 537 0 68051204 200518 0 0 0 19160502 9060902 0 16181515 5182024 0 6181512 0 113 0  
5240918 20091406 0 19201820 3202117 5643013 5136 27 0 0 0 0 19200437 38031520 19 0 3433 0 520063







62 0 19120101 27 27 39 0 67180500 2004 0 200804 0 6091818 19 0 19121519 0 58151404 60200808 1803  
0 4152313 0 6181512 0 200804 0 20151631 0 19200117 20091406 0 6181512 0 200804 0 201515 0  
6010304 58 27 19121519 91 0 62 0 80011819 64130110 5661523 589631 0 979431 0 9431 0 70180504  
67656863 86050319 15185857 92949659 96593393 31 0 9131 0 96 0 59 0 945958 27 27 39 0 67180500  
2004 0 200804 0 19050314 1403 0 19121519 0 58202314 60200808 180418 0 4152313 0 6181512 0  
200804 0 20151631 0 19200117 20091406 0 6181512 0 200804 0 201515 0 6010304 58 27 19121519 92 0  
62 0 80011819 64130110 5661523 589631 0 979431 0 9431 0 70180504 67656863 86050319 15185892  
57589293 96609658 339431 0 9131 0 96 0 59 0 945958 27 27 39 0 672119 0 200804 0 19121519 18 0  
6181512 0 200804 0 19120101 27 3212091 0 62 0 19120101 64032119 58191214 209258 27 3212092 0 62  
0 3212091 64032119 58191214 209358 27 19120101 61150209 64830800 1604 0 62 0 3212092 27 27 39 0  
82050314 13162119 4 0 200804 0 4150320 13051419 27 4150363 18050314 13162119 55858 27 27  
67151315 21200517 641624 0 47 27 27 9131614 1819 0 13012015 12152011 9026415 25161214 19 0 118  
0 161219 27 9131614 1819 0 13012015 12152011 9026415 1200307 518 0 118 0 16012002 80518 27 27 39  
0 83120101 0 4091304 14190914 1418 27 19120101 61120513 72007 0 62 0 929495 27 19120101  
61230903 2007 0 62 0 9793 27 19120101 61200808 3111404 1918 0 62 0 96 27 27 39 0 83121519 0  
4091304 14190914 1418 27 19121519 61120513 72007 0 62 0 95 27 19121519 61230903 2007 0 62 0 9793  
27 19121519 61040515 2007 0 62 0 93 27 27 39 0 83121519 0 16151908 20091513 18 27 19121519  
926123 0 62 0 58191200 2611204 14072007 0 59 0 19121519 61120513 7200858 0 32 0 93 27 19121519  
936123 0 62 0 92 0 56 0 58191200 2611204 14072007 0 59 0 19121519 61120513 7200858 0 32 0 93 27 27  
39 0 67180500 2004 0 6090720 1804 0 11403 0 1240918 27 6090731 0 123 0 62 0 16122063 19210215  
12152018 58060906 19092604 63589290 31 0 975958 27 27 39 0 67180500 2004 0 200804 0 19120101 0  
118 0 0 0 18050319 1140711 4 27 19120101 0 62 0 16012002 8051963 82050319 1140711 5585890 31 0  
915931 0 19120101 61120513 7200831 0 19120101 61200808 3111404 191931 0 12091404 23090419  
8639231 0 5040704 3151214 18634601 12010310 4631 0 6010304 3151214 18634606 18012545 58 27  
1246400 4046115 1200307 58191200 258 27 27 39 0 67180500 2004 0 200804 0 6091818 19 0 19121519  
27 19121519 91 0 62 0 16012002 8051963 82050319 1140711 5585818 12152091 612431 0 19120101  
61200808 3111404 1918 0 59 0 19121519 61040515 20085931 0 19121519 61120513 7200831 0  
19121519 61040515 200831 0 12091404 23090419 8639231 0 5040704 3151214 18634601 12010310  
4631 0 6010304 3151214 18634622 8092004 4658 27 1246400 4046115 1200307 58191214 209258 27 27  
39 0 67180500 2004 0 200804 0 19050314 1403 0 19121519 27 19121519 92 0 62 0 16012002 8051963  
82050319 1140711 5585818 12152092 612431 0 19120101 61200808 3111404 1918 0 59 0 19121519  
61040515 20085931 0 19121519 61120513 7200831 0 19121519 61040515 200831 0 12091404 23090419  
8639231 0 5040704 3151214 18634601 12010310 4631 0 6010304 3151214 18634622 8092004 4658 27  
1246400 4046115 1200307 58191214 209358 27 27 39 0 830519 0 200804 0 12091308 2018 0 11403 0  
1191604 319 0 18012008 14 27 1246418 5206123 12091357 60929131 0 19120101 61120513 72007 0 61 0  
929158 27 1246418 5206124 12091357 60929131 0 19120101 61200808 3111404 1918 0 61 0 929158 27  
1246418 5206100 19160502 20584604 17210111 4658 27 27 39 0 650403 0 12010204 1218 0 11403 0  
20092011 4 27 1246418 5206123 12010204 12584675 5140719 7 0 58131358 4658 27 1246418 5206124  
12010204 12584683 8090310 14051918 0 58131358 4658 27 1246418 5206119 9201204 58468319  
1091411 51918 0 83200504 11 0 83120101 0 23092007 0 83121519 194658 27 27 39 0 83081522 0 200804  
0 16121519 27 16122063 7180903 58841820 558 27 16122063 19081522 5858 27 27 69142112 5180119  
15186415 24 0 47 27 27 9131614 1819 0 13012015 12152011 9026415 25161214 19 0 118 0 161219 27  
6181512 0 13161260 20151511 11092018 64131611 15209403 64011819 9403 0 9131614 1819 0  
80151224 94686714 12120502 20091513 27 9131614 1819 0 14211315 24 0 118 0 1415 27 27 39 0  
83120101 0 4091304 14190914 1418 27 19120101 61120513 72007 0 62 0 929495 27 19120101 61230903  
2007 0 62 0 9793 27 19120101 61200808 3111404 1918 0 62 0 96 27 27 39 0 83121519 0 4091304  
14190914 1418 27 19121519 61120513 72007 0 62 0 95 27 19121519 61230903 2007 0 62 0 9793 27  
19121519 61040515 2007 0 62 0 93 27 27 39 0 83121519 0 16151908 20091513 18 27 19121519 926123 0  
62 0 58191200 2611204 14072007 0 59 0 19121519 61120513 7200858 0 32 0 93 27 19121519 936123 0  
62 0 92 0 56 0 58191200 2611204 14072007 0 59 0 19121519 61120513 7200858 0 32 0 93 27 19121519

6124 0 62 0 90 27 19121519 6125 0 62 0 19120101 61200808 3111404 1918 0 59 0 19121519 61040515  
2007 27 27 39 0 67180500 2004 0 6090720 1804 27 60906 0 62 0 16122063 6090720 18055858 27 123 0  
62 0 6090763 1040460 19210215 12152057 92929231 0 16181509 5032008 15146345 94044658 27 27 39  
0 67180500 2004 0 22051819 9030518 0 1505 0 200804 0 19120101 27 22051819 9030518 0 62  
0 14166400 18180124 5842 27 0 0 0 439131 0 9131 0 914231 0 43191200 2611204 14072007 31 0 9131  
0 914231 0 43191200 2611204 14072007 31 0 19120101 61230903 200831 0 914231 0 439131 0 19120101  
61230903 200831 0 914231 0 39 0 66152019 1512 0 6010304 27 0 0 0 439131 0 9131 0 19120101  
61200808 3111404 19194231 0 43191200 2611204 14072007 31 0 9131 0 19120101 61200808 3111404  
19194231 0 43191200 2611204 14072007 31 0 19120101 61230903 200831 0 19120101 61200808  
3111404 19194231 0 439131 0 19120101 61230903 200831 0 19120101 61200808 3111404 191941 0 0 39  
0 841515 0 6010304 27 4258 27 27 39 0 68050608 1404 0 200804 0 96 0 6010304 18 0 1505 0 200804 0  
19120101 27 6010304 18 0 62 0 42 27 0 0 0 43220517 20090304 19439141 31 0 22051819 9030518  
43924231 0 22051819 9030518 43964231 0 22051819 9030518 43954241 31 0 39 0 70181513 19 0  
6010304 27 0 0 0 43220517 20090304 19439241 31 0 22051819 9030518 43934231 0 22051819 9030518  
43974231 0 22051819 9030518 43964241 31 0 39 0 82090707 19 0 6010304 27 0 0 0 43220517  
20090304 19439341 31 0 22051819 9030518 43944231 0 22051819 9030518 43984231 0 22051819  
9030518 43974241 31 0 39 0 66010310 0 6010304 27 0 0 0 43220517 20090304 19439441 31 0  
22051819 9030518 43914231 0 22051819 9030518 43954231 0 22051819 9030518 43984241 31 0 39 0  
76050619 0 6010304 27 0 0 0 43220517 20090304 19439141 31 0 22051819 9030518 43924231 0  
22051819 9030518 43934231 0 22051819 9030518 43944241 31 0 39 0 66152019 1512 0 6010304 27 0 0  
0 0 43220517 20090304 19439541 31 0 22051819 9030518 43964231 0 22051819 9030518 43974231 0  
22051819 9030518 43984241 31 0 39 0 841515 0 6010304 27 41 27 27 39 0 650403 0 200804 0 6010304  
18 0 1505 0 200804 0 19120101 0 2014 0 200804 0 16121519 27 1246400 4046102 15121204 3200914  
14940457 80151224 94686714 12120502 20091513 58060102 51931 0 6010304 3151214 18196345  
7180124 4631 0 12091404 23090419 8196391 31 0 5040704 3151214 18196345 2120102 114631 0  
1121607 1639163 965958 27 27 39 0 68050608 1404 0 200804 0 19121519 18 0 118 0 19130111 120517 0  
2152404 18 0 11403 0 10403 0 20080512 0 2014 0 200804 0 16121519 27 19121519 92612204 18200902  
518 0 62 0 14166400 18180124 5842 27 0 0 0 43191214 20926123 31 0 19121519 612531 0 19121519  
61264231 0 43191214 20926123 0 61 0 19121519 61120513 7200831 0 19121519 612531 0 19121519  
61264231 0 43191214 20926123 0 61 0 19121519 61120513 7200831 0 19121519 6124 0 61 0 19121519  
61230903 200831 0 19121519 61264231 0 43191214 20926123 31 0 19121519 6124 0 61 0 19121519  
61230903 200831 0 19121519 61264231 27 0 0 0 43191214 20926123 31 0 19121519 612531 0  
19120101 61200808 3111404 19194231 0 43191214 20926123 0 61 0 19121519 61120513 7200831 0  
19121519 612531 0 19120101 61200808 3111404 19194231 0 43191214 20926123 0 61 0 19121519  
61120513 7200831 0 19121519 6124 0 61 0 19121519 61230903 200831 0 19120101  
19194231 0 43191214 20926123 31 0 19121519 6124 0 61 0 19121519 61230903 200831 0 19120101  
61200808 3111404 191941 27 4258 27 27 19121519 93612204 18200902 518 0 62 0 14166400 18180124  
5842 27 0 0 0 43191214 20936123 31 0 19121519 612531 0 19121519 61264231 0 43191214 20936123 0  
61 0 19121519 61120513 7200831 0 19121519 612531 0 19121519 61264231 0 43191214 20936123 0 61 0  
19121519 61120513 7200831 0 19121519 6124 0 61 0 19121519 61230903 200831 0 19121519 61264231  
0 43191214 20936123 31 0 19121519 6124 0 61 0 19121519 61230903 200831 0 19121519 61264231 27 0  
0 0 0 43191214 20936123 31 0 19121519 612531 0 19120101 61200808 3111404 19194231 0 43191214  
20936123 0 61 0 19121519 61120513 7200831 0 19121519 612531 0 19120101 61200808 3111404  
19194231 0 43191214 20936123 0 61 0 19121519 61120513 7200831 0 19121519 6124 0 61 0 19121519  
61230903 200831 0 19120101 61200808 3111404 19194231 0 43191214 20936123 31 0 19121519 6124 0  
61 0 19121519 61230903 200831 0 19120101 61200808 3111404 191941 27 4258 27 27 19121519  
61060102 51991 0 62 0 42 27 0 0 0 43191214 20926121 5182008 3051942 914231 0 19121519 92612204  
18200902 5194391 4231 0 19121519 92612204 18200902 5194395 4231 0 19121519 92612204 18200902  
5194394 424231 27 0 0 0 43191214 20926121 5182008 3051942 924231 0 19121519 92612204  
18200902 5194392 4231 0 19121519 92612204 18200902 5194396 4231 0 19121519 92612204 18200902

5194395 424231 27 0 0 0 43191214 20926121 5182008 3051942 934231 0 19121519 92612204  
18200902 5194393 4231 0 19121519 92612204 18200902 5194397 4231 0 19121519 92612204 18200902  
5194396 424231 27 0 0 0 43191214 20926121 5182008 3051942 944231 0 19121519 92612204  
18200902 5194390 4231 0 19121519 92612204 18200902 5194394 4231 0 19121519 92612204 18200902  
5194397 424231 27 0 0 0 43191214 20926121 5182008 3051942 914231 0 19121519 92612204  
18200902 5194391 4231 0 19121519 92612204 18200902 5194392 4231 0 19121519 92612204 18200902  
5194393 424231 27 0 0 0 43191214 20926121 5182008 3051942 954231 0 19121519 92612204  
18200902 5194395 4231 0 19121519 92612204 18200902 5194396 4231 0 19121519 92612204 18200902  
5194397 4241 27 41 27 27 19121519 61060102 51992 0 62 0 42 27 0 0 0 0 43191214 20936121 5182008  
3051942 914231 0 19121519 93612204 18200902 5194391 4231 0 19121519 93612204 18200902  
5194395 4231 0 19121519 93612204 18200902 5194394 424231 27 0 0 0 0 43191214 20936121 5182008  
3051942 924231 0 19121519 93612204 18200902 5194392 4231 0 19121519 93612204 18200902  
5194396 4231 0 19121519 93612204 18200902 5194395 424231 27 0 0 0 0 43191214 20936121 5182008  
3051942 934231 0 19121519 93612204 18200902 5194393 4231 0 19121519 93612204 18200902  
5194397 4231 0 19121519 93612204 18200902 5194396 424231 27 0 0 0 0 43191214 20936121 5182008  
3051942 944231 0 19121519 93612204 18200902 5194390 4231 0 19121519 93612204 18200902  
5194394 4231 0 19121519 93612204 18200902 5194397 424231 27 0 0 0 0 43191214 20936121 5182008  
3051942 914231 0 19121519 93612204 18200902 5194391 4231 0 19121519 93612204 18200902  
5194392 4231 0 19121519 93612204 18200902 5194393 424231 27 0 0 0 0 43191214 20936121 5182008  
3051942 954231 0 19121519 93612204 18200902 5194395 4231 0 19121519 93612204 18200902  
5194396 4231 0 19121519 93612204 18200902 5194397 424231 27 41 27 27 1246400 4046102 15121204  
3200914 14940457 80151224 94686714 12120502 20091513 58191214 20610600 3051991 31 0 6010304  
3151214 18196345 23080919 54631 0 12091404 23090419 8196391 31 0 5040704 3151214 18196345  
2120102 11465958 27 1246400 4046102 15121204 3200914 14940457 80151224 94686714 12120502  
20091513 58191214 20610600 3051992 31 0 6010304 3151214 18196345 23080919 54631 0 12091404  
23090419 8196391 31 0 5040704 3151214 18196345 2120102 11465958 27 27 39 0 830519 0 200804 0  
12091308 2018 0 11403 0 12010204 1218 27 1246418 5206123 12091357 439131 0 19120101 61120513  
7200841 58 27 1246418 5206124 12091357 439131 0 19120101 61230903 20084258 27 1246418 5206125  
12091357 439131 0 19120101 61200808 3111404 19194258 27 1246418 5206123 12010204 12584675  
5140719 7 0 58131358 4658 27 1246418 5206124 12010204 12584686 9042007 0 58131358 4658 27  
1246418 5206125 12010204 12584683 8090310 14051918 0 58131358 4658 27 1246418 5206119  
9201204 58469467 0 86090522 0 1505 0 83200108 14120518 18 0 83200504 11 0 83120101 0 23092007 0  
83121519 194658 27 27 39 0 69172100 11 0 1191604 319 0 18012008 14 27 1246418 5206101 15246100  
19160502 20584318 12010260 12051406 200831 0 19120101 61230903 200831 0 19120101 61200808  
3111404 19194258 0 0 39 0 65191604 319 0 18012008 14 0 918 0 92389237 91 27 27 39 0 83081522 0  
200804 0 16121519 27 16122063 19081522 5858 27 27 76736782 69788368 0 47 27 27 777383 0  
76090304 141904 27 27 67151624 18090707 19 0 580358 0 93919394 0 68797772 65886970 6868 27 27  
80051812 9191908 1513 0 918 0 8051804 224 0 7180113 20050431 0 6180504 0 1505 0 3080117 70531 0  
2014 0 11424 0 16051818 1513 0 15022000 9140913 6 0 0 0 3151624 27 1505 0 20080918 0 19150619  
23011804 0 11403 0 1191914 3090119 503 0 4150320 13051419 12090114 13 0 6091204 18 0 58200804 0  
51831505 20230117 5515931 0 2014 0 4050111 27 913 0 200804 0 83150619 23011804 0 23092007  
152119 0 18051919 18090319 9151431 0 9140311 21040913 6 0 23092007 152119 0 12091308 20012008  
1513 0 200804 0 18090707 2018 27 2014 0 21190531 0 3151624 31 0 13150408 62531 0 13051806 531 0  
16210211 9190831 0 4091919 18090220 200531 0 19210211 9030513 190531 0 1140432 1517 0  
19051211 27 3151608 518 0 1505 0 200804 0 83150619 23011804 31 0 11403 0 2014 0 16051812 919 0  
16051818 151418 0 2014 0 23081512 0 200804 0 83150619 23011804 0 918 27 6211813 9190804 3 0 2014  
0 414 0 191531 0 19210209 50319 0 2014 0 200804 0 6151211 15230913 6 0 3151403 9200914 141937 27  
27 840804 0 1021521 4 0 3151624 18090707 19 0 14152008 304 0 11403 0 20080918 0 16051812 9191908  
1513 0 14152008 304 0 19080111 11 0 204 0 9140311 21040503 0 913 0 11211 27 3151608 518 0 1517 0  
19210218 20011419 90111 0 16151819 9151418 0 1505 0 200804 0 83150619 23011804 63 27 27 847268

0 83797083 87658268 0 7382 0 80827985 73686967 0 516582 0 73835131 0 87738471 798583 0 87658281  
65788488 0 7969 0 657888 0 75737867 31 0 69888081 698382 0 7981 27 73778075 73696831 0 73786775  
85687377 70 0 668583 0 787983 0 76737772 846967 0 8478 0 847268 0 87658281 65788472 6982 0 7969  
0 77698266 72657883 65667375 73848931 27 70738477 698382 0 707981 0 64 0 80658283 73678575  
6581 0 80858279 798368 0 657867 0 78797872 78708272 78716976 69788463 0 7377 0 7878 0 69866977  
83 0 83726575 75 0 847268 27 65858471 798282 0 7981 0 67798088 82737171 83 0 72797667 698282 0  
6668 0 76736565 7668 0 707981 0 657888 0 67766572 7731 0 68657764 716982 0 7981 0 79847268 81 27  
76736565 73767383 8931 0 87726983 726981 0 7377 0 6577 0 65678472 7977 0 7969 0 67797883  
82656783 31 0 84798283 0 7981 0 79847268 82877382 6931 0 65827382 737870 0 70827976 31 27  
798583 0 7969 0 7981 0 7377 0 67797877 69678472 7977 0 87738471 0 847268 0 83797083 87658268 0  
7981 0 847268 0 858368 0 7981 0 79847268 81 0 68696575 73787182 0 7377 0 847268 27 83797083  
87658268 63 27

|            |            |            |
|------------|------------|------------|
| 0000000000 | 0000010100 | 0000101000 |
| 0000000001 | 0000010101 | 0000101001 |
| 0000000010 | 0000010110 | 0000101010 |
| 0000000011 | 0000010111 | 0000101011 |
| 0000000100 | 0000011000 | 0000101100 |
| 0000000101 | 0000011001 | 0000101101 |
| 0000000110 | 0000011010 | 0000101110 |
| 0000000111 | 0000011011 | 0000101111 |
| 0000001000 | 0000011100 | 0000110000 |
| 0000001001 | 0000011101 | 0000110001 |
| 0000001010 | 0000011110 | 0000110010 |
| 0000001011 | 0000011111 | 0000110011 |
| 0000001100 | 0000100000 | 0000110100 |
| 0000001101 | 0000100001 | 0000110101 |
| 0000001110 | 0000100010 | 0000110110 |
| 0000001111 | 0000100011 | 0000110111 |
| 0000010000 | 0000100100 | 0000111000 |
| 0000010001 | 0000100101 | 0000111001 |
| 0000010010 | 0000100110 | 0000111010 |
| 0000010011 | 0000100111 | 0000111011 |

|            |            |            |
|------------|------------|------------|
| 0000111100 | 0001010000 | 0001100100 |
| 0000111101 | 0001010001 | 0001100101 |
| 0000111110 | 0001010010 | 0001100110 |
| 0000111111 | 0001010011 | 0001100111 |
| 0001000000 | 0001010100 | 0001101000 |
| 0001000001 | 0001010101 | 0001101001 |
| 0001000010 | 0001010110 | 0001101010 |
| 0001000011 | 0001010111 | 0001101011 |
| 0001000100 | 0001011000 | 0001101100 |
| 0001000101 | 0001011001 | 0001101101 |
| 0001000110 | 0001011010 | 0001101110 |
| 0001000111 | 0001011011 | 0001101111 |
| 0001001000 | 0001011100 | 0001110000 |
| 0001001001 | 0001011101 | 0001110001 |
| 0001001010 | 0001011110 | 0001110010 |
| 0001001011 | 0001011111 | 0001110011 |
| 0001001100 | 0001100000 | 0001110100 |
| 0001001101 | 0001100001 | 0001110101 |
| 0001001110 | 0001100010 | 0001110110 |
| 0001001111 | 0001100011 | 0001110111 |

|            |            |            |
|------------|------------|------------|
| 0001111000 | 0010001100 | 0010100000 |
| 0001111001 | 0010001101 | 0010100001 |
| 0001111010 | 0010001110 | 0010100010 |
| 0001111011 | 0010001111 | 0010100011 |
| 0001111100 | 0010010000 | 0010100100 |
| 0001111101 | 0010010001 | 0010100101 |
| 0001111110 | 0010010010 | 0010100110 |
| 0001111111 | 0010010011 | 0010100111 |
| 0010000000 | 0010010100 | 0010101000 |
| 0010000001 | 0010010101 | 0010101001 |
| 0010000010 | 0010010110 | 0010101010 |
| 0010000011 | 0010010111 | 0010101011 |
| 0010000100 | 0010011000 | 0010101100 |
| 0010000101 | 0010011001 | 0010101101 |
| 0010000110 | 0010011010 | 0010101110 |
| 0010000111 | 0010011011 | 0010101111 |
| 0010001000 | 0010011100 | 0010110000 |
| 0010001001 | 0010011101 | 0010110001 |
| 0010001010 | 0010011110 | 0010110010 |
| 0010001011 | 0010011111 | 0010110011 |

|            |            |            |
|------------|------------|------------|
| 0010110100 | 0011001000 | 0011011100 |
| 0010110101 | 0011001001 | 0011011101 |
| 0010110110 | 0011001010 | 0011011110 |
| 0010110111 | 0011001011 | 0011011111 |
| 0010111000 | 0011001100 | 0011100000 |
| 0010111001 | 0011001101 | 0011100001 |
| 0010111010 | 0011001110 | 0011100010 |
| 0010111011 | 0011001111 | 0011100011 |
| 0010111100 | 0011010000 | 0011100100 |
| 0010111101 | 0011010001 | 0011100101 |
| 0010111110 | 0011010010 | 0011100110 |
| 0010111111 | 0011010011 | 0011100111 |
| 0011000000 | 0011010100 | 0011101000 |
| 0011000001 | 0011010101 | 0011101001 |
| 0011000010 | 0011010110 | 0011101010 |
| 0011000011 | 0011010111 | 0011101011 |
| 0011000100 | 0011011000 | 0011101100 |
| 0011000101 | 0011011001 | 0011101101 |
| 0011000110 | 0011011010 | 0011101110 |
| 0011000111 | 0011011011 | 0011101111 |

|            |            |            |
|------------|------------|------------|
| 0011110000 | 0100000100 | 0100011000 |
| 0011110001 | 0100000101 | 0100011001 |
| 0011110010 | 0100000110 | 0100011010 |
| 0011110011 | 0100000111 | 0100011011 |
| 0011110100 | 0100001000 | 0100011100 |
| 0011110101 | 0100001001 | 0100011101 |
| 0011110110 | 0100001010 | 0100011110 |
| 0011110111 | 0100001011 | 0100011111 |
| 0011111000 | 0100001100 | 0100100000 |
| 0011111001 | 0100001101 | 0100100001 |
| 0011111010 | 0100001110 | 0100100010 |
| 0011111011 | 0100001111 | 0100100011 |
| 0011111100 | 0100010000 | 0100100100 |
| 0011111101 | 0100010001 | 0100100101 |
| 0011111110 | 0100010010 | 0100100110 |
| 0011111111 | 0100010011 | 0100100111 |
| 0100000000 | 0100010100 | 0100101000 |
| 0100000001 | 0100010101 | 0100101001 |
| 0100000010 | 0100010110 | 0100101010 |
| 0100000011 | 0100010111 | 0100101011 |

|            |            |            |
|------------|------------|------------|
| 0100101100 | 0101000000 | 0101010100 |
| 0100101101 | 0101000001 | 0101010101 |
| 0100101110 | 0101000010 | 0101010110 |
| 0100101111 | 0101000011 | 0101010111 |
| 0100110000 | 0101000100 | 0101011000 |
| 0100110001 | 0101000101 | 0101011001 |
| 0100110010 | 0101000110 | 0101011010 |
| 0100110011 | 0101000111 | 0101011011 |
| 0100110100 | 0101001000 | 0101011100 |
| 0100110101 | 0101001001 | 0101011101 |
| 0100110110 | 0101001010 | 0101011110 |
| 0100110111 | 0101001011 | 0101011111 |
| 0100111000 | 0101001100 | 0101100000 |
| 0100111001 | 0101001101 | 0101100001 |
| 0100111010 | 0101001110 | 0101100010 |
| 0100111011 | 0101001111 | 0101100011 |
| 0100111100 | 0101010000 | 0101100100 |
| 0100111101 | 0101010001 | 0101100101 |
| 0100111110 | 0101010010 | 0101100110 |
| 0100111111 | 0101010011 | 0101100111 |

|            |            |            |
|------------|------------|------------|
| 0101101000 | 0101111100 | 0110010000 |
| 0101101001 | 0101111101 | 0110010001 |
| 0101101010 | 0101111110 | 0110010010 |
| 0101101011 | 0101111111 | 0110010011 |
| 0101101100 | 0110000000 | 0110010100 |
| 0101101101 | 0110000001 | 0110010101 |
| 0101101110 | 0110000010 | 0110010110 |
| 0101101111 | 0110000011 | 0110010111 |
| 0101110000 | 0110000100 | 0110011000 |
| 0101110001 | 0110000101 | 0110011001 |
| 0101110010 | 0110000110 | 0110011010 |
| 0101110011 | 0110000111 | 0110011011 |
| 0101110100 | 0110001000 | 0110011100 |
| 0101110101 | 0110001001 | 0110011101 |
| 0101110110 | 0110001010 | 0110011110 |
| 0101110111 | 0110001011 | 0110011111 |
| 0101111000 | 0110001100 | 0110100000 |
| 0101111001 | 0110001101 | 0110100001 |
| 0101111010 | 0110001110 | 0110100010 |
| 0101111011 | 0110001111 | 0110100011 |

|            |            |            |
|------------|------------|------------|
| 0110100100 | 0110111000 | 0111001100 |
| 0110100101 | 0110111001 | 0111001101 |
| 0110100110 | 0110111010 | 0111001110 |
| 0110100111 | 0110111011 | 0111001111 |
| 0110101000 | 0110111100 | 0111010000 |
| 0110101001 | 0110111101 | 0111010001 |
| 0110101010 | 0110111110 | 0111010010 |
| 0110101011 | 0110111111 | 0111010011 |
| 0110101100 | 0111000000 | 0111010100 |
| 0110101101 | 0111000001 | 0111010101 |
| 0110101110 | 0111000010 | 0111010110 |
| 0110101111 | 0111000011 | 0111010111 |
| 0110110000 | 0111000100 | 0111011000 |
| 0110110001 | 0111000101 | 0111011001 |
| 0110110010 | 0111000110 | 0111011010 |
| 0110110011 | 0111000111 | 0111011011 |
| 0110110100 | 0111001000 | 0111011100 |
| 0110110101 | 0111001001 | 0111011101 |
| 0110110110 | 0111001010 | 0111011110 |
| 0110110111 | 0111001011 | 0111011111 |

|            |            |            |
|------------|------------|------------|
| 0111100000 | 0111110100 | 1000001000 |
| 0111100001 | 0111110101 | 1000001001 |
| 0111100010 | 0111110110 | 1000001010 |
| 0111100011 | 0111110111 | 1000001011 |
| 0111100100 | 0111111000 | 1000001100 |
| 0111100101 | 0111111001 | 1000001101 |
| 0111100110 | 0111111010 | 1000001110 |
| 0111100111 | 0111111011 | 1000001111 |
| 0111101000 | 0111111100 | 1000010000 |
| 0111101001 | 0111111101 | 1000010001 |
| 0111101010 | 0111111110 | 1000010010 |
| 0111101011 | 0111111111 | 1000010011 |
| 0111101100 | 1000000000 | 1000010100 |
| 0111101101 | 1000000001 | 1000010101 |
| 0111101110 | 1000000010 | 1000010110 |
| 0111101111 | 1000000011 | 1000010111 |
| 0111110000 | 1000000100 | 1000011000 |
| 0111110001 | 1000000101 | 1000011001 |
| 0111110010 | 1000000110 | 1000011010 |
| 0111110011 | 1000000111 | 1000011011 |

|            |            |            |
|------------|------------|------------|
| 1000011100 | 1000110000 | 1001000100 |
| 1000011101 | 1000110001 | 1001000101 |
| 1000011110 | 1000110010 | 1001000110 |
| 1000011111 | 1000110011 | 1001000111 |
| 1000100000 | 1000110100 | 1001001000 |
| 1000100001 | 1000110101 | 1001001001 |
| 1000100010 | 1000110110 | 1001001010 |
| 1000100011 | 1000110111 | 1001001011 |
| 1000100100 | 1000111000 | 1001001100 |
| 1000100101 | 1000111001 | 1001001101 |
| 1000100110 | 1000111010 | 1001001110 |
| 1000100111 | 1000111011 | 1001001111 |
| 1000101000 | 1000111100 | 1001010000 |
| 1000101001 | 1000111101 | 1001010001 |
| 1000101010 | 1000111110 | 1001010010 |
| 1000101011 | 1000111111 | 1001010011 |
| 1000101100 | 1001000000 | 1001010100 |
| 1000101101 | 1001000001 | 1001010101 |
| 1000101110 | 1001000010 | 1001010110 |
| 1000101111 | 1001000011 | 1001010111 |

|            |            |            |
|------------|------------|------------|
| 1001011000 | 1001101100 | 1010000000 |
| 1001011001 | 1001101101 | 1010000001 |
| 1001011010 | 1001101110 | 1010000010 |
| 1001011011 | 1001101111 | 1010000011 |
| 1001011100 | 1001110000 | 1010000100 |
| 1001011101 | 1001110001 | 1010000101 |
| 1001011110 | 1001110010 | 1010000110 |
| 1001011111 | 1001110011 | 1010000111 |
| 1001100000 | 1001110100 | 1010001000 |
| 1001100001 | 1001110101 | 1010001001 |
| 1001100010 | 1001110110 | 1010001010 |
| 1001100011 | 1001110111 | 1010001011 |
| 1001100100 | 1001111000 | 1010001100 |
| 1001100101 | 1001111001 | 1010001101 |
| 1001100110 | 1001111010 | 1010001110 |
| 1001100111 | 1001111011 | 1010001111 |
| 1001101000 | 1001111100 | 1010010000 |
| 1001101001 | 1001111101 | 1010010001 |
| 1001101010 | 1001111110 | 1010010010 |
| 1001101011 | 1001111111 | 1010010011 |

|            |            |            |
|------------|------------|------------|
| 1010010100 | 1010101000 | 1010111100 |
| 1010010101 | 1010101001 | 1010111101 |
| 1010010110 | 1010101010 | 1010111110 |
| 1010010111 | 1010101011 | 1010111111 |
| 1010011000 | 1010101100 | 1011000000 |
| 1010011001 | 1010101101 | 1011000001 |
| 1010011010 | 1010101110 | 1011000010 |
| 1010011011 | 1010101111 | 1011000011 |
| 1010011100 | 1010110000 | 1011000100 |
| 1010011101 | 1010110001 | 1011000101 |
| 1010011110 | 1010110010 | 1011000110 |
| 1010011111 | 1010110011 | 1011000111 |
| 1010100000 | 1010110100 | 1011001000 |
| 1010100001 | 1010110101 | 1011001001 |
| 1010100010 | 1010110110 | 1011001010 |
| 1010100011 | 1010110111 | 1011001011 |
| 1010100100 | 1010111000 | 1011001100 |
| 1010100101 | 1010111001 | 1011001101 |
| 1010100110 | 1010111010 | 1011001110 |
| 1010100111 | 1010111011 | 1011001111 |

|            |            |            |
|------------|------------|------------|
| 1011010000 | 1011100100 | 1011111000 |
| 1011010001 | 1011100101 | 1011111001 |
| 1011010010 | 1011100110 | 1011111010 |
| 1011010011 | 1011100111 | 1011111011 |
| 1011010100 | 1011101000 | 1011111100 |
| 1011010101 | 1011101001 | 1011111101 |
| 1011010110 | 1011101010 | 1011111110 |
| 1011010111 | 1011101011 | 1011111111 |
| 1011011000 | 1011101100 | 1100000000 |
| 1011011001 | 1011101101 | 1100000001 |
| 1011011010 | 1011101110 | 1100000010 |
| 1011011011 | 1011101111 | 1100000011 |
| 1011011100 | 1011110000 | 1100000100 |
| 1011011101 | 1011110001 | 1100000101 |
| 1011011110 | 1011110010 | 1100000110 |
| 1011011111 | 1011110011 | 1100000111 |
| 1011100000 | 1011110100 | 1100001000 |
| 1011100001 | 1011110101 | 1100001001 |
| 1011100010 | 1011110110 | 1100001010 |
| 1011100011 | 1011110111 | 1100001011 |

|            |            |            |
|------------|------------|------------|
| 1100001100 | 1100100000 | 1100110100 |
| 1100001101 | 1100100001 | 1100110101 |
| 1100001110 | 1100100010 | 1100110110 |
| 1100001111 | 1100100011 | 1100110111 |
| 1100010000 | 1100100100 | 1100111000 |
| 1100010001 | 1100100101 | 1100111001 |
| 1100010010 | 1100100110 | 1100111010 |
| 1100010011 | 1100100111 | 1100111011 |
| 1100010100 | 1100101000 | 1100111100 |
| 1100010101 | 1100101001 | 1100111101 |
| 1100010110 | 1100101010 | 1100111110 |
| 1100010111 | 1100101011 | 1100111111 |
| 1100011000 | 1100101100 | 1101000000 |
| 1100011001 | 1100101101 | 1101000001 |
| 1100011010 | 1100101110 | 1101000010 |
| 1100011011 | 1100101111 | 1101000011 |
| 1100011100 | 1100110000 | 1101000100 |
| 1100011101 | 1100110001 | 1101000101 |
| 1100011110 | 1100110010 | 1101000110 |
| 1100011111 | 1100110011 | 1101000111 |

|            |            |            |
|------------|------------|------------|
| 1101001000 | 1101011100 | 1101110000 |
| 1101001001 | 1101011101 | 1101110001 |
| 1101001010 | 1101011110 | 1101110010 |
| 1101001011 | 1101011111 | 1101110011 |
| 1101001100 | 1101100000 | 1101110100 |
| 1101001101 | 1101100001 | 1101110101 |
| 1101001110 | 1101100010 | 1101110110 |
| 1101001111 | 1101100011 | 1101110111 |
| 1101010000 | 1101100100 | 1101111000 |
| 1101010001 | 1101100101 | 1101111001 |
| 1101010010 | 1101100110 | 1101111010 |
| 1101010011 | 1101100111 | 1101111011 |
| 1101010100 | 1101101000 | 1101111100 |
| 1101010101 | 1101101001 | 1101111101 |
| 1101010110 | 1101101010 | 1101111110 |
| 1101010111 | 1101101011 | 1101111111 |
| 1101011000 | 1101101100 | 1110000000 |
| 1101011001 | 1101101101 | 1110000001 |
| 1101011010 | 1101101110 | 1110000010 |
| 1101011011 | 1101101111 | 1110000011 |

|            |            |            |
|------------|------------|------------|
| 1110000100 | 1110011000 | 1110101100 |
| 1110000101 | 1110011001 | 1110101101 |
| 1110000110 | 1110011010 | 1110101110 |
| 1110000111 | 1110011011 | 1110101111 |
| 1110001000 | 1110011100 | 1110110000 |
| 1110001001 | 1110011101 | 1110110001 |
| 1110001010 | 1110011110 | 1110110010 |
| 1110001011 | 1110011111 | 1110110011 |
| 1110001100 | 1110100000 | 1110110100 |
| 1110001101 | 1110100001 | 1110110101 |
| 1110001110 | 1110100010 | 1110110110 |
| 1110001111 | 1110100011 | 1110110111 |
| 1110010000 | 1110100100 | 1110111000 |
| 1110010001 | 1110100101 | 1110111001 |
| 1110010010 | 1110100110 | 1110111010 |
| 1110010011 | 1110100111 | 1110111011 |
| 1110010100 | 1110101000 | 1110111100 |
| 1110010101 | 1110101001 | 1110111101 |
| 1110010110 | 1110101010 | 1110111110 |
| 1110010111 | 1110101011 | 1110111111 |

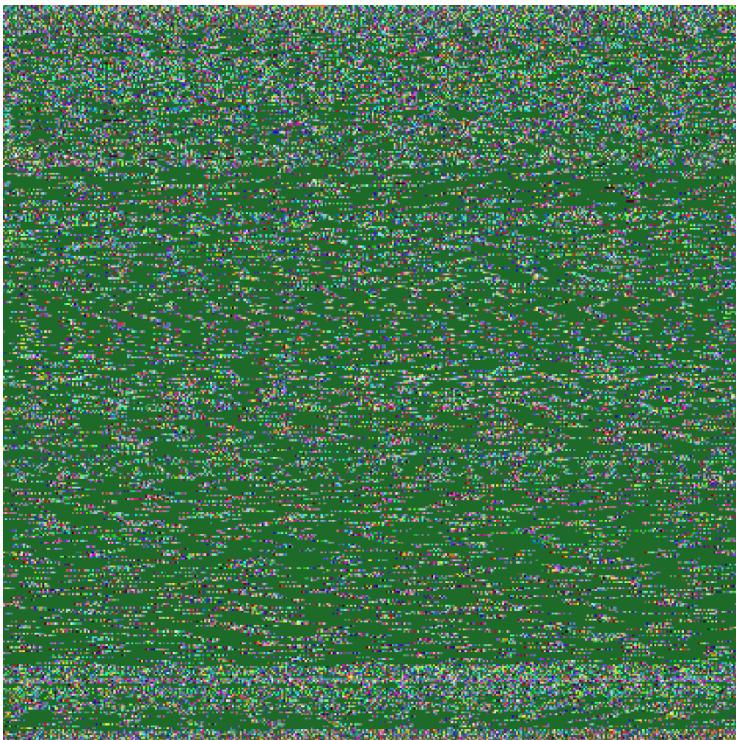
|            |            |            |
|------------|------------|------------|
| 1111000000 | 1111010100 | 1111101000 |
| 1111000001 | 1111010101 | 1111101001 |
| 1111000010 | 1111010110 | 1111101010 |
| 1111000011 | 1111010111 | 1111101011 |
| 1111000100 | 1111011000 | 1111101100 |
| 1111000101 | 1111011001 | 1111101101 |
| 1111000110 | 1111011010 | 1111101110 |
| 1111000111 | 1111011011 | 1111101111 |
| 1111001000 | 1111011100 | 1111110000 |
| 1111001001 | 1111011101 | 1111110001 |
| 1111001010 | 1111011110 | 1111110010 |
| 1111001011 | 1111011111 | 1111110011 |
| 1111001100 | 1111100000 | 1111110100 |
| 1111001101 | 1111100001 | 1111110101 |
| 1111001110 | 1111100010 | 1111110110 |
| 1111001111 | 1111100011 | 1111110111 |
| 1111010000 | 1111100100 | 1111111000 |
| 1111010001 | 1111100101 | 1111111001 |
| 1111010010 | 1111100110 | 1111111010 |
| 1111010011 | 1111100111 | 1111111011 |

1111111100

1111111101

1111111110

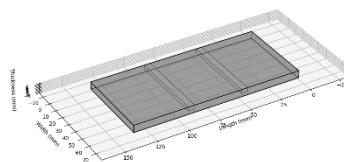
1111111111



*1 Hash of this programmatic Text*



3D View of Stainless Steel Slab with Slots



### ----- Content of 1.cpp -----

```

int cell;
int rdiv;
unsigned long long id;
id = 0;
int rin;
int nbr_comb = pow(k+1, n);
if(cheque == 1){
 for (row=0; row < nbr_comb; row++){
 id++; fprintf(p,"%d ", (id- 1));
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);
 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 fprintf(p,"%c", a[rin]);
 }
 fprintf(p,"\n");
 //printf("\n");
 }
}
if(cheque == 0){
 for (row=0; row < nbr_comb; row++){
 id++; fprintf(p,"%d ", (id- 1));
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 cell = (row/rdiv) % (k+1);
 fprintf(p,"%c", a[cell]);
 }
 fprintf(p,"\n");
 //printf("\n");
 }
}
//fprintf(p,"\n\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
fclose(p);
//end of adaptation
return 0;
}

```

----- Content of 2.cpp-----

```

#include <iostream>
#include <fstream>
#include <vector>
#include <filesystem>
#include <string>
#include <unordered_map>

using namespace std;
namespace fs = std::filesystem;

vector<unordered_map<string, string>> readMappings() {

```

```

vector<unordered_map<string, string>> mappings(4);
for (int i = 1; i <= 4; ++i) {
 ifstream file(to_string(i) + ".txt");
 if (!file) {
 cerr << "Error opening file " << i << ".txt" << endl;
 exit(EXIT_FAILURE);
 }
 string line;
 while (getline(file, line)) {
 int spacePos = line.find(' ');
 if (spacePos != string::npos) {
 string numberStr = line.substr(0, spacePos);
 string str = line.substr(spacePos + 1);
 mappings[i-1][str] = numberStr;
 }
 }
}
return mappings;
}

int main() {
 auto mappings = readMappings();

 int start, end;
 cout << "Enter the range of files to read (e.g., 5 15 to read from 5.txt to 15.txt): ";
 cin >> start >> end;

 int maxFileNameNumber = -1;

 for (const auto& entry : fs::directory_iterator("./")) {
 string filename = entry.path().filename().string();
 if (filename.find(".txt") != string::npos) {
 int fileNameNumber = stoi(filename.substr(0, filename.find(".txt")));
 maxFileNameNumber = max(maxFileNameNumber, fileNameNumber);
 }
 }

 for (int i = start; i <= end; ++i) {
 ifstream inputFile("inputs/" + to_string(i) + ".txt");
 if (!inputFile) {
 cerr << "Could not open the file " + to_string(i) + ".txt" << endl;
 continue;
 }

 string line, content;
 while (getline(inputFile, line)) {
 content += line + "\n";
 }
 }
}

```

```

inputFile.close();

string outputContent;
for (size_t i = 0; i < content.size(); ++i) {
 if (content[i] == '\n') {
 outputContent += "1.27 "; // Assuming 27 is the mapping for
newline character
 continue;
 }
 bool found = false;
 for (int len = 4; len >= 1;--len) {
 if (i + len - 1 < content.size()) {
 string substr = content.substr(i, len);
 if (mappings[len - 1].find(substr) != mappings[len -
1].end()) {
 int number = stoi(mappings[len - 1][substr]);
 outputContent += to_string(len) + "." +
to_string(number + 1) + " ";
 i += len - 1;
 found = true;
 break;
 }
 }
 }
 if (!found) {
 outputContent += "0 ";
 }
}

ofstream outputFile(to_string(maxFileNameNumber + 1) + ".txt");
if (outputFile) {
 outputFile << outputContent;
 outputFile.close();
 maxFileNameNumber++;
} else {
 cerr << "Could not create the file " + to_string(maxFileNameNumber + 1) + ".txt" << endl;
}
}

return 0;
}

```

----- Content of 3.cpp-----

```

#include <iostream>
#include <fstream>
#include <vector>
#include <unordered_map>
#include <sstream>

```

```
using namespace std;

vector<unordered_map<string, string>> readMappings() {
 vector<unordered_map<string, string>> mappings(4);
 for (int i = 1; i <= 4; ++i) {
 ifstream file(to_string(i) + ".txt");
 if (!file) {
 cerr << "Error opening file " << i << ".txt" << endl;
 exit(EXIT_FAILURE);
 }
 string line;
 while (getline(file, line)) {
 int spacePos = line.find(' ');
 if (spacePos != string::npos) {
 string numberStr = line.substr(0, spacePos);
 string str = line.substr(spacePos + 1);
 mappings[i-1][numberStr] = str;
 }
 }
 }
 return mappings;
}

int main() {
 auto mappings = readMappings();

 string inputFilePath;
 cout << "Enter the path of the file to re-convert: ";
 cin >> inputFilePath;

 ifstream inputFile(inputFilePath);
 if (!inputFile) {
 cerr << "Could not open the file " << inputFilePath << endl;
 exit(EXIT_FAILURE);
 }

 string line, content;
 while (getline(inputFile, line)) {
 content += line + '\n';
 }
 inputFile.close();

 stringstream ss(content);
 string token;
 string outputContent;
 while (getline(ss, token, ',')) {
 int dotPos = token.find('.');
 if (dotPos != string::npos) {
 int len = stoi(token.substr(0, dotPos));
 if (len <= 4) {
 outputContent += mappings[len - 1][token] + ", ";
 }
 } else {
 outputContent += token + ", ";
 }
 }
 cout << outputContent << endl;
}
```

```

 string numberStr = token.substr(dotPos + 1);
 if (numberStr == "27" && len == 1) {
 outputContent += '\n';
 } else if (numberStr != "0") {
 numberStr = to_string(stoi(numberStr) - 1);
 if (mappings[len - 1].find(numberStr) != mappings[len - 1].end()) {
 outputContent += mappings[len - 1][numberStr];
 } else {
 outputContent += "?"; // Placeholder for unmapped
 }
 }
 }

 values
}

}

}

ofstream outputFile("0.txt");
if (outputFile) {
 outputFile << outputContent;
 outputFile.close();
} else {
 cerr << "Could not create the file 0.txt" << endl;
}

return 0;
}

```

----- Content of 4.cpp-----

```

//compile cptoeexe.cpp in terminal or CMD Prompt
//g++-std=c++17-lstdc++fs cptoeexe.cpp-o cptoeexe
#include <iostream>
#include <cstdlib>
#include <filesystem>

int main() {
 // Get the current working directory
 std::filesystem::path current_path = std::filesystem::current_path();

 // Iterate through each file in the current directory
 for (const auto& entry : std::filesystem::directory_iterator(current_path)) {
 std::filesystem::path file_path = entry.path();

 // Check if the file has a .cpp extension
 if (file_path.extension() == ".cpp") {
 std::string compile_command = "g++ " + file_path.string() + " -o " + file_path.stem().string();

 // Compile the .cpp file
 std::cout << "Compiling: " << file_path.string() << std::endl;
 std::system(compile_command.c_str());
 }
 }
}

```

```
 }
 }

 return 0;
}
```

----- Content of 5.py-----

```
from tkinter import Tk, Label, Button, filedialog
from PIL import Image, ImageDraw
import math

Predefined color palette with 12 unique colors
COLOR_PALETTE = [
 '#FF5733', '#33FF57', '#3357FF', '#F1C40F', '#8E44AD',
 '#1ABC9C', '#E74C3C', '#2ECC71', '#3498DB', '#9B59B6',
 '#FF33A1', '#A1FF33'
]

class App:
 def __init__(self, master):
 self.master = master
 master.title("Hex Color Picker")

 self.label = Label(master, text="Choose Text File")
 self.label.pack()

 self.open_file_button = Button(master, text="Open File", command=self.open_file)
 self.open_file_button.pack()

 self.close_button = Button(master, text="Generate Image", command=self.generate_image)
 self.close_button.pack()

 self.save_pdf_button = Button(master, text="Save as PDF", command=self.save_as_pdf)
 self.save_pdf_button.pack()

 self.data = []
 self.color_encoding = {}

 def open_file(self):
 file_path = filedialog.askopenfilename(filetypes=[("Text files", "*.txt"), ("All files", "*.*")])
 if file_path:
 try:
 with open(file_path, 'r') as file:
 self.data = [float(num) for num in file.read().split()]
 self.label.config(text="File successfully opened.")
 except Exception as e:
 self.label.config(text=f"Error opening file: {e}")
 else:
 self.label.config(text="No file selected.")
```

```

def generate_image(self):
 print("Generating ...")
 if hasattr(self, 'data'):
 self.label.config(text="Generating, please wait...")
 self.master.update_idletasks() # Force update of the GUI

 # Convert data to colors
 color_data = [self.data_to_color(datum) for datum in self.data]

 self.label.config(text="Data converted to colors. Creating image...")
 self.master.update_idletasks() # Force update of the GUI

 create_image(color_data, self.data)

 self.label.config(text="Image successfully generated.")
else:
 self.label.config(text="Please select a file first.")

def data_to_color(self, datum):
 # Perform the modular arithmetic-based color encoding
 first_digit = int(str(datum)[0]) - 1
 fractional_part = int(str(datum).split('.')[1])
 cpoint = fractional_part % 12
 pointer = (first_digit + cpoint) % 12
 return COLOR_PALETTE[pointer]

def save_as_pdf(self):
 if hasattr(self, 'generated_image'):
 file_path = filedialog.asksaveasfilename(defaultextension='.pdf', filetypes=[("PDF files", "*.pdf")])
 if file_path:
 # A5 dimensions in pixels (previously calculated)
 a5_width_pixels = 1748
 a5_height_pixels = 2480

 # Call the function to save the image as a PDF
 save_image_as_pdf(self.generated_image, file_path, a5_width_pixels, a5_height_pixels)
 self.label.config(text="Saved as PDF.")

 else:
 self.label.config(text="No file selected.")

 else:
 self.label.config(text="Generate an image first.")

def create_image(color_data, data):
 # Count unique string encodings
 unique_encodings = set(data)
 num_unique_encodings = len(unique_encodings)
 total_encodings = len(data) # This will count all the encodings
 print(f"Number of unique encodings: {num_unique_encodings}")
 print(f"Total number of encodings: {total_encodings}")

```

```

Request image dimensions and pixel size from user
pixel_side_length = int(input("Enter the square side length of each pixel to be created: "))
num_pixels_width = int(input("Enter the pixel width of the image, by number of pixels: "))
num_pixels_height = int(input("Enter the pixel height of the image, by number of pixels: "))

img_width = pixel_side_length * num_pixels_width
img_height = pixel_side_length * num_pixels_height

Create an empty white canvas
image = Image.new('RGB', (img_width, img_height), color="white")

Create a drawing context for the image
draw = ImageDraw.Draw(image)

Save the image in the object for later PDF conversion
app.generated_image = image

data_idx = 0 # Index to track data position
for i in range(num_pixels_height):
 for j in range(num_pixels_width):
 if data_idx < len(color_data):
 hex_value = color_data[data_idx]
 top_left = (j * pixel_side_length, i * pixel_side_length)
 bottom_right = ((j + 1) * pixel_side_length, (i + 1) * pixel_side_length)
 draw.rectangle([top_left, bottom_right], fill=hex_value)
 data_idx += 1

image.show()
image.save('generated_image.png')

return image

def save_image_as_pdf(image, file_path, a5_width_pixels, a5_height_pixels, margin_mm=10):
 margin_pixels = int(margin_mm * (300 / 25.4))
 drawable_width = a5_width_pixels - (2 * margin_pixels)
 drawable_height = a5_height_pixels - (2 * margin_pixels)
 num_pages_horizontal = math.ceil(image.width / drawable_width)
 num_pages_vertical = math.ceil(image.height / drawable_height)

 pdf_pages = []

 for y in range(num_pages_vertical):
 for x in range(num_pages_horizontal):
 left = x * drawable_width
 upper = y * drawable_height
 right = min(left + drawable_width, image.width)
 lower = min(upper + drawable_height, image.height)
 crop_area = (left, upper, right, lower)

```

```

cropped_image = image.crop(crop_area)
pdf_page = Image.new('RGB', (a5_width_pixels, a5_height_pixels), 'white')

paste_position =
 margin_pixels + max(0, (drawable_width - cropped_image.width) // 2),
 margin_pixels + max(0, (drawable_height - cropped_image.height) // 2)
)

pdf_page.paste(cropped_image, paste_position)
pdf_pages.append(pdf_page)

pdf_pages[0].save(file_path, save_all=True, append_images=pdf_pages[1:], resolution=100.0)

root = Tk()
app = App(root)
root.mainloop()

----- Content of AlgorithmicProof.c-----
//Version of A Mathematical Proof Assistant Console Application
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

int main()
{
 FILE *p;
 p = fopen("method.txt", "w");
 start:
 printf("\n\tHave Your Statement at the ready. 1 is for yes, 2 is for no. Anything else will produce
an error. Note, the output of this program is in the same directory as this program. The output is named,
'method.txt'.");
 int bin, trya;
 puts("\n\nCan you write the statement in the form: P(x) for some element in X?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 puts("Can you spot a value of x being an element of X for when P(x) is true?\n\t");
 scanf("%d", &bin);
 if(bin == 1{
 //Proof_Path_1
 puts("Prove P(x) by construction. Start your proof with a commonly known
definition.");
 fprintf(p, "Prove P(x) by construction. Start your proof with a commonly
known definition.");
 }
 }
 if(bin == 0){
 puts("Can you write the statement in the form: P(x) for all x as elements in X?\n\t");
 scanf("%d", &bin);
 }
}

```

```

if(bin == 1){
 puts("Can you spot a value of x as an element of X for which P(x) is
false?\n\t");
 if(bin == 1){
 //Proof_Path_2
 puts("Disprove P(x) by counter-example. Start your proof with a
commonly known definition.");
 fprintf(p, "Disprove P(x) by counter-example. Start your proof with
a commonly known definition.");
 }
 if(bin == 0){
 puts("Is x small enough for each x to be checked in turn?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 //Proof_Path_3
 puts("Prove P(x) by exhaustion. Start your proof with a
commonly known definition.");
 fprintf(p, "Prove P(x) by exhaustion. Start your proof with
a commonly known definition.");
 }
 if(bin == 0){
 puts("Is X the set of natural numbers greater than or
equal to some N as an element of the Set of Natural numbers?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 puts("Can you show P(N) is true, and prove the
inductive step P(k) which implies P(k + 1) for k greater than or equal to N?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 //Proof_Path_4
 puts("Prove P(N) and the inductive
step P(k) which implies P(k + 1) by induction. Start your proof with a commonly known definition.");
 fprintf(p, "Prove P(N) and the
inductive step P(k) which implies P(k + 1) by induction. Start your proof with a commonly known
definition.");
 }
 if(bin == 0){
 puts("Can you write the statement in
the form P implies Q and Q implies P?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 puts("Can you find a
sequence of equivalences where P implies P sub 1 ... which implies Q and Q implies P sub 1 ... which
implies P?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 //Proof_Path_5
 puts("Prove by
deduction. Start your proof with a commonly known definition.");
 fprintf(p, "Prove
");
 }
 }
 }
 }
 }
 }
}

```



```

scanf("%d", &bin);
if(bin == 1){
 puts("Can you find a sequence of equivalences where P implies P
sub 1 ... which implies Q and Q implies P sub 1 ... which implies P?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 //Proof_Path_7
 puts("Prove P implies Q and Q implies P by deduction.

Start your proof with a commonly known definition.");
 fprintf(p, "Prove P implies Q and Q implies P by
deduction. Start your proof with a commonly known definition.");
 }
 if(bin == 0){
 puts("Can you find the sequences of implications P
implies P sub 1 ... which implies Q and Q implies Q sub 1 which implies P?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 //Proof_Path_8
 puts("Prove P implies Q and Q implies P by
deduction. Start your proof with a commonly known definition.");
 fprintf(p, "Prove P implies Q and Q implies P by
deduction. Start your proof with a commonly known definition.");
 }
 if(bin == 0){
 puts("Your statement is either invalid or you
have misunderstood one of the previous questions. Please try again. Or your statement is beyond the scope
of this program's algorithm.");
 puts("Do you want to try again?\n\t");
 scanf("%d", &trya);
 if(trya == 1){
 goto start;
 }
 if(trya == 0){
 fprintf(p, "An error was produced
(Error Code 002). Please try again or, use another Statement. Thank you.");
 exit(0);
 }
 }
 }
}
if(bin == 0){
 puts("Can you write the statement in the form P implies Q?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 puts("Can you find a sequence of implications where P
implies P sub 1 ... which implies P?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 //Proof_Path_9
 puts("Prove P implies Q by deduction. Start

```

```

your proof with a commonly known definition.");
fprintf(p, "Prove P implies Q by deduction. Start
your proof with a commonly known definition.");
}
if(bin == 0){
 puts("Can you find a sequence of implications for
the contrapositive: NOT Q implies P sub 1 ... which implies NOT P?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 //Proof_Path_10
 puts("Prove P implies Q by
contraposition. Start your proof with a commonly known definition.");
 fprintf(p, "Prove P implies Q by
contraposition. Start your proof with a commonly known definition.");
 }
 if(bin == 0){
 puts("Your statement is either invalid
or you have misunderstood one of the previous questions. Please try again. Or your statement is beyond
the scope of this program's algorithm.");
 puts("Do you want to try
again?\n\t");
 scanf("%d", &trya);
 if(trya == 1){
 goto start;
 }
 if(trya == 0){
 fprintf(p, "An error was
produced (Error Code 003). Please try again or, use another Statement. Thank you.");
 exit(0);
 }
 }
}
if(bin == 0){
 puts("Let the statement be P. Can you find a sequence P
implies P sub 1 ... which implies P sub n which implies Q and Q implies P sub n which implies P sub 1 ...
which implies P, where Q is true?\n\t");
 scanf("%d", &bin);
 if(bin == 1){
 //Proof_Path_11
 puts("Prove P by equivalence. Start your proof
with a commonly known definition.");
 fprintf(p, "Prove P by equivalence. Start your
proof with a commonly known definition.");
 }
 if(bin == 0){
 puts("Let the statement be P. Can you find a
sequence NOT P implies P sub 1 ... which implies P sub n, where P sub n is false?\n\t");
 scanf("%d", &bin);
 if(bin == 1){

```

```

//Proof_Path_12
puts("Prove P by contradiction. Start
your proof with a commonly known definition.");
Start your proof with a commonly known definition.");
}
if(bin == 0){
 puts("Your statement is either invalid
or you have misunderstood one of the previous questions. Please try again. Or your statement is beyond
the scope of this program's algorithm.");
 puts("Do you want to try
again?\n\t");
 scanf("%d", &trya);
 if(trya == 1){
 goto start;
 }
 if(trya == 0){
 fprintf(p, "An error was
produced (Error Code 004). Please try again or, use another Statement. Thank you.");
 exit(0);
 }
}
}
}
}
fclose(p);
return 0;
}

```

----- Content of Arithmetic\_Modulo\_n.py-----

```

import tkinter as tk
from tkinter import ttk

def create_table(op_type, n):
 for widget in table_frame.winfo_children():
 widget.destroy()

 for i in range(n+1):
 for j in range(n+1):
 if i == 0 and j == 0:
 label = tk.Label(table_frame, text=" ", width=5, height=2, borderwidth=1, relief="solid")
 elif i == 0:
 label = tk.Label(table_frame, text=str(j-1), width=5, height=2, borderwidth=1, relief="solid")
 elif j == 0:
 label = tk.Label(table_frame, text=str(i-1), width=5, height=2, borderwidth=1, relief="solid")
 else:
 if op_type == "Addition":

```

```

label = tk.Label(table_frame, text=str((i-1)+(j-1) % n), width=5, height=2, borderwidth=1,
relief="solid")
elif op_type == "Multiplication":
 label = tk.Label(table_frame, text=str((i-1)*(j-1) % n), width=5, height=2, borderwidth=1,
relief="solid")

label.grid(row=i, column=j)

root = tk.Tk()
root.title("Modular Arithmetic Table")

control_frame = tk.Frame(root)
control_frame.pack(side=tk.TOP, padx=10, pady=10)

table_frame = tk.Frame(root)
table_frame.pack(side=tk.BOTTOM, padx=10, pady=10)

operation_label = tk.Label(control_frame, text="Operation:")
operation_label.pack(side=tk.LEFT)

operation_combobox = ttk.Combobox(control_frame, values=["Addition", "Multiplication"],
state="readonly")
operation_combobox.pack(side=tk.LEFT)
operation_combobox.current(0)

n_label = tk.Label(control_frame, text="Modulo (n):")
n_label.pack(side=tk.LEFT)

n_entry = tk.Entry(control_frame, width=5)
n_entry.pack(side=tk.LEFT)
n_entry.insert(0, "6")

def update_table():
 op_type = operation_combobox.get()
 n = int(n_entry.get())
 create_table(op_type, n)

update_button = tk.Button(control_frame, text="Update Table", command=update_table)
update_button.pack(side=tk.LEFT)

create_table("Addition", 6)

root.mainloop()

```

----- Content of bertotools.py-----

```

import tkinter as tk
import random
from PIL import ImageGrab
import pickle

```

```
from tkinter import simpledialog
import os
from tkinter import Menu, Checkbutton, BooleanVar
from tkinter.colorchooser import askcolor
import logging
import time
import json
import os
import pyautogui
import keyboard

current_col = 0
current_row = 0

def get_random_color():
 r = lambda: random.randint(0,255)
 return '#%02X%02X%02X' % (r(),r(),r())

def get_random_char():
 random_int = random.randint(0x0021, 0x007E)
 return chr(random_int)

def draw_grid():
 for i in range(grid_size):
 for j in range(grid_size):
 color = get_random_color()
 square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size, (i+1)*square_size,
 fill=color)
 char = get_random_char()
 text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2, text=char,
 font=("Arial", 18), anchor="center")

def draw_grid_IDE():
 global col
 try:
 for i in range(grid_size):
 for j in range(grid_size):
 color = "blue"
 square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size,
 (i+1)*square_size, fill=color, outline = "white")
 if typing_mode:
 # In the function where you create the canvas...
 canvas.bind("<Key>", on_key_press)
 canvas.focus_set()
 except:
 print("Submit an app mode ...")
```

```

def toggle_color_mode():
 """Toggles between manual color mode and random color mode"""
 global color_mode_var
 color_mode_var.set(not color_mode_var.get())

font_size = 9

def draw_char1(canvas, char, row, col, font_size, color):
 global cell_size, font_color, square_size, grid_size, last_drawn, note, char_note

 ...
 def draw_char1(canvas, char, row, col, font_size):
 Draws a character on the Tkinter canvas at the specified row and column with the specified font size.

 Parameters:
 canvas (tk.Canvas): The Tkinter canvas to draw on.
 char (str): The character to draw.
 row (int): The row to draw the character at.
 col (int): The column to draw the character at.
 font_size (int): The font size to use for the character.

 ...

try:
 x = col * square_size + square_size / 2
 y = row * square_size + square_size / 2

 print(f"Drawing char '{char}' at ({x}, {y})") # Debug print to check coordinates
 canvas.create_rectangle(col * square_size, row * square_size, (col+1) * square_size, (row+1) * square_size, fill=color)
 canvas.create_text(x, y, text=char, fill=font_color, font=('Calibri', int(font_size)))
 note = char_note.get()
 # Log the color and character info
 logging.info(f"Position:{x};{y};Xp:{x};Yp:{y};Character:{char};Square color:{color};Font color:{font_color};Note:{note}")
 # Log the color and character info in a text file
 if(note == ""):
 note = "emptyNote"
 with open("color_log.txt", "a", encoding='utf-8') as login:
 print(f"{x},{y},{char},{color},{font_color},{note}", file=login)

except Exception as e:
 print(f"Something Went Wrong... Error: {e}")

def draw_char(i=None, j=None):
 try:

```

```

"""Draws a character from input field in a specific square"""
global char_count, IDE_mode, typing_mode, last_drawn, font_color
if(mode == 'IDE'):
 IDE_mode = True
Check if last_drawn is defined, otherwise define it
if 'last_drawn' not in globals():
 last_drawn = []

If i, j are not provided, calculate them based on char_count
if i is None or j is None:
 i, j = divmod(char_count, grid_size)
 # Handle out-of-grid situations
 if i >= grid_size or j >= grid_size:
 print('Out of grid!')
 return

Generate a random color if not in typing mode, white otherwise
##color = get_random_color() if not typing_mode else "white"

Check color mode
if color_mode_var.get():
 # Manual color mode
 # Show a color picker and get the chosen color for the square
 color = askcolor(title="Choose square color")[1]

 # Ask for the font color
 color_result = askcolor(title="Choose font color")
 if color_result is not None:
 font_color = color_result[1]
 else:
 # Handle the case when the user cancelled the color selection
 font_color = "#000000" # default to black, for example

else:
 # Random color mode
 color = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
 color1 = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
 font_color = color1 # "#000000"

square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size, (i+1)*square_size,
fill=color)
char = char_entry.get()[1]
note = char_note.get()
text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2, text=char,
font=("Arial", font_size), fill=font_color, anchor="center")

"""

Log the color and character info

```

```

logging.info(f"Position:{i};{j}',Xp:{i}',Yp:{j}',Character:{char}',Square color:{color}',Font
color:{font_color}'")
Log the color and character info in a text file
with open("color_log.txt", "a") as login:
 print(f"{i},{j},{char},{color},{font_color}", file=login)
 ""

Log the color and character info
logging.info(f"Position:{i};{j}',Xp:{i}',Yp:{j}',Character:{char}',Square color:{color}',Font
color:{font_color}',Note:{note}'")
Log the color and character info in a text file
if(note == ""):
 note = "emptyNote"
with open("color_log.txt", "a", encoding='utf-8') as login:
 print(f"{i},{j},{char},{color},{font_color},{note}", file=login)

if len(char_entry.get()) > 0: # Check if there's more than one character
 char_entry.delete(0, 1) # Delete the first character

last_drawn.append((square, text))

if not IDE_mode:
 char_count = (char_count + 1) % (grid_size * grid_size)
 if char_count == 0: # If we've filled the canvas, clear it
 canvas.delete('all')

 return square, text
except:
 print("Submit an app mode ...")

def adjust_grid_and_font():
 global grid_size, square_size, font_size, canvas_width, current_row, current_col

 n_g_s = simpledialog.askstring("Change Grid Size (Grid Width)", "Enter New Grid Size (+Integer):")
 if(n_g_s != ""):
 new_grid_size = n_g_s
 n_f_s = simpledialog.askstring("Change Font Size (Default = 9)", "Enter New Font Size (+Integer):")
 if(n_f_s != ""):
 new_font_size = n_f_s
 # Update the global variables
 if(new_grid_size==None and new_font_size==None):
 grid_size = 10
 font_size = 9
 if((not new_grid_size==None) and (not new_font_size==None)):
 try:
 grid_size = int(new_grid_size)
 square_size = canvas_width / grid_size
 font_size = int(new_font_size)

```

```
except:
 print("positive integers please")

 """
Get new values from input fields
new_grid_size = grid_size_entry.get()
new_font_size = font_size_entry.get()

Update the global variables
grid_size = int(new_grid_size)
square_size = canvas_width // grid_size
font_size = int(new_font_size)
"""

Redraw the grid
refresh_canvas()
qit = input("Are you in typing mode, or want to enter typing mode? 1 for (No), 2 for (yes): ")
if(qit == '2'):
 current_row = 0
 current_col = 0
 show_typing_mode_menu()

def refresh_canvas():
 """Clears the canvas and resets the char_count"""
 global char_count
 canvas.delete('all')
 char_count = 0
 draw_grid_IDE()
 last_drawn.clear()

def undo_last():
 """Undoes the last drawing operation"""
 if last_drawn:
 square, text = last_drawn.pop()
 canvas.delete(square)
 canvas.delete(text)

def update_canvas():
 if not paused.get():
 canvas.delete('all')
 draw_grid()
 root.after(8000, update_canvas)

def toggle_typing_mode():
 global typing_mode, canvas

 typing_mode = not typing_mode
 if typing_mode:
 canvas.focus_set() # Set focus to the canvas for keyboard input
```

```
def on_canvas_click(event):
 try:
 global mode, typing_mode
 if mode == 'IDE' and not typing_mode:
 j = event.x // square_size
 i = event.y // square_size
 draw_char(i, j)
 else:
 paused.set(not paused.get())
 except:
 print("Submit an app mode ...")

def on_key_press(event):
 global char_count, typing_mode
 if typing_mode:
 char_entry.delete(0, 'end') # Clear the entry box
 try:
 utf8_char = event.char.encode('utf-8').decode('utf-8')
 char_entry.insert(0, utf8_char) # Insert the typed character
 except UnicodeDecodeError:
 print("Non UTF-8 character detected")
 return
 draw_char() # Draw the character
 char_count += 1 # Increment the count

 # If char_count exceeds the total number of squares in the grid, reset it
 if char_count >= grid_size ** 2:
 char_count = 0

def submit_mode():

 global char_count, mode, typing_mode

 mode = mode_entry.get().upper()

 """
 if mode in ['editor']:
 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 draw_grid()
```

```
char_label.pack()
char_entry.pack()
char_button.pack()

"""

Draw the initial grid
if(mode != 'IDE'):
 print("A valid mode ...")

"""

if(mode == 'normal'):

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 draw_grid()
 create_menu_1()

if mode == 'normal':
 # Schedule the first update
 root.after(8000, update_canvas)
"""

if mode == 'IDE':

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 char_count = 0
 draw_grid_IDE()

 control_frame.pack()
 char_label.pack()
 char_entry.pack()
 #grid_size_label.pack()
 #grid_size_entry.pack()
 #font_size_label.pack()
 #font_size_entry.pack()
 #adjust_button.pack()
 char_note_label.pack()
 char_note.pack()
 #refresh_button.pack(side="left")
 #undo_button.pack(side="left")
```

```

#save_button = tk.Button(root, text='Save', command=save_canvas, bg="white", padx=5, pady=0)
#save_button.pack(side="left")
#save_Sbutton.pack(side="left")
#load_button.pack(side="left")
create_menu()

return mode

def save_canvas():
 global IC_value, xSwitch, ing
 xSwitch = 1
 ing = 1
 if(ing == 0):
 ing = 2
 ing = 1
 if(xSwitch == 1 and ing == 1):
 IC_value = 0
 ing = 0
 """Save the current state of the canvas to a .png file"""
 #filename = simpledialog.askstring("Save Canvas", "Enter filename:")
 filename = f"ImageCanvas{IC_value}_rename"

if filename: # Only save the canvas if a filename was entered
 # Get the bounding box of the canvas
 x = root.winfo_rootx() + canvas.winfo_x()
 y = root.winfo_rooty() + canvas.winfo_y()
 x1 = x + canvas.winfo_width()
 y1 = y + canvas.winfo_height()
 time.sleep(3)
 # Grab the image, crop it to the bounding box, and save it
 ImageGrab.grab().crop((x, y, x1, y1)).save(filename + ".png")

def save_session():
 """Save the current session to a file using pickle"""
 global last_drawn
 # Ask the user to enter a filename
 filename = simpledialog.askstring("Save Session", "Enter filename:")
 if filename: # Only save the session if a filename was entered
 session_data = [(canvas.coords(square), canvas.itemcget(square, "fill"),
 canvas.coords(text), canvas.itemcget(text, "text"), canvas.itemcget(text, "fill"))
 for square, text in last_drawn]
 with open(filename + ".pkl", "wb") as f:
 pickle.dump(session_data, f)

def load_session():

```

```

"""Load a saved session from a file using pickle"""
global last_drawn
Ask the user to enter a filename
filename = simpledialog.askstring("Load Session", "Enter filename:")
if filename: # Only try to load a session if a filename was entered
 try:
 with open(filename + ".pkl", "rb") as f:
 session_data = pickle.load(f)
 # Clear the canvas and redraw all elements from the loaded session
 canvas.delete('all')
 draw_grid_IDE()
 last_drawn = []
 for square_coords, square_fill, text_coords, text_content, text_fill in session_data:
 square = canvas.create_rectangle(*square_coords, fill=square_fill)
 text = canvas.create_text(*text_coords, text=text_content, fill=text_fill, font=("Arial", font_size),
 anchor="center")
 last_drawn.append((square, text))
 except FileNotFoundError:
 print(f"No session named '{filename}' found.")

#def exit_app():
exit()

def show_about():
 about_window = tk.Toplevel(root)
 about_window.title("About")
 about_msg = "This is a program created to learn and experiment with Tkinter. In IDE mode, you can draw characters on a grid (By typing or copying and pasting then clicking the squares you want to draw each character on), adjust the grid and font size, save and load sessions, and more, like saving the grid as an image (For example). C(num) was written with the aid of Chat GPT. Enjoy!\n\nNote: the workflow ... IDE to color_log.txt (Compiled by compile#.exe or compile#.py);\\nindex_1.txt contains fields of study or research and development and\\nindex_2.txt contains fields of study or research and development also ...\\n\\nBertootools Digital"
 tk.Message(about_window, text=about_msg, width=500).pack()
 tk.Button(about_window, text="OK", command=about_window.destroy).pack()

def compileGOSmX():
 MAX_LINE_LENGTH = 256
 MAX_TOPICS = 256

 global index_1, index_2

 index_1 = [
 "Axiom",
 "Theorem",
 "Lemma",
 "Proposition",
 "Corollary",
 "Conjecture",
 "Proof",
]

```

"Premise",  
"Conclusion",  
"Hypothesis",  
"Counterexample",  
"Direct Proof",  
"Indirect Proof",  
"Proof by Contradiction (Reductio ad absurdum)",  
"Proof by Induction",  
"Proof by Contrapositive",  
"Deductive Reasoning",  
"Inference",  
"Assumption",  
"Statement",  
"Postulate",  
"Proof by Exhaustion",  
"Syllogism",  
"Constructive Proof",  
"Non-Constructive Proof",  
"Trivial Proof",  
"Vacuous Proof",  
"Biconditional",  
"Condition",  
"Sufficiency",  
"Necessity",  
"Quantifier",  
"Universal Quantifier",  
"Existential Quantifier",  
"Bound Variable",  
"Free Variable",  
"Predicate",  
"Propositional Logic",  
"Modus Ponens",  
"Modus Tollens",  
"Discrete Mathematics",  
"Set Theory",  
"Function",  
"Bijection",  
"Injection",  
"Surjection",  
"Equivalence Relation",  
"Partial Order",  
"Total Order",  
"Well-Order",  
"Reflexivity",  
"Symmetry",  
"Transitivity",  
"Antisymmetry",  
"Completeness",  
"Compactness",  
"Connectedness",

"Convergence",  
"Divergence",  
"Limit",  
"Sequence",  
"Series",  
"Monotonicity",  
"Cauchy Sequence",  
"Infinite Set",  
"Finite Set",  
"Cardinality",  
"Countable Set",  
"Uncountable Set",  
"Subset",  
"Superset",  
"Intersection",  
"Union",  
"Empty Set",  
"Power Set",  
"Cartesian Product",  
"Equivalence Class",  
"Partition",  
"Field",  
"Ring",  
"Group",  
"Abelian Group",  
"Non-abelian Group",  
"Matrix",  
"Vector Space",  
"Linear Transformation",  
"Eigenvalue",  
"Eigenvector",  
"Norm",  
"Inner Product",  
"Orthogonality",  
"Basis",  
"Dimension",  
"Rank",  
"Nullity",  
"Determinant",  
"Graph Theory",  
"Vertex",  
"Edge",  
"Connectivity",  
"Cycle",  
"Path",  
"Degree",  
"Subgraph",  
"Tree",  
"Forest",  
"Planar Graph",

"Bipartite Graph",  
"Directed Graph (Digraph)",  
"Eulerian Graph",  
"Hamiltonian Graph",  
"Adjacency Matrix",  
"Incidence Matrix",  
"Isomorphism",  
"Homeomorphism",  
"Topology",  
"Open Set",  
"Closed Set",  
"Boundary",  
"Compact Space",  
"Hausdorff Space",  
"Continuity",  
"Differential",  
"Derivative",  
"Integral",  
"Partial Derivative",  
"Multivariable Calculus",  
"Laplace Transform",  
"Fourier Transform",  
"Taylor Series",  
"Maclaurin Series",  
"Conic Sections",  
"Hyperbola",  
"Ellipse",  
"Parabola",  
"Asymptote",  
"Limits at Infinity",  
"Complex Number",  
"Imaginary Unit",  
"Real Number",  
"Rational Number",  
"Irrational Number",  
"Prime Number",  
"Composite Number",  
"GCD (Greatest Common Divisor)",  
"LCM (Least Common Multiple)",  
"Permutation",  
"Combination",  
"Probability",  
"Statistics",  
"Expected Value",  
"Variance",  
"Standard Deviation",  
"Normal Distribution",  
"Poisson Distribution",  
"Binomial Distribution",  
"Hypothesis Testing",

"Regression",  
"Correlation",  
"Matrix Algebra",  
"Linear Algebra",  
"Vector Calculus",  
"Optimization",  
"Algorithm",  
"Computational Complexity",  
"Big O Notation",  
"Pigeonhole Principle",  
"Principle of Inclusion-Exclusion",  
"Turing Machine",  
"Computability",  
"Unsolvability",  
"Parity",  
"Diophantine Equations",  
"Cryptography",  
"Fermat's Last Theorem",  
"Pythagorean Theorem",  
"Triangle Inequality",  
"Trigonometric Functions",  
"Trigonometric Identities",  
"Polar Coordinates",  
"Euler's Formula",  
"Riemann Zeta Function",  
"P vs NP Problem",  
"NP-complete Problem",  
"Stochastic Process",  
"Markov Chain",  
"Random Variable",  
"Conditional Probability",  
"Bayes' Theorem",  
"Monte Carlo Method",  
"Fractal",  
"Chaos Theory",  
"Game Theory",  
"Nash Equilibrium",  
"Zero-Sum Game",  
"Non-Zero-Sum Game",  
"Linear Programming",  
"Nonlinear Programming",  
"Quadratic Programming",  
"Dynamic Programming",  
"Integer Programming",  
"Graph Coloring",  
"Network Flow",  
"Spanning Tree",  
"Bellman-Ford Algorithm",  
"Dijkstra's Algorithm",  
"Kruskal's Algorithm",

"Prim's Algorithm",  
"Floyd-Warshall Algorithm",  
"Euler's Method",  
"Runge-Kutta Method",  
"Numerical Integration",  
"Numerical Differentiation",  
"Bisection Method",  
"Newton's Method",  
"Secant Method",  
"Fixed Point Iteration",  
"Linear Interpolation",  
"Polynomial Interpolation",  
"Lagrange Interpolation",  
"Splines",  
"Fourier Series",  
"Laplace's Equation",  
"Heat Equation",  
"Wave Equation",  
"Schrodinger Equation",  
"Ordinary Differential Equation (ODE)",  
"Partial Differential Equation (PDE)",  
"Boundary Value Problem",  
"Initial Value Problem",  
"Green's Theorem",  
"Stoke's Theorem",  
"Divergence Theorem",  
"Curl",  
"Gradient",  
"Divergence",  
"Tensor",  
"Manifold",  
"Topological Space",  
"Measure Theory",  
"Lebesgue Integral",  
"Borel Set",  
"Hilbert Space",  
"Banach Space",  
"Category Theory",  
"Functor",  
"Natural Transformation",  
"Sheaf",  
"Homotopy",  
"Homology",  
"Cohomology",  
"Galois Theory",  
"Algebraic Geometry",  
"Topological K-Theory",  
"Knot Theory",  
"Lattice Theory"

]

```
index_2 = [
 "Biochemistry",
 "Biophysics",
 "Molecular biology",
 "Genetics",
 "Immunology",
 "Cell biology",
 "Microbiology",
 "Neuroscience",
 "Pharmacology",
 "Bioinformatics",
 "Biotechnology",
 "Proteomics",
 "Genomics",
 "Structural biology",
 "Virology",
 "Systems biology",
 "Developmental biology",
 "Evolutionary biology",
 "Synthetic biology",
 "Metabolomics",
 "Epigenetics",
 "Tissue engineering",
 "Nanotechnology",
 "Materials science",
 "Quantum physics",
 "Condensed matter physics",
 "Particle physics",
 "Astrophysics",
 "Cosmology",
 "Optics",
 "Atomic and molecular physics",
 "Fluid mechanics",
 "Thermodynamics",
 "Environmental science",
 "Climate science",
 "Geology",
 "Oceanography",
 "Atmospheric science",
 "Ecology",
 "Conservation biology",
 "Botany",
 "Zoology",
 "Entomology",
 "Marine biology",
 "Paleontology",
 "Anthropology",
 "Archaeology",
 "Psychology",
```

"Cognitive science",  
"Social psychology",  
"Linguistics",  
"Artificial intelligence",  
"Machine learning",  
"Computer vision",  
"Natural language processing",  
"Human-computer interaction",  
"Robotics",  
"Computer graphics",  
"Data science",  
"Mathematical modeling",  
"Mathematical physics",  
"Number theory",  
"Algebraic geometry",  
"Differential equations",  
"Computational physics",  
"Mathematical biology",  
"Operations research",  
"Biostatistics",  
"Epidemiology",  
"Cancer research",  
"Diabetes research",  
"Heart disease research",  
"Infectious diseases research",  
"Immunotherapy",  
"Stem cell research",  
"Gene therapy",  
"Drug discovery",  
"Precision medicine",  
"Health informatics",  
"Renewable energy",  
"Energy storage",  
"Sustainable materials",  
"Environmental engineering",  
"Water management",  
"Transportation engineering",  
"Civil engineering",  
"Chemical engineering",  
"Aerospace engineering",  
"Biomedical engineering",  
"Electrical engineering",  
"Mechanical engineering",  
"Robotics engineering",  
"Quantum computing",  
"Cryptography",  
"Cybersecurity",  
"Network engineering",  
"Telecommunications",  
"Human genetics",

"Forensic science",  
"Space exploration and research",  
"Planetary science",  
"Astrobiology",  
"Astrochemistry",  
"Astrogeology",  
"Astroinformatics",  
"Exoplanet research",  
"Stellar evolution",  
"Galactic astronomy",  
"Observational astronomy",  
"Computational astrophysics",  
"Quantum chemistry",  
"Computational chemistry",  
"Organic chemistry",  
"Inorganic chemistry",  
"Physical chemistry",  
"Environmental chemistry",  
"Analytical chemistry",  
"Agricultural science",  
"Food science",  
"Nutritional science",  
"Exercise physiology",  
"Sports science",  
"Biomechanics",  
"Plant physiology",  
"Plant genetics",  
"Plant pathology",  
"Soil science",  
"Hydrology",  
"Geochemistry",  
"Geophysics",  
"Geomorphology",  
"Remote sensing",  
"Geotechnical engineering",  
"Petroleum engineering",  
"Aerospace materials",  
"Nanomaterials",  
"Polymer science",  
"Computational materials science",  
"Photonics",  
"Physical optics",  
"Quantum optics",  
"Neuroengineering",  
"Brain imaging",  
"Cognitive neuroscience",  
"Neuroinformatics",  
"Psychophysics",  
"Developmental psychology",  
"Personality psychology",

"Clinical psychology",  
"Industrial-organizational psychology",  
"Educational psychology",  
"Psycholinguistics",  
"Human genetics",  
"Evolutionary genetics",  
"Population genetics",  
"Genetic engineering",  
"Genetic counseling",  
"Epigenomics",  
"Cardiovascular research",  
"Respiratory research",  
"Gastroenterology research",  
"Endocrinology research",  
"Nephrology research",  
"Hematology research",  
"Ophthalmology research",  
"Orthopedic research",  
"Dermatology research",  
"Veterinary medicine",  
"Animal behavior",  
"Conservation ecology",  
"Wildlife biology",  
"Environmental microbiology",  
"Agricultural economics",  
"Development economics",  
"Behavioral economics",  
"Econometrics",  
"Financial mathematics",  
"Operations management",  
"Supply chain management",  
"Industrial engineering",  
"Human-computer interaction",  
"Virtual reality",  
"Augmented reality",  
"Data mining",  
"Text mining",  
"Big data analytics",  
"Computational linguistics",  
"Quantum information science",  
"Quantum cryptography",  
"Biometrics",  
"Information retrieval",  
"Software engineering",  
"Computer networks",  
"Embedded systems",  
"Human-robot interaction",  
"Control systems",  
"Biopharmaceuticals",  
"Drug delivery systems",

"Clinical trials",  
"Regenerative medicine",  
"Agricultural biotechnology",  
"Plant breeding",  
"Animal breeding",  
"Food technology",  
"Sensory science",  
"Poultry science",  
"Aquaculture",  
"Marine ecology",  
"Limnology",  
"Population ecology",  
"Landscape ecology",  
"Evolutionary ecology",  
"Environmental toxicology",  
"Environmental chemistry",  
"Environmental microbiology",  
"Ecotoxicology",  
"Green chemistry",  
"Space physics",  
"Space weather",  
"Astrostatistics",  
"Computational fluid dynamics",  
"Mathematical optimization",  
"Operations research",  
"Human genetics",  
"Functional genomics",  
"Molecular genetics",  
"Cancer genetics",  
"Psychiatric genetics",  
"Population genomics",  
"Bioengineering",  
"Biomaterials",  
"Biomechatronics",  
"Cardiovascular engineering",  
"Neural engineering",  
"Rehabilitation engineering",  
"Genetic engineering",  
"Environmental engineering",  
"Water resources engineering",  
"Structural engineering",  
"Robotics engineering",  
"Quantum information theory",  
"Quantum simulation",  
"Quantum sensing",  
"Geographical information systems (GIS)",  
"Urban planning",  
"Renewable energy systems",  
"Solar cell technology",  
"Wind energy research",

```

"Energy policy and economics",
"Computational neuroscience",
"Neurobiology",
"Cognitive neuroscience",
"Systems neuroscience",
"Human-robot interaction",
"Evolutionary psychology",
"Social network analysis"
]

def square_print_topic_indices(r_value, g_value, b_value, file):
 index1 = r_value % len(index_1)
 index2 = g_value % len(index_1)
 index3 = b_value % len(index_1)
 ...
 file.write("\t\t\t{topics[index1]}\n")
 file.write("\t\t\t{topics[index2]}\n")
 file.write("\t\t\t{topics[index3]}\n")
 ...
 file.write("\t\t\t{index_1[index1]}\n")
 file.write("\t\t\t{index_1[index2]}\n")
 file.write("\t\t\t{index_1[index3]}\n")

 squareRatio = str(index1) + ":" + str(index2) + ":" + str(index3)
 return squareRatio

def font_print_topic_indices(r_value, g_value, b_value, file):
 index1_ = r_value % len(index_2)
 index2_ = g_value % len(index_2)
 index3_ = b_value % len(index_2)
 file.write("\t\t\t{index_2[index1_]}\n")
 file.write("\t\t\t{index_2[index2_]}\n")
 file.write("\t\t\t{index_2[index3_]}\n")

 fontRatio = str(index1_) + ":" + str(index2_) + ":" + str(index3_)
 return fontRatio

def get_index(i, j, grid_size):
 return i * grid_size + j

def extract_rgb_values(color):
 if len(color) != 7 or color[0] != '#':
 raise ValueError('Input should be a hex color code in the format "#RRGGBB"')
 try:
 red = int(color[1:3], 16)
 green = int(color[3:5], 16)

```

```

blue = int(color[5:7], 16)
return red, green, blue
except ValueError:
 raise ValueError('Invalid color code. RGB values should be hex digits (0-9, A-F)')

def GO():
 log_file = "color_log.txt"
 ## topic_file = "index_1.txt"
 ## topic2_file = "index_2.txt" # Corrected this line to read from a different file
 gridWidth = int(input("Enter gridWidth: "))
 # Read topic files to get the topics and topic2s
 ## with open(topic_file, "r", encoding='utf8') as file:
 ## topics = [line.strip() for line in file]
 ##
 ## with open(topic2_file, "r", encoding='utf8') as file:
 ## topic2s = [line.strip() for line in file]

 numTopics = len(index_1)
 numTopic2s = len(index_2)

 # Read color log file
 with open(log_file, "r") as file:
 lines = file.readlines()

 lineNumber = 0
 # Save output to a file
 output_file = "output.txt"
 with open(output_file, "w") as file:
 print()
 for line in lines:
 parts = line.strip().split(",")
 if len(parts) >= 5:
 xcoor = int(parts[1])
 ycoor = int(parts[0])
 index_ = (gridWidth * ycoor) + xcoor
 character = parts[2]
 squareColor = parts[3]
 fontColor = parts[4]
 notes = parts[5]

 # Extract RGB values from squareColor
 red, green, blue = extract_rgb_values(squareColor)

 # Extract RGB values from fontColor
 fontRed, fontGreen, fontBlue = extract_rgb_values(fontColor)

 # Calculate the grid size
 grid_size = int(parts[0])

```

```

Select topics based on RGB values from squareColor
topicIndices = [(red + get_index(i, j, grid_size)) % numTopics for i in range(grid_size) for j in
range(grid_size)]

lineNumber += 1

with open(output_file, "a") as file:
 file.write(f"Compiling Line/s: {lineNumber}\n\n")
 file.write(f"Line({lineNumber})\n\n")
 file.write(f"0\tLineNumber: {lineNumber}\ttx-coordinate: {xcoor}\ty-coordinate: {ycoor}\t"
Character: {character}\tRGB: {squareColor}\tRGB: {fontColor}\t Note: {notes}\t GridDimensions:
{gridWidth}x{gridWidth}\n\n")
 #print(topicIndices)
 file.write(f"Line/s({lineNumber}) Output:\n\n")
 file.write(f"#define Line({lineNumber}){(\n\n")
 # Inside the main function
 file.write(f"\ti_0(squareColor){(\n\n")
 file.write(f"\t\t{character}\n")
 file.write(f"\t\t(R: {red}, G: {green}, B: {blue})\n")
 s_ratio = square_print_topic_indices(red, green, blue, file)
 #square_print_topic_indices(red, green, blue, topics, file) # Modify this line
 file.write("\t)\n")
 file.write("\t;\n\n")
 file.write(f"\ti_1(fontColor){(\n\n")
 file.write(f"\t\t{character}\n")
 file.write(f"\t\t(R: {fontRed}, G: {fontGreen}, B: {fontBlue})\n")
 f_ratio = font_print_topi_indices(fontRed, fontGreen, fontBlue, file)
 #font_print_topic_indices(fontRed, fontGreen, fontBlue, topic2s, file) # And this line
 file.write("\t)\n")
 file.write(f"\treturn squareRatio({s_ratio}), fontRatio({f_ratio}), sqaureIndex({index_})\n")
 #file.write(f"\treturn fontRatio({f_ratio})\n")
 file.write("}\n\n")

```

GO()

```

def create_menu():
 global control_frame
 def home():
 canvas.pack_forget()
 menubar.destroy()
 control_frame.pack()
 char_entry.pack_forget()
 char_label.pack_forget()
 #grid_size_label.pack_forget()
 #grid_size_entry.pack_forget()
 #font_size_label.pack_forget()
 #font_size_entry.pack_forget()

```

```

#adjust_button.pack_forget()
char_note.pack_forget()
char_note_label.pack_forget()

root.destroy()
switch = 1
setup_mode(switch)

Create a menubar
menubar = tk.Menu(root)

Create an options menu and add it to the menubar
options_menu = tk.Menu(menubar, tearoff=0)
#menubar.add_cascade(label="Options", menu=options_menu)

Create a dropdown menu and add it to the menubar
dropdown = tk.Menu(menubar, tearoff=0)
menubar.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"

Add commands to dropdown menu

Add checkbutton to options menu
dropdown.add_command(label="Toggle Manual/Random Colour", command=manual_colour)

dropdown.add_command(label="Refresh", command=refresh_canvas)
dropdown.add_command(label="Undo", command=undo_last)
dropdown.add_command(label="Save Session", command=save_session)
dropdown.add_command(label="Load Session", command=load_session)
dropdown.add_command(label="Save as Image", command=save_canvas)
dropdown.add_command(label="Compile Canvas", command=compileGOSmX)
#dropdown.add_command(label="Exit", command=exit_app)
#dropdown.add_command(label="Toggle Type/Click Mode", command=toggle_typing_mode)
dropdown.add_command(label="About Program", command=show_about)
dropdown.add_command(label="Go Home", command=home)
dropdown.add_command(label="Edit Grid Dimensions", command=adjust_grid_and_font)
dropdown.add_command(label="Typing Mode", command=show_typing_mode_menu)

Set the menubar
root.config(menu=menubar)

```

```

def create_menu_1():
 global control_frame
 def home1():
 canvas.pack_forget()
 menubar1.destroy()
 root.destroy()
 switch = 1

```

```
setup_mode(switch)

Create a menubar
menubar1 = tk.Menu(root)

Create an options menu and add it to the menubar
options_menu1 = tk.Menu(menubar1, tearoff=0)
#menubar.add_cascade(label="Options", menu=options_menu)

Create a dropdown menu and add it to the menubar
dropdown = tk.Menu(menubar1, tearoff=0)
menubar1.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"

Add commands to dropdown menu

Add checkbutton to options menu
dropdown.add_command(label="About Program", command=show_about)
dropdown.add_command(label="Change Mode", command=home1)

Set the menubar
root.config(menu=menubar1)

def manual_colour():
 # Toggle the value of color_mode_var
 current_value = color_mode_var.get()
 color_mode_var.set(not current_value)

"""

Create a tkinter window
root = tk.Tk()
root.title("General Operating System")
root.iconbitmap("logo.ico")
"""

def show_typing_mode_menu():
 global color_
 while True:
 print("1. Use default UTF-8 mapping")
 print("4. Exit")
 choice = input("Choose an option: ")

 if choice == "1":
 # Use default UTF-8 mapping
 print("Using default UTF-8 mapping.")
 color_ = input("Use Default Colour Scheme (1): ")
 if(color_ == '1'):
 # Initialize bg_color and font_color with default values at the global scope
```

```
 bg_color = "#FFFFFF" # Default white background color
 font_color = "#000000" # Default black font color
 break
 else:
 print("Invalid choice. Please try again.")

canvas.bind("<Key>", type_character)
canvas.focus_set()

def set_background_color():
 global color
 #bg_color = tk.colorchooser.askcolor()[1]
 color = input("Enter background color (#FFFFFF for example):")

def set_font_color():
 global font_color, font_size
 #font_color = tk.colorchooser.askcolor()[1]
 font_color = input("Enter background color (#000000 for example): ")
 font_size = input("Enter font_size: ")

def type_character(event):
 global current_row, current_col, col, font_size, font_color, color, canvas_width, square_size
 color = "#FFFFFF"
 font_color = "#000000"
 char = event.char
 if char: # Ignore non-character events
 try:

 draw_char1(canvas, char, current_row, current_col, font_size, color)
 current_col += 1
 if (current_col >= canvas_width/square_size):
 current_col = 0
 current_row += 1
 if(current_row >= canvas_width/square_size):
 current_col = 0
 current_row = 0
 except Exception as e:
 print("Something Went Wrong...")
 print(e)

def setup_mode(sw):
```

```
global mode_entry, mode_label, square_size, grid_size, char_count, last_drawn
global root, color_mode_var, logging, paused, canvas_width, canvas_height, canvas
global grid_size_label, grid_size_entry, font_size_label, font_size_entry, adjust_button
global save_Sbutton, load_button, char_label, char_entry, char_note_label, char_note, char_button
global refresh_button, undo_button, control_frame, submit_button, col, color_
global

current_row = 0
current_col = 0
color_ = 1

Define the size of the squares and the grid
square_size = 70
grid_size = 10
char_count = 0
last_drawn = []

window_height = square_size * grid_size + (square_size * 2)
window_width = square_size * grid_size

if(sw == 1):
 # Create a tkinter window
 root = tk.Tk()
 root.title("C22")
 #root.iconbitmap("logo.ico")
 sw = 0
 switch = 0

####

Get screen width and height
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

Calculate position of the window
position_top = int(screen_height / 2 - window_height / 2)
position_right = int(screen_width / 2 - window_width / 2)

Set the dimensions of the window and where it is placed
root.geometry(f'{window_width}x{window_height}+{position_right}+{position_top}')

####

Create a BooleanVar for the color mode and set it to False initially
color_mode_var = BooleanVar(value=False)

Setup logging
logging.basicConfig(filename='logging.txt', level=logging.INFO)
```

```

paused = tk.BooleanVar()
paused.set(False)

canvas_width = grid_size * square_size
canvas_height = grid_size * square_size
canvas = tk.Canvas(root, width=canvas_width, height=canvas_height)
col = canvas_width
canvas.pack()
canvas.bind("<Button-1>", on_canvas_click)

In the function where you create the canvas...
canvas.bind("<Key>", on_key_press)
canvas.focus_set()

Create input fields and buttons for grid and font size input
grid_size_label = tk.Label(root, text="Grid size:")
grid_size_entry = tk.Entry(root, width=5)
font_size_label = tk.Label(root, text="Font size:")
font_size_entry = tk.Entry(root, width=5)
adjust_button = tk.Button(root, text='Adjust Grid & Font', command=adjust_grid_and_font)

Create the save and load buttons
save_Sbutton = tk.Button(root, text='Save Session', command=save_session, bg="white", padx=5,
pady=0)
load_button = tk.Button(root, text='Load Session', command=load_session, bg="white", padx=5, pady=0)

Create input fields and buttons for mode and character input
mode_label = tk.Label(root, text="Enter ... 'IDE' for an\nInteractive Development Environment\nfor
Learning anything!")
mode_entry = tk.Entry(root)
submit_button = tk.Button(root, text='LEARN', command=submit_mode)

"""

filename = simpledialog.askstring("Save Canvas", "Enter filename:")
if filename:
"""

char_label = tk.Label(root, text="Enter symbol/s to draw: ")
char_entry = tk.Entry(root, width=12)
char_note_label = tk.Label(root, text="Enter Note: ")
char_note = tk.Entry(root, width=12)
#char_button = tk.Button(root, text='Draw Character', command=draw_char)

refresh_button = tk.Button(root, text='Refresh', command=refresh_canvas, bg="white", padx=5, pady=0)
undo_button = tk.Button(root, text='Undo', command=undo_last, bg="white", padx=5, pady=0)

control_frame = tk.Frame(root)

```

```
mode_label.pack()
mode_entry.pack()
submit_button.pack()
create_menu(root)

Run the tkinter main loop
root.mainloop()
```

```
global switch
switch = 1
```

```
setup_mode(switch)
```

----- Content of BlockDesignCraft.py-----

```
import tkinter as tk
from tkinter import simpledialog
from tkinter import colorchooser

class BlockDiagramDesigner(tk.Tk):
 def __init__(self):
 super().__init__()

 self.title("Block Diagram Designer")
 self.geometry("800x600")

 print("Welcome to the Block Diagram Designer!")
 print("Instructions:")
 print("Left-click to add a block.")
 print("Double-click on a block to customize its color and label.")
 print("Press and hold the right mouse button, drag, and release to draw an arrow.")
 print("Press Ctrl+Z to undo the last action.")

 self.canvas = tk.Canvas(self, bg="white")
 self.canvas.pack(fill=tk.BOTH, expand=True)
 self.shift_pressed = False
 self.grid_size = 20 # Size of each grid cell
 self.block_counter = 0 # Counter to keep track of block IDs
 self.draw_grid()

 self.canvas.bind("<Button-1>", self.add_block)
 self.canvas.bind("<ButtonPress-3>", self.start_arrow)
 self.canvas.bind("<ButtonRelease-3>", self.end_arrow)
 self.canvas.bind("<B3-Motion>", self.drag_arrow)
 self.canvas.bind("<Double-Button-1>", self.double_click_block) # Only keep this for double click
 self.canvas.bind("<Shift-KeyPress>", self.shift_pressed_event)
 self.canvas.bind("<Shift-KeyRelease>", self.shift_released_event)

 self.undo_stack = [] # List to store actions for undo
```

```

self.bind("<Control-z>", self.undo)

self.labels = {} # To keep track of block labels
self.lines = [] # To store lines
self.arrows = [] # To store arrows

self.blocks = {}
self.arrows = []
self.action_stack = [] # For undo functionality
self.temp_arrow = None # For drag-to-draw arrow

Variables for arrow drawing
self.start_position = None
self.end_position = None
self.drawing_arrow = False

def draw_grid(self):
 for i in range(0, 800, self.grid_size):
 self.canvas.create_line([(i, 0), (i, 600)], tag='grid_line', fill="#d3d3d3")

 for i in range(0, 600, self.grid_size):
 self.canvas.create_line([(0, i), (800, i)], tag='grid_line', fill="#d3d3d3")

def snap_to_grid(self, x, y):
 return round(x / self.grid_size) * self.grid_size, round(y / self.grid_size) * self.grid_size

def shift_pressed_event(self, event):
 self.shift_pressed = True

def shift_released_event(self, event):
 self.shift_pressed = False

def add_block(self, event):
 x, y = self.snap_to_grid(event.x, event.y)
 block_width = self.grid_size
 block_height = self.grid_size
 x1, y1 = x - block_width // 2, y - block_height // 2
 x2, y2 = x + block_width // 2, y + block_height // 2
 block = self.canvas.create_rectangle(x1, y1, x2, y2, fill='gold', tags="block")
 self.blocks[block] = {'coords': [x1, y1, x2, y2], 'fill': 'gold', 'label': None, 'font_color': None}

 # Store the block information with its ID as the key
 self.blocks[block] = {'id': block, 'coords': [x1, y1, x2, y2], 'fill': 'gold', 'label': None, 'font_color': None}
 self.block_counter += 1 # Increment the block counter
 self.undo_stack.append(("add_block", block))

def change_block_color(self, event):
 # Find closest block
 x, y = self.snap_to_grid(event.x, event.y)
 closest_block = self.canvas.find_closest(x, y, halo=20, start=None)

```

```

if closest_block:
 block = closest_block[0]
 new_color = colorchooser.askcolor()[1]
 if new_color:
 self.canvas.itemconfig(block, fill=new_color)

New code to change label color
if block in self.labels:
 label = self.labels[block]
 new_font_color = colorchooser.askcolor()[1]
 if new_font_color:
 self.canvas.itemconfig(label, fill=new_font_color)

def double_click_block(self, event):
 clicked_items = self.canvas.find_withtag(tk.CURRENT)
 for item in clicked_items:
 if "block" in self.canvas.gettags(item):
 # Code to show dialog boxes for customizing the block
 block_fill = colorchooser.askcolor(title="Choose block color")[1]
 self.canvas.itemconfig(item, fill=block_fill)

 label_font_color = colorchooser.askcolor(title="Choose label font color")[1]

 label_text = simpledialog.askstring("Input", "Enter the label:")

 self.canvas.create_text((self.blocks[item]['coords'][0] + self.blocks[item]['coords'][2]) // 2,
 (self.blocks[item]['coords'][1] + self.blocks[item]['coords'][3]) // 2,
 text=label_text, fill=label_font_color)

def label_block(self, event):
 item = self.canvas.find_closest(event.x, event.y)
 if 'block' in self.canvas.gettags(item):
 label = simpledialog.askstring("Input", "Enter block label:")
 if label:
 label = label.encode('utf-8').decode('utf-8')
 x, y = self.blocks[item[0]]
 self.canvas.create_text(x, y, text=label)

def undo(self, event=None):
 if self.undo_stack:
 action, item = self.undo_stack.pop()
 if action == "add_block":
 # Delete the last added block
 self.canvas.delete(item)
 elif action == "add_arrow" or action == "add_line":
 # Delete the last added arrow or line
 self.canvas.delete(item)
 if action == "add_arrow" and item in self.arrows:
 self.arrows.remove(item)
 elif action == "add_line" and item in self.lines:

```

```

self.lines.remove(item)
elif action == "add_label":
 # Delete the last added label
 self.canvas.delete(item)

def start_arrow(self, event):
 x, y = self.snap_to_grid(event.x, event.y)
 self.start_position = [x, y]
 self.drawing_arrow = True
 self.end_position = [x, y] # Start with the end position at the cursor

 # Create the temporary arrow (temp_arrow)
 if self.drawing_arrow:
 self.temp_arrow = self.canvas.create_line(x, y, x, y, arrow=tk.LAST, tags="temp_arrow")

def drag_arrow(self, event):
 if self.drawing_arrow:
 x, y = self.snap_to_grid(event.x, event.y)
 self.end_position = [x, y]
 if self.temp_arrow:
 self.canvas.coords(self.temp_arrow, self.start_position[0], self.start_position[1], x, y)

def end_arrow(self, event):
 #global terminating_coords
 if self.drawing_arrow:
 x, y = self.snap_to_grid(event.x, event.y)
 closest_block = self.canvas.find_closest(x, y, halo=20, start=None)
 if closest_block:
 block = closest_block[0]

 # Check if the block ID exists in self.blocks
 if block in self.blocks:
 x1, y1, x2, y2 = self.blocks[block]['coords']
 cx, cy = (x1 + x2) / 2, (y1 + y2) / 2 # center coordinates

 # Determine the end position based on user input
 if self.end_position[0] == x and self.end_position[1] == y:
 terminating_coords = self.end_position # User clicked to set the end coordinates
 else:
 # User specified a direction (top, right, bottom, left)
 direction = self.end_position
 if direction == "top":
 terminating_coords = [cx, y1]
 elif direction == "right":
 terminating_coords = [x2, cy]
 elif direction == "bottom":
 terminating_coords = [cx, y2]
 elif direction == "left":

```

```

 terminating_coords = [x1, cy]
 else:
 terminating_coords = self.end_position

 # Draw an arrow and store it in self.arrows
 arrow = self.canvas.create_line(self.start_position[0], self.start_position[1],
terminating_coords[0], terminating_coords[1], arrow=tk.LAST, tags="arrow")
 self.action_stack.append(("add_arrow", arrow))
 self.arrows.append(arrow) # Append the arrow to the arrows list
 #self.arrows.append(arrow)

 # Reset the arrow drawing state
 self.drawing_arrow = False

 # Delete the temporary arrow if it exists
 #if self.temp_arrow:
 # self.canvas.delete(self.temp_arrow)

if __name__ == "__main__":
 app = BlockDiagramDesigner()
 app.mainloop()

```

----- Content of BooleanWords.py-----

```

import itertools

def is_valid_expression(expression):
 last_was_operator = True # Start with True to disallow leading operators
 for token in expression:
 if token in ['AND', 'OR', 'XOR']:
 if last_was_operator:
 return False
 last_was_operator = True
 elif token == 'NOT':
 if not last_was_operator:
 return False
 last_was_operator = True
 else:
 if not last_was_operator:
 return False
 last_was_operator = False
 return not last_was_operator # Expression can't end with an operator

```

```

Function to create B# Boolean statements
def create_boolean_statements(array, mapping, num_placeholders):
 statements = []
 if num_placeholders % 2 == 0:
 print("Number of placeholders must be odd.")

```

```

return

placeholders = ["_" for _ in range(num_placeholders)]

combinations = list(itertools.product(array, repeat=num_placeholders))

for combination in combinations:
 statement = placeholders.copy()
 for i, val in enumerate(combination):
 statement[i] = mapping[val]
 if is_valid_expression(statement):
 statements.append(" ".join(statement))

return statements

Function to load senses from an external file
def load_senses_from_file(filename):
 mapping_ = {}
 with open(filename, 'r') as file:
 for line in file:
 index, label = line.strip().split(" ")
 mapping_[int(index)] = label
 return mapping_, len(mapping_)

Ask the user for the size of the array
ask = input("Type defaults to use default settings else type continue or custom to load settings from a file: ")
if(ask == 'continue'):
 size = int(input("Enter the size of your array: "))

array = []

Dictionary to hold default settings
if(ask == 'defaults'):
 size = 9
 mapping = {
 1: "Sound",
 3: "Taste",
 4: "Touch",
 5: "Light",
 6: "Smell",
 7: "AND",
 8: "OR",
 9: "NOT",
 10: "XOR"
 }
 array = list(mapping.keys())[:size]

```

```
elif(ask == 'continue'):
 #mapping = {}

 for i in range(size):
 num = int(input(f"Enter element {i+1}: "))
 array.append(num)
 label = input(f"Enter the label for {num}: ")
 mapping[num] = label

elif(ask == 'custom'):
 filename = input("Enter 'filename.txt' to load custom settings: ")
 mapping, size = load_senses_from_file(filename)
 array = list(mapping.keys())[:size]

else:
 print("typo ...")

num_placeholders = int(input("Enter the number of placeholders (must be odd): "))

statements = create_boolean_statements(array, mapping, num_placeholders)
with open('words.txt', 'w') as f:
 for statement in statements:
 #print(statement)
 f.write(f'{statement}\n')

print("Boolean statements have been saved in 'words.txt'")
```



```

 for (row=0; row < nbr_comb; row++){
 id++;
 //fprintf(p,"n\nF%d\nn\n", id);
 fprintf(p,"%d ", id);
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);
 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 fprintf(p,"%c", a[rin]);
 }
 fprintf(p,"\n");
 }
 }
 if(cheque == 0){
 for (row=0; row < nbr_comb; row++){
 id++;
 //fprintf(p,"n\nF%d\nn\n", id);
 fprintf(p,"%d ", id);
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 cell = (row/rdiv) % (k+1);
 fprintf(p,"%c", a[cell]);
 }
 fprintf(p,"\n");
 }
 }
 //fprintf(p,"n\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
 fclose(p);
 //end of adaptation
 return 0;
}

```

----- Content of calculator\_app.py-----

```

import tkinter as tk
from tkinter import ttk

def calc_action(button, entry):
 if button == 'C':
 entry.delete(0, tk.END)
 elif button == '=':
 try:
 entry.insert(tk.END, " = " + str(eval(entry.get())))
 except:
 entry.delete(0, tk.END)
 entry.insert(tk.END, "Error")
 else:
 entry.insert(tk.END, button)

def open_calculator(root):

```

```

calculator = tk.Toplevel(root)
calculator.title("Calculator")

entry = ttk.Entry(calculator)
entry.grid(row=0, column=0, columnspan=4)

buttons = [
 '7', '8', '9', '+',
 '4', '5', '6', '-',
 '1', '2', '3', '*',
 'C', '0', '=', '/'
]

row_val = 1
col_val = 0
for button in buttons:
 ttk.Button(calculator, text=button, command=lambda b=button: calc_action(b,
entry)).grid(row=row_val, column=col_val)
 col_val += 1
 if col_val > 3:
 col_val = 0
 row_val += 1

if __name__ == "__main__":
 root = tk.Tk()
 root.title("Main Window")
 ttk.Button(root, text="Open Calculator", command=lambda: open_calculator(root)).pack()
 root.mainloop()

```

----- Content of canvas2.py-----

```

import tkinter as tk
from tkinter.colorchooser import askcolor

class DigitalInkApp:
 def __init__(self, root):
 self.root = root
 self.root.title("Digital Ink Application")

 self.canvas_bg = "#ffffff"
 self.ink_color = "#000000"

 self.canvas = tk.Canvas(root, bg=self.canvas_bg, width=800, height=600)
 self.canvas.pack(expand=tk.YES, fill=tk.BOTH)

 self.canvas.bind("<B1-Motion>", self.paint)

menu = tk.Menu(root)
root.config(menu=menu)

```

```

colorMenu = tk.Menu(menu)
menu.add_cascade(label="Colors", menu=colorMenu)

colorMenu.add_command(label="Canvas Background", command=self.change_canvas_bg)
colorMenu.add_command(label="Ink Color", command=self.change_ink_color)

editMenu = tk.Menu(menu)
menu.add_cascade(label="Edit", menu=editMenu)
editMenu.add_command(label="Clear Canvas", command=self.clear_canvas)

def paint(self, event):
 x1, y1 = (event.x- 1), (event.y- 1)
 x2, y2 = (event.x + 1), (event.y + 1)
 self.canvas.create_oval(x1, y1, x2, y2, fill=self.ink_color, outline=self.ink_color, width=2)

def change_canvas_bg(self):
 color = askcolor()[1]
 if color:
 self.canvas_bg = color
 self.canvas.config(bg=self.canvas_bg)

def change_ink_color(self):
 color = askcolor()[1]
 if color:
 self.ink_color = color

def clear_canvas(self):
 self.canvas.delete("all")

if __name__ == "__main__":
 root = tk.Tk()
 app = DigitalInkApp(root)
 root.mainloop()

```

----- Content of canvas3.py-----

```

import tkinter as tk
from tkinter.colorchooser import askcolor
from PIL import Image, ImageDraw

class DigitalInkApp:
 def __init__(self, root):
 self.root = root
 self.root.title("Digital Ink Application")

 self.canvas_bg = "#ffffff"
 self.ink_color = "#000000"
 self.width = 800
 self.height = 600

```

```
self.canvas = tk.Canvas(root, bg=self.canvas_bg, width=self.width, height=self.height)
self.canvas.pack(expand=tk.YES, fill=tk.BOTH)

self.canvas.bind("<B1-Motion>", self.paint)

menu = tk.Menu(root)
root.config(menu=menu)

colorMenu = tk.Menu(menu)
menu.add_cascade(label="Colors", menu=colorMenu)

colorMenu.add_command(label="Canvas Background", command=self.change_canvas_bg)
colorMenu.add_command(label="Ink Color", command=self.change_ink_color)

editMenu = tk.Menu(menu)
menu.add_cascade(label="Edit", menu=editMenu)
editMenu.add_command(label="Clear Canvas", command=self.clear_canvas)

sizeMenu = tk.Menu(menu)
menu.add_cascade(label="Size", menu=sizeMenu)
sizeMenu.add_command(label="Resize Window", command=self.resize_window)

fileMenu = tk.Menu(menu)
menu.add_cascade(label="File", menu=fileMenu)
fileMenu.add_command(label="Save", command=self.save_canvas)

def paint(self, event):
 x1, y1 = (event.x- 1), (event.y- 1)
 x2, y2 = (event.x + 1), (event.y + 1)
 self.canvas.create_oval(x1, y1, x2, y2, fill=self.ink_color, outline=self.ink_color, width=1)

def change_canvas_bg(self):
 color = askcolor()[1]
 if color:
 self.canvas_bg = color
 self.canvas.config(bg=self.canvas_bg)

def change_ink_color(self):
 color = askcolor()[1]
 if color:
 self.ink_color = color

def clear_canvas(self):
 self.canvas.delete("all")

def resize_window(self):
 new_width = tk.simpledialog.askinteger("Width", "Enter new width:", initialvalue=self.width)
 new_height = tk.simpledialog.askinteger("Height", "Enter new height:", initialvalue=self.height)
 if new_width and new_height:
 self.width, self.height = new_width, new_height
```

```

 self.canvas.config(width=self.width, height=self.height)

def save_canvas(self):
 # Specify the full path where 'canvas.ps' will be saved
 ps_file_path = "D:\\\\learn\\\\canvas.ps"

 # Save the canvas as a PostScript file
 self.canvas.postscript(file=ps_file_path, colormode='color')

 # Open the saved PostScript file and convert it to an image
 img = Image.open(ps_file_path)
 img.save("canvas.png", "png")

if __name__ == "__main__":
 import tkinter.simpledialog # Importing here to avoid error if not used elsewhere
 root = tk.Tk()
 app = DigitalInkApp(root)
 root.mainloop()

----- Content of charSF3.py-----
import os
import re

def extract_characters_from_file(file_path):
 with open(file_path, 'r', encoding='utf-8') as file:
 return re.findall(r'\bU\+\w{4,5}\b', file.read())

def write_characters_to_file(characters, output_file_path):
 with open(output_file_path, 'w', encoding='utf-8') as file:
 for char in characters:
 file.write(chr(int(char[2:], 16)) + '\n')

def main():
 directory = '.' # Specify the directory where your files are located
 output_file_path = 'charSF3.txt'
 all_characters = set()
 files_to_process = ['Unihan_Readings', 'Unihan_IRGSources', 'Unihan_Variants', 'Unihan_NumericValues',
 'Unihan_RadicalStrokeCounts.txt', 'Unihan_DictionaryLikeData'] # Add more file names as needed ...

 for filename in os.listdir(directory):
 for file_prefix in files_to_process:
 if filename.startswith(file_prefix) and filename.endswith('.txt'):
 file_path = os.path.join(directory, filename)
 characters = extract_characters_from_file(file_path)
 all_characters.update(characters)

 write_characters_to_file(all_characters, output_file_path)
 print("Characters extracted and saved to", output_file_path)

```

```
if __name__ == "__main__":
 main()
```

----- Content of CMD-Terminal.py-----

```
import tkinter as tk
from tkinter import scrolledtext
import subprocess

class CustomTerminal(tk.Tk):
 def __init__(self):
 super().__init__()

 self.title("Python Terminal")
 self.geometry("800x600")

 self.text_widget = scrolledtext.ScrolledText(self, wrap='word', height=30, width=80)
 self.text_widget.pack()

 self.command_entry = tk.Entry(self, width=80)
 self.command_entry.pack()
 self.command_entry.bind('<Return>', self.execute_command)

 def execute_command(self, event):
 command = self.command_entry.get()
 self.command_entry.delete(0, tk.END)
 self.text_widget.insert(tk.END, f'{command}\n')

 # Specify that stdin is also a pipe, so we can write to it
 process = subprocess.Popen(command, shell=True, stdin=subprocess.PIPE, stdout=subprocess.PIPE,
 stderr=subprocess.PIPE, text=True)

 # Assuming some_input is the input you want to provide to the executable
 some_input = "your_input_here\n"
 process.stdin.write(some_input)
 process.stdin.flush() # Ensure the input is sent to the subprocess

 stdout, stderr = process.communicate()
 self.text_widget.insert(tk.END, stdout + stderr)

if __name__ == "__main__":
 terminal = CustomTerminal()
 terminal.mainloop()
```

----- Content of ComputationalChemistry.py-----

```
import tkinter as tk
from tkinter import ttk
```

```

print("Photosynthesis: Time = 30 minutes\nCellular Respiration: Time = 45 minutes\nCombustion of Methane: Time = 5 minutes\nHydrolysis of Sodium Chloride (Salt): Time = 15 minutes\nFermentation of Glucose: Time = 60 minutes\nOxidation of Iron (Rust Formation): Time = 120 minutes\nNeutralization of Hydrochloric Acid with Sodium Hydroxide: Time = 10 minutes\nPolymerization of Ethylene (Plastic Formation): Time = 90 minutes\nDecomposition of Hydrogen Peroxide: Time = 20 minutes\nPrecipitation of Barium Sulfate: Time = 25 minutes\nPhotosensitive Reaction in Photography: Time = 180 minutes\nEsterification Reaction (Production of Esters): Time = 75 minutes")

def create_table(op_type, n, reaction_time1, reaction_time2):
 for widget in table_frame.winfo_children():
 widget.destroy()

 for i in range(n+1):
 for j in range(n+1):
 if i == 0 and j == 0:
 text = f"ReactionTime1\nReactionTime2"
 elif i == 0:
 text = f"{j-1}"
 elif j == 0:
 text = f"{i-1}"
 else:
 if op_type == "Addition":
 result = (i-1 + j-1) % n
 elif op_type == "Multiplication":
 result = (i-1) * (j-1) % n
 text = f"{result} (Least Residue)"

 label = tk.Label(table_frame, text=text, width=15, height=2, borderwidth=1, relief="solid")
 label.grid(row=i, column=j)

 # Display the result for the input reaction times
 if op_type == "Addition":
 result_text = f"{{reaction_time1} + {reaction_time2}} mod {n} = {{reaction_time1 + reaction_time2} % n} (Least Residue)"
 elif op_type == "Multiplication":
 result_text = f"{{reaction_time1} * {reaction_time2}} mod {n} = {{reaction_time1 * reaction_time2} % n} (Least Residue)"
 result_label.config(text=result_text)

root = tk.Tk()
root.title("Modular Arithmetic Table- Computational Biology")

... (Rest of the script remains the same)

control_frame = tk.Frame(root)
control_frame.pack(side=tk.TOP, padx=10, pady=10)

```

```
table_frame = tk.Frame(root)
table_frame.pack(side=tk.BOTTOM, padx=10, pady=10)

operation_label = tk.Label(control_frame, text="Operation:")
operation_label.pack(side=tk.LEFT)

operation_combobox = ttk.Combobox(control_frame, values=["Addition", "Multiplication"],
state="readonly")
operation_combobox.pack(side=tk.LEFT)
operation_combobox.current(0)

n_label = tk.Label(control_frame, text="Modulo (n ≤ 12):")
n_label.pack(side=tk.LEFT)

n_entry = tk.Entry(control_frame, width=5)
n_entry.pack(side=tk.LEFT)
n_entry.insert(0, "3")

rt1_label = tk.Label(control_frame, text="Reaction Time 1:")
rt1_label.pack(side=tk.LEFT)

rt1_entry = tk.Entry(control_frame, width=5)
rt1_entry.pack(side=tk.LEFT)
rt1_entry.insert(0, "1")

rt2_label = tk.Label(control_frame, text="Reaction Time 2:")
rt2_label.pack(side=tk.LEFT)

rt2_entry = tk.Entry(control_frame, width=5)
rt2_entry.pack(side=tk.LEFT)
rt2_entry.insert(0, "1")

result_label = tk.Label(root, text="")
result_label.pack(side=tk.TOP, pady=5)

def update_table():
 op_type = operation_combobox.get()
 n = int(n_entry.get())
 reaction_time1 = int(rt1_entry.get())
 reaction_time2 = int(rt2_entry.get())
 create_table(op_type, n, reaction_time1, reaction_time2)

update_button = tk.Button(control_frame, text="Update Table", command=update_table)
update_button.pack(side=tk.LEFT)

update_table()

root.mainloop()
```

```
----- Content of Comsz_Engine.cpp-----
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <map>
#include <stdlib.h>
#include "database.h"

using namespace std;

map<string, string> database;

// Function to add a new record to the database
void addRecord(string key, string value)
{
 cout << "Adding record..." << endl;
 database[key] = value;
 cout << "Record added successfully!" << endl;
}

// Function to delete a record from the database
void deleteRecord(string key)
{
 cout << "Deleting record..." << endl;
 database.erase(key);
 cout << "Record deleted successfully!" << endl;
}

// Function to search for a record in the database
void searchRecord(string key)
{
 cout << "Searching record..." << endl;
 if (database.find(key) != database.end())
 {
 cout << "Record found!" << endl;
 cout << "Value: " << database[key] << endl;
 }
 else
 {
 cout << "Record not found!" << endl;
 }
}

// Function to update a record in the database
void updateRecord(string key, string value)
{
 cout << "Updating record..." << endl;
 if (database.find(key) != database.end())

```

```
{
 database[key] = value;
 cout << "Record updated successfully!" << endl;
}
else
{
 cout << "Record not found!" << endl;
}
}

// Function to print all records in the database
void printAllRecords()
{
 cout << "Printing all records..." << endl;
 for (auto& record : database)
 {
 cout << record.first << ":" << record.second << endl;
 }
}

// Function to save the database to a file
void saveDatabase()
{
 ofstream file("database.txt");
 for (auto& record : database)
 {
 file << record.first << "," << record.second << endl;
 }
 file.close();
}

// Function to load the database from a file
void loadDatabase()
{
 ifstream file("database.txt");
 string line;
 while (getline(file, line))
 {
 stringstream sstream(line);
 string key, value;
 getline(sstrm, key, ',');
 getline(sstrm, value, ',');
 database[key] = value;
 }
 file.close();
}

int main()
{
 // Load the database from the file
```

```
loadDatabase();
int mode;

int xin;
cout << "\n\nContinue? 1 for yes, 0 for no: ";
cin >> xin;
while (xin == 1) {
 cout << "\n\nEnter mode: \n1 (Add record)\n2 (Search for record)\n3 (Update existing record)\n4
(Delete a record)\n5 (Print all records)\n6 (Save the database to file (database.txt))-- ";
 cin >> mode;
 if (mode == 1) {
 // Add a new record
 string jhin, valued;
 cout << "\nEnter entry ID: ";
 cin >> jhin;

 cout << "\nEnter entry value: ";
 cin >> valued;

 addRecord(jhin, valued);

 }
 else if (mode == 2) {
 // Search for a record
 string mode2;
 cout << "\nEnter record ID: ";
 cin >> mode2;
 searchRecord(mode2);
 }
 else if (mode == 3) {
 // Update an existing record
 string mode3a, mode3b;
 cout << "\nEnter record ID: ";
 cin >> mode3a;
 cout << "\nEnter new record value: ";
 cin >> mode3b;
 updateRecord(mode3a, mode3b);
 }
 else if (mode == 4) {
 // Delete a record
 string mode4;
 cout << "\nEnter record ID to be deleted: ";
 cin >> mode4;
 deleteRecord(mode4);
 }
 else if (mode == 5) {
 // Print all records
 printAllRecords();
 }
}
```

```

 else {
 // Save the database to the file
 saveDatabase();
 }
 cout << "\n\nContinue? 1 for yes, 0 for no: ";
 cin >> xin;
}

system("Comsz_OperatingSystem.exe");

return 0;

}

```

----- Content of Comsz\_OperatingSystem.c-----

```

//Written with the aid of Chat GPT-3
//include necessary libraries
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
 char input[100]; //stores the user input
 char command[100]; //stores the command
 char argument[100]; //stores the argument
 int quit = 0; //boolean for exiting the system

 printf("Welcome to my simple text-based operating system!\n");
 printf("Type 'help' for a list of commands\n");

 //run the system until the user quits
 while(quit != 1)
 {
 //get user input
 printf("\n>");
 fgets(input, sizeof(input), stdin);
 sscanf(input, "%s %s", command, argument); //separate the command and argument

 //execute the command
 if (strcmp(command, "help") == 0)
 {
 printf("Available commands:\n");
 printf("help: Prints a list of available commands\n");
 printf("echo: Prints the given argument\n");
 printf("startd: runs file of the given filename/ argument.\n");
 printf("quit: Quits the system\n");
 }
 }
}

```

```

else if (strcmp(command, "echo") == 0)
{
 printf("%s\n", argument);
}
else if (strcmp(command, "quit") == 0)
{
 printf("Goodbye!\n");
 quit = 1;
}
else if(strcmp(command, "startd") == 0){
 system(argument);
}
else
{
 printf("Invalid command.\n");
}
}

return 0;
}

```

----- Content of ConsolePrint.cpp-----

```

#include <iostream>
#include <fstream>
#include <string>

int main() {
 std::string filename;
 std::cout << "Enter the filename: ";
 std::getline(std::cin, filename);

 std::ifstream inputFile(filename);
 if (!inputFile.is_open()) {
 std::cerr << "Error: unable to open file \\" " << filename << "\\n";
 return 1;
 }

 std::string line;
 int lineNumber = 1;
 while (std::getline(inputFile, line)) {
 std::cout << lineNumber << ":" << line << std::endl;
 ++lineNumber;
 }

 inputFile.close();
 return 0;
}

```

```
----- Content of Control.py-----
import pygame
import importlib
import requests
import os
import subprocess
import webbrowser
import csv
from bs4 import BeautifulSoup
import datetime
import sys
import math
import random
import json
import re

#####import pywebbrowser
#create a custom event type
MY_CUSTOM_EVENT = pygame.USEREVENT + 1

Initialize pygame
pygame.init()

Initialize joystick module
pygame.joystick.init()

Check if any joysticks are connected
if pygame.joystick.get_count() > 0:
 # Get the first joystick
 joystick = pygame.joystick.Joystick(0)
 # Initialize the joystick
 joystick.init()

Set screen size
#enter = int(input("Enter the dimension--> options 422, and multiples of 422"))
enter = 422
screen_width = enter
screen_height = enter

bls = 52
screen = pygame.display.set_mode((screen_width, screen_height))
#screen = pygame.display.set_mode((screen_width, screen_height), pygame.RESIZABLE)
#####
#####

#Create Commands
bus_ = 255
amount_ = (bus_ + 1) * (bus_ + 1)
line_number_ = 0
```

```

create_f = open("Command_Template.txt", "w")
while(line_number_ < amount_):
 #i = log10(line_number)
 create_f.write("print(\"[Empty Command Slot] (Change using a text-editor to Update this slot in\nCommand_Template.txt\") #remembering to rename Command_Template.txt\\n\")\n")
 line_number_ = line_number_ + 1

create_f.close()

#LoadCommands
xin = 0
filnam = "Command_Template.txt"
with open(filnam, 'r') as file:
 commands = file.readlines()
 xin = xin + 1

star = [0]

for i, command in enumerate(commands):
 commands[i] = command.strip()
 #cmd = command.split(",")
 star.append(command)
del star[0]

#####
#####

#LoadCoordinates
xin2 = 0
with open('Coordinates_Python.txt', 'r') as file2:
 commands2 = file2.readlines()
 xin2 = xin2 + 1

starsx = [0]
starsy = [0]

for i2, command2 in enumerate(commands2):
 commands2[i2] = command2.strip()
 cmd2 = command2.split(",")
 starsx.append(int(cmd2[1]))
 varys = int(cmd2[2].strip())
 starsy.append(varys)

del starsx[0]
del starsy[0]

#####

```

```

for i3, commandy in starsy:
 new = starsy[i3].strip()
 starsy[i3] = int(new)
 """
 print(starsx)
 print(starsy)
 #####
 # Load images
 block_images = []
 for i in range(256):
 block_images.append(pygame.image.load(f"block{i}.png"))

 # Maze layout
 maze_layout = [
 [0, 1, 2, 3],
 [4, 5, 6, 7],
 [8, 9, 10, 11],
 [12, 13, 14, 15]
]

 #Function to check if a position is inside the maze
 def is_inside_maze(x, y, maze_width, maze_height, bls):
 return 0 <= x < maze_width * bls and 0 <= y < maze_height * bls

 # Load player images
 player_image = pygame.image.load("player.png")
 player_image2 = pygame.image.load("player2.png")

 #scaled_block_images = [pygame.transform.scale(img, (int(img.get_width() * scale_factor),
 int(img.get_height() * scale_factor))) for img in block_images]
 #scaled_player_image = pygame.transform.scale(player_image, (int(player_image.get_width() *
 scale_factor), int(player_image.get_height() * scale_factor)))

 # Set player position
 player_x = 0
 player_y = 0
 con_program = int(input("Continue? 1 [yes], 0 [no]: "))
 if(con_program == 1):
 print("Ok ...")
 if(con_program == 0):
 print("Exiting ...")
 exit()
 if(con_program != 0 and con_program != 1):
 print("Error")

```

```
 exit()
Set initial input type
typei = "k"
print("Switch Command Matrix [s],\n mouse [m],\nkeyboard arrows [k],\nD-Pad (Xbox One Controller)
[d]\nPress [SPACEBAR] to call a command from Commands_Template.txt (Commands can be changed using
a text-editor) ... ")

some input variable
input_variable = 0
add the custom event to the event queue
pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": input_variable}))
#etch loop
etch = 0

#maze width
maze_width = 16
maze_height = 16
Main game loop
running = True
#new screen size settings
##screen_width, screen_height = pygame.display.get_surface().get_size()
scale_factor_x = screen_width / maze_width
##scale_factor_y = screen_height / maze_height
##scale_factor = min(scale_factor_x, scale_factor_y)
scale_factor = scale_factor_x

bls = int(bls * 0.5)

scaled_block_images = []
for ni in range(256):
 s_i = pygame.transform.scale(block_images[ni], (bls, bls))
 block_images[ni] = s_i

while running:
 for event in pygame.event.get():
 if event.type == pygame.QUIT:
 running = False
#####
 elif event.type == pygame.KEYDOWN:
 # Check for key press event
 if event.key == pygame.K_m and typei != "m":
 # Update input type
 typei = "m"
 elif event.key == pygame.K_k and typei != "k":
 # Update input type
 typei = "k"
 elif event.key == pygame.K_s and typei != "s":
 # Update input type
 typei = "s"
 elif event.key == pygame.K_d and typei != "d":
```

```

Update input type
typei = "d"
elif event.type == pygame.KEYDOWN:
 if event.key == pygame.K_SPACE:
 #update input_type
 typei = "n"
 #check for status
 enter_cma = int(input("Enter command? 1[yes], 0 [no]: "))
 """
 # user input variable
 if(enter_cma == 0):
 typei = "k"
 player_x = 0
 player_y = 0
 etch = 1
 """
 if(enter_cma == 0):
 typei = "k"
 player_x = 0
 player_y = 0
 if(enter_cma == 1):
 input_variable = int(input("Enter command number ID (0 to 65535): "))
 # add the custom event to the event queue
 pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": input_variable}))
 #pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": cma}))
 """
 elif event.type == pygame.MOUSEBUTTONUP:
 # Get mouse position
 mouse_x, mouse_y = pygame.mouse.get_pos()

 # Check if the clicked position is inside the maze
 if is_inside_maze(mouse_x, mouse_y, len(maze_layout[0]), len(maze_layout), bls):
 # Calculate the clicked cell
 cell_x = mouse_x // bls
 cell_y = mouse_y // bls

 # Execute the desired action (e.g., print a message with the image ID)
 print(f"Hello, world! Image ID: {maze_layout[cell_y][cell_x]}")
 """
check for custom event type
for event in pygame.event.get(MY_CUSTOM_EVENT):
 # access the input variable from the event object
 cma = event.input_variable
 # do something with the input variable
 #pass...
 if(typei == 'n'):

 #enter_cma = int(input("Enter command? 1[yes], 0 [no]: "))

```

```

if(enter_cma == 1 and enter_cma != 0):
 player_x = 0
 player_y = 0

 # Clear screen
 screen.fill((255, 255, 255))
 # Update display
 #pygame.display.update()

#cma = int(input("Enter command number ID (0 to 65535): "))
try:
 exec(star[cma], globals())
except:
 print("Try again, there was an error")
#update command index pointer
bus = 255 + 1
if(cma <= 255):
 cmd_index = 0
 command_point = cma
elif(cma > 255):
 cmd_index = cma//bus
 command_point = cma
for tip in range(0, cmd_index):
 command_point = command_point - (bus)
#print(command_point)

prompt = "Executing Command " + str(cma) + " from index: " + str(cmd_index)
print(prompt)
#print(starsx)

player_x = starsy[cmd_index]
player_y = starsx[cmd_index]
command_point_x = starsy[command_point]
command_point_y = starsx[command_point]

Render player image at new position-> screen.blit(player_image, (player_x, player_y))

#####
Draw blocks
for y in range(int(screen_height/bls)):
 for x in range(int(screen_width /bls)):
 relative_x = x - player_x
 relative_y = y - player_y
 screen.blit(block_images[relative_x + relative_y * int(screen_width/bls)], (x * bls, y * bls))
 #screen.blit(scaled_block_images[relative_x + relative_y * int(screen_width // (bls * 2))], (x * bls, y * bls))

```

```

scale_factor))), (x * bls * scale_factor, y * bls * scale_factor))

if(typei == 'n'):
 #Draw2 player
 screen.blit(player_image2, (player_x * bls, player_y * bls))
 #Draw Command pointer
 screen.blit(player_image, (command_point_x * bls, command_point_y * bls))
 #screen.blit(scaled_player_image, (player_x * bls * scale_factor, player_y * bls *
scale_factor))

 # Update display
 pygame.display.update()

#cm = int(input("Enter command number ID (0 to 65535): "))

#####
if(typei == 'd'):
 # Check for Xbox controller input
 if pygame.joystick.get_count() > 0:
 joystick = pygame.joystick.Joystick(0)
 joystick.init()
 # Check for D-Pad input
 dpad = joystick.get_hat(0)
 if dpad == (1, 0):
 player_x += 1
 elif dpad == (-1, 0):
 player_x-= 1
 elif dpad == (0, 1):
 player_y-= 1
 elif dpad == (0,-1):
 player_y += 1
 if player_x < 0:
 player_x = 0
 if player_x >= int(screen_width / bls):
 player_x = int(screen_width / bls)- 1
 if player_y < 0:
 player_y = 0
 if player_y >= int(screen_height / bls):
 player_y = int(screen_height / bls)- 1

if(typei == 's'):

#####
#LoadCommands
xin = 0
filnam = input("Enter name of Command Matrix file, 'filename.txt': ")

```

```

quet = int(input("Enter 1 if file already exists [Read mode], Else Enter 2 [Write mode], choose carefully: "))

bus = 255
amount = (bus + 1) * (bus + 1)
line_number = 0
#ai = log10(amount)
if(quet == 2):
 create_f = open(filnam, "w")
 while(line_number < amount):
 #i = log10(line_number)
 create_f.write("print(\"[Empty Command Slot] Change using a text-editor to Update this slot in\nCommand_Template.txt after renaming it accordingly.\")\n")
 line_number = line_number + 1

 create_f.close()

with open(filnam, 'r') as file:
 commands = file.readlines()
 xin = xin + 1

star = [0]

for i, command in enumerate(commands):
 commands[i] = command.strip()
 #cmd = command.split(",")
 star.append(command)
del star[0]

#reset state of matrix
typei = "k"
player_x = 0
player_y = 0

if(typei == 'k'):
 # Get user keyboard input
 keys = pygame.key.get_pressed()

 # Update move flag based on keyboard input
 if keys[pygame.K_LEFT] or keys[pygame.K_RIGHT] or keys[pygame.K_UP] or keys[pygame.K_DOWN]:
 move = True
 else:
 move = False

 # Update player position based on keyboard input
 if(move):
 if keys[pygame.K_LEFT]:
 player_x -= 1
 if keys[pygame.K_RIGHT]:
 player_x += 1

```

```

if keys[pygame.K_UP]:
 player_y-= 1
if keys[pygame.K_DOWN]:
 player_y += 1
if player_x < 0:
 player_x = 0
if player_x >= int(screen_width / bls):
 player_x = int(screen_width / bls)- 1
if player_y < 0:
 player_y = 0
if player_y >= int(screen_height / bls):
 player_y = int(screen_height / bls)- 1

if(typei == 'm'):
 # Get mouse position
 mouse_x, mouse_y = pygame.mouse.get_pos()

 # Update player position based on mouse position
 player_x = mouse_x // bls
 player_y = mouse_y // bls

Clear screen
screen.fill((255, 255, 255))

Draw blocks
if(typei == 'k' or typei == 'm' or typei == 'd'):
 for y in range(int(screen_height/bls)):
 for x in range(int(screen_width/bls)):
 relative_x = x- player_x
 relative_y = y- player_y
 screen.blit(block_images[relative_x + relative_y * int(screen_width // (bls))], (x * bls, y * bls))
 if(typei == 'k' or typei == 'm' or typei == 'd'):
 screen.blit(player_image, (player_x * bls, player_y * bls))

if(typei == 'k' or typei == 'm' or typei == 'd'):
 # Update display
 pygame.display.update()

Quit pygame
pygame.quit()

```

----- Content of Controller30IDE.py-----

```

import pygame
import importlib

```

```
import requests
import os
import subprocess
import webbrowser
import csv
from bs4 import BeautifulSoup
import datetime
import sys
import math
import random
import json
import re

#####import pywebdriver
#create a custom event type
MY_CUSTOM_EVENT = pygame.USEREVENT + 1

Initialize pygame
pygame.init()

Initialize joystick module
pygame.joystick.init()

Check if any joysticks are connected
if pygame.joystick.get_count() > 0:
 # Get the first joystick
 joystick = pygame.joystick.Joystick(0)
 # Initialize the joystick
 joystick.init()

Set screen size
#enter = int(input("Enter the dimension--> options 422, and multiples of 422"))
enter = 422
screen_width = enter
screen_height = enter

bls = 52
screen = pygame.display.set_mode((screen_width, screen_height))
#screen = pygame.display.set_mode((screen_width, screen_height), pygame.RESIZABLE)
#####
#####

#Create Commands
bus_ = 255
amount_ = (bus_ + 1) * (bus_ + 1)
line_number_ = 0

create_f = open("Command_Template.txt", "w")
while(line_number_ < amount_):
 #i = log10(line_number)
```

```

create_f.write("print(\"[Empty Command Slot] (Change using a text-editor to Update this slot in\n"
Command_Template.txt\") #remembering to rename Command_Template.txt\\n\")\n")
line_number_ = line_number_ + 1

create_f.close()

#LoadCommands
xin = 0
filnam = "Command_Template.txt"
with open(filnam, 'r') as file:
 commands = file.readlines()
 xin = xin + 1

star = [0]

for i, command in enumerate(commands):
 commands[i] = command.strip()
 #cmd = command.split(",")
 star.append(command)
del star[0]

#####
#####

#LoadCoordinates
xin2 = 0
with open('Coordinates_Python.txt', 'r') as file2:
 commands2 = file2.readlines()
 xin2 = xin2 + 1

starsx = [0]
starsy = [0]

for i2, command2 in enumerate(commands2):
 commands2[i2] = command2.strip()
 cmd2 = command2.split(",")
 starsx.append(int(cmd2[1]))
 varys = int(cmd2[2].strip())
 starsy.append(varys)

del starsx[0]
del starsy[0]

#####
#####

for i3, commandy in starsy:
 new = starsy[i3].strip()
 starsy[i3] = int(new)

```

```
#####
print(starsx)
print(starsy)
#####
Load images
block_images = []
for i in range(256):
 block_images.append(pygame.image.load(f"block{i}.png"))

Maze layout
maze_layout = [
 [0, 1, 2, 3],
 [4, 5, 6, 7],
 [8, 9, 10, 11],
 [12, 13, 14, 15]
]

#Function to check if a position is inside the maze
def is_inside_maze(x, y, maze_width, maze_height, bls):
 return 0 <= x < maze_width * bls and 0 <= y < maze_height * bls

Load player images
player_image = pygame.image.load("player.png")
player_image2 = pygame.image.load("player2.png")

#scaled_block_images = [pygame.transform.scale(img, (int(img.get_width() * scale_factor),
int(img.get_height() * scale_factor))) for img in block_images]
#scaled_player_image = pygame.transform.scale(player_image, (int(player_image.get_width() * scale_factor), int(player_image.get_height() * scale_factor)))

Set player position
player_x = 0
player_y = 0
con_program = int(input("Continue? 1 [yes], 0 [no]: "))
if(con_program == 1):
 print("Ok ...")
if(con_program == 0):
 print("Exiting ...")
 exit()
if(con_program != 0 and con_program != 1):
 print("Error")
 exit()
Set initial input type
typei = "k"
```

```
print("Switch Command Matrix [s],\n mouse [m],\nkeyboard arrows [k],\nD-Pad (Xbox One Controller)\n[d]\nPress [SPACEBAR] to call a command from Commands_Template.txt (Commands can be changed using\na text-editor) ... ")

some input variable
input_variable = 0
add the custom event to the event queue
pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": input_variable}))
#etchn loop
etch = 0

#maze width
maze_width = 16
maze_height = 16
Main game loop
running = True
#new screen size settings
##screen_width, screen_height = pygame.display.get_surface().get_size()
scale_factor_x = screen_width / maze_width
#scale_factor_y = screen_height / maze_height
#scale_factor = min(scale_factor_x, scale_factor_y)
scale_factor = scale_factor_x

bls = int(bls * 0.5)

scaled_block_images = []
for ni in range(256):
 s_i = pygame.transform.scale(block_images[ni], (bls, bls))
 block_images[ni] = s_i

while running:
 for event in pygame.event.get():
 if event.type == pygame.QUIT:
 running = False

 elif event.type == pygame.KEYDOWN:
 # Check for key press event
 if event.key == pygame.K_m and typei != "m":
 # Update input type
 typei = "m"
 elif event.key == pygame.K_k and typei != "k":
 # Update input type
 typei = "k"
 elif event.key == pygame.K_s and typei != "s":
 # Update input type
 typei = "s"
 elif event.key == pygame.K_d and typei != "d":
 # Update input type
 typei = "d"
 elif event.type == pygame.KEYDOWN:
```

```

if event.key == pygame.K_SPACE:
 #update input_type
 typei = "n"
 #check for status
 enter_cma = int(input("Enter command? 1[yes], 0 [no]: "))
 """
 # user input variable
 if(enter_cma == 0):
 typei = "k"
 player_x = 0
 player_y = 0
 etch = 1
 """
 if(enter_cma == 0):
 typei = "k"
 player_x = 0
 player_y = 0
 if(enter_cma == 1):
 input_variable = int(input("Enter command number ID (0 to 65535): "))
 # add the custom event to the event queue
 pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": input_variable}))
 #pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": cma}))
 """
 elif event.type == pygame.MOUSEBUTTONUP:
 # Get mouse position
 mouse_x, mouse_y = pygame.mouse.get_pos()

 # Check if the clicked position is inside the maze
 if is_inside_maze(mouse_x, mouse_y, len(maze_layout[0]), len(maze_layout), bls):
 # Calculate the clicked cell
 cell_x = mouse_x // bls
 cell_y = mouse_y // bls

 # Execute the desired action (e.g., print a message with the image ID)
 print(f"Hello, world! Image ID: {maze_layout[cell_y][cell_x]}")
 """

check for custom event type
for event in pygame.event.get(MY_CUSTOM_EVENT):
 # access the input variable from the event object
 cma = event.input_variable
 # do something with the input variable
 #pass...
 if(typei == 'n'):

 #enter_cma = int(input("Enter command? 1[yes], 0 [no]: "))
 if(enter_cma == 1 and enter_cma != 0):
 player_x = 0
 player_y = 0

```

```

Clear screen
screen.fill((255, 255, 255))
Update display
#pygame.display.update()

#cma = int(input("Enter command number ID (0 to 65535): "))
try:
 exec(star[cma], globals())
except:
 print("Try again, there was an error")
#update command index pointer
bus = 255 + 1
if(cma <= 255):
 cmd_index = 0
 command_point = cma
elif(cma > 255):
 cmd_index = cma//bus
 command_point = cma
 for tip in range(0, cmd_index):
 command_point = command_point- (bus)
#print(command_point)

prompt = "Executing Command " + str(cma) + " from index: " + str(cmd_index)
print(prompt)
#print(starsx)

player_x = starsy[cmd_index]
player_y = starsx[cmd_index]
command_point_x = starsy[command_point]
command_point_y = starsx[command_point]

Render player image at new position--> screen.blit(player_image, (player_x, player_y))

#####
Draw blocks
for y in range(int(screen_height/bls)):
 for x in range(int(screen_width /bls)):
 relative_x = x- player_x
 relative_y = y- player_y
 screen.blit(block_images[relative_x + relative_y * int(screen_width/bls)], (x * bls, y * bls))
 #screen.blit(scaled_block_images[relative_x + relative_y * int(screen_width // (bls * scale_factor))], (x * bls * scale_factor, y * bls * scale_factor))

if(typei == 'n'):

```

```

#Draw2 player
screen.blit(player_image2, (player_x * bls, player_y * bls))
#Draw Command pointer
screen.blit(player_image, (command_point_x * bls, command_point_y * bls))
#screen.blit(scaled_player_image, (player_x * bls * scale_factor, player_y * bls *
scale_factor))

Update display
pygame.display.update()

#cma = int(input("Enter command number ID (0 to 65535): "))

#####
if(typei == 'd'):
 # Check for Xbox controller input
 if pygame.joystick.get_count() > 0:
 joystick = pygame.joystick.Joystick(0)
 joystick.init()
 # Check for D-Pad input
 dpad = joystick.get_hat(0)
 if dpad == (1, 0):
 player_x += 1
 elif dpad == (-1, 0):
 player_x-= 1
 elif dpad == (0, 1):
 player_y-= 1
 elif dpad == (0,-1):
 player_y += 1
 if player_x < 0:
 player_x = 0
 if player_x >= int(screen_width / bls):
 player_x = int(screen_width / bls)- 1
 if player_y < 0:
 player_y = 0
 if player_y >= int(screen_height / bls):
 player_y = int(screen_height / bls)- 1

if(typei == 's'):

#####
#LoadCommands
xin = 0
filnam = input("Enter name of Command Matrix file, 'filename.txt': ")
quet = int(input("Enter 1 if file already exists [Read mode], Else Enter 2 [Write mode], choose carefully: "))

```

```

bus = 255
amount = (bus + 1) * (bus + 1)
line_number = 0
#ai = log10(amount)
if(quet == 2):
 create_f = open(filnam, "w")
 while(line_number < amount):
 #i = log10(line_number)
 create_f.write("print(\"[Empty Command Slot] Change using a text-editor to Update this slot in\nCommand_Template.txt after renaming it accordingly.\")\n")
 line_number = line_number + 1

 create_f.close()

with open(filnam, 'r') as file:
 commands = file.readlines()
 xin = xin + 1

star = [0]

for i, command in enumerate(commands):
 commands[i] = command.strip()
 #cmd = command.split(",")
 star.append(command)
del star[0]

#reset state of matrix
typei = "k"
player_x = 0
player_y = 0

if(typei == 'k'):
 # Get user keyboard input
 keys = pygame.key.get_pressed()

 # Update move flag based on keyboard input
 if keys[pygame.K_LEFT] or keys[pygame.K_RIGHT] or keys[pygame.K_UP] or keys[pygame.K_DOWN]:
 move = True
 else:
 move = False

 # Update player position based on keyboard input
 if(move):
 if keys[pygame.K_LEFT]:
 player_x -= 1
 if keys[pygame.K_RIGHT]:
 player_x += 1
 if keys[pygame.K_UP]:
 player_y -= 1
 if keys[pygame.K_DOWN]:
 player_y += 1

```

```

 player_y += 1
 if player_x < 0:
 player_x = 0
 if player_x >= int(screen_width / bls):
 player_x = int(screen_width / bls) - 1
 if player_y < 0:
 player_y = 0
 if player_y >= int(screen_height / bls):
 player_y = int(screen_height / bls) - 1

if(typei == 'm'):
 # Get mouse position
 mouse_x, mouse_y = pygame.mouse.get_pos()

 # Update player position based on mouse position
 player_x = mouse_x // bls
 player_y = mouse_y // bls

Clear screen
screen.fill((255, 255, 255))

Draw blocks
if(typei == 'k' or typei == 'm' or typei == 'd'):
 for y in range(int(screen_height/bls)):
 for x in range(int(screen_width/bls)):
 relative_x = x - player_x
 relative_y = y - player_y
 screen.blit(block_images[relative_x + relative_y * int(screen_width // (bls))], (x * bls, y * bls))
 if(typei == 'k' or typei == 'm' or typei == 'd'):
 screen.blit(player_image, (player_x * bls, player_y * bls))

if(typei == 'k' or typei == 'm' or typei == 'd'):
 # Update display
 pygame.display.update()

Quit pygame
pygame.quit()

```

----- Content of conv\_ps\_png2-nlv.py-----

```

from PIL import Image
nof = input("Input number of files: ")
nof_ = int(nof)
ins_x = input("Enter first file id: ")
x = int(ins_x)

```

```

donu = input("Do you have more than one file (y [Yes], n [No]? ")
if(donu == 'y'):
 ins_y = input("Enter last file id: ")
 y = int(ins_y)
ins = input("Enter path of files: ")

#def convert_to_png(path):

if(nof_ > 1):
 for i in range(x,y+1):
 ixy = str(i)
 #pre = input("Enter filename-prefix: ")
 path= ins + "\\\" + "Map_" + ix + ".ps"
 img = Image.open(path)
 img.save("Map_" + ix + ".png")
if(nof_ == 1):
 #pre = input("Enter filename-prefix: ")
 path = ins + "\\\" + "Map_" + ins_x + ".ps"
 img = Image.open(path)
 img.save("Map_" + ins_x + ".png")

#path= ins
#convert_to_png(path)

```

----- Content of conv\_ps\_png2.py-----

```

from PIL import Image
nof = input("Input number of files: ")
nof_ = int(nof)
ins_x = input("Enter first file id: ")
x = int(ins_x)
donu = input("Do you have more than one file (y [Yes], n [No]? ")
if(donu == 'y'):
 ins_y = input("Enter last file id: ")
 y = int(ins_y)
ins = input("Enter path of files: ")

```

```

#def convert_to_png(path):

if(nof_ > 1):
 for i in range(x,y+1):
 ixy = str(i)
 #pre = input("Enter filename-prefix: ")
 path= ins + "\\\" + "M" + ix + ".ps"
 img = Image.open(path)
 img.save("M" + ix + ".png")
if(nof_ == 1):
 #pre = input("Enter filename-prefix: ")
 path = ins + "\\\" + "M" + ins_x + ".ps"
 img = Image.open(path)

```

```

img.save("M" + ins_x + ".png")

#path= ins
#convert_to_png(path)

----- Content of conv_ps_png3.py-----
from PIL import Image
nof = input("Input number of files: ")
nof_ = int(nof)
ins_x = input("Enter first file id: ")
x = int(ins_x)
donu = input("Do you have more than one file (y [Yes], n [No]? ")
if(donu == 'y'):
 ins_y = input("Enter last file id: ")
 y = int(ins_y)
ins = input("Enter path of files: ")

#def convert_to_png(path):

if(nof_ > 1):
 for i in range(x,y+1):
 ixy = str(i)
 #pre = input("Enter filename-prefix: ")
 path= ins + "\\\" + ix + ".ps"
 img = Image.open(path)
 img.save(ixy + ".png")
if(nof_ == 1):
 #pre = input("Enter filename-prefix: ")
 path = ins + "\\\" + ins_x + ".ps"
 img = Image.open(path)
 img.save(ins_x + ".png")

#path= ins
#convert_to_png(path)

```

```

----- Content of cpp.cpp-----
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

int main() {
 std::string inputFileName, outputFileName;
 std::vector<std::string> lines;

 // Get input file name from the user
 std::cout << "Enter the input file name: ";
 std::cin >> inputFileName;

```

```
// Get output file name from the user
std::cout << "Enter the output file name: ";
std::cin >> outputFileName;

// Open the input file
std::ifstream inputFile(inputFileName);
if (!inputFile) {
 std::cerr << "Error opening input file: " << inputFileName << std::endl;
 return 1;
}

// Read the input file and store lines in the vector
std::string line;
while (std::getline(inputFile, line)) {
 lines.push_back(line);
}

inputFile.close();

bool continueEditing = true;

while (continueEditing) {
 // Display the current contents of the file
 std::cout << "Current contents of the file:" << std::endl;
 for (size_t i = 0; i < lines.size(); ++i) {
 std::cout << "[" << i + 1 << "] " << lines[i] << std::endl;
 }
}

int choice;
std::cout << "Choose an option:" << std::endl;
std::cout << "1. Edit a line" << std::endl;
std::cout << "2. Delete a line" << std::endl;
std::cout << "3. Add a line after a specified line" << std::endl;
std::cout << "4. Exit" << std::endl;
std::cin >> choice;

if (choice == 1) {
 int lineToEdit;
 std::string newContent;

 // Get the line number to edit from the user
 std::cout << "Enter the line number to edit (1-based index): ";
 std::cin >> lineToEdit;

 // Make sure the line number is valid
 if (lineToEdit < 1 || lineToEdit > lines.size()) {
 std::cerr << "Invalid line number." << std::endl;
 continue;
 }
}
```

```
// Display the current content of the line to be edited
std::cout << "Current content of line " << lineToEdit << ":" << lines[lineToEdit- 1] << std::endl;

// Get the new content from the user
std::cin.ignore(); // Clear the newline character from the buffer
std::cout << "Enter the new content for line " << lineToEdit << ":" ;
std::getline(std::cin, newContent);

// Modify the specified line with the new content
lines[lineToEdit- 1] = newContent;

std::cout << "Line " << lineToEdit << " edited successfully." << std::endl;
} else if (choice == 2) {
 int lineToDelete;

 // Get the line number to delete from the user
 std::cout << "Enter the line number to delete (1-based index): ";
 std::cin >> lineToDelete;

 // Make sure the line number is valid
 if (lineToDelete < 1 || lineToDelete > lines.size()) {
 std::cerr << "Invalid line number." << std::endl;
 continue;
 }

 // Delete the specified line
 lines.erase(lines.begin() + lineToDelete- 1);

 std::cout << "Line " << lineToDelete << " deleted successfully." << std::endl;
} else if (choice == 3) {
 int lineToAddAfter;
 std::string newLineContent;

 // Get the line number to add a new line after from the user
 std::cout << "Enter the line number to add a new line after (1-based index): ";
 std::cin >> lineToAddAfter;

 // Make sure the line number is valid
 if (lineToAddAfter < 1 || lineToAddAfter > lines.size()) {
 std::cerr << "Invalid line number." << std::endl;
 continue;
 }

 // Get the new line content from the user
 std::cin.ignore(); // Clear the newline character from the buffer
 std::cout << "Enter the new line content: ";
 std::getline(std::cin, newLineContent);

 // Insert the new line after the specified line
 lines.insert(lines.begin() + lineToAddAfter, newLineContent);
}
```

```

 lines.insert(lines.begin() + lineToAddAfter, newLineContent);

 std::cout << "New line added successfully after line " << lineToAddAfter << "." << std::endl;
 } else if (choice == 4) {
 continueEditing = false;
 } else {
 std::cerr << "Invalid option. Please choose a valid option." << std::endl;
 }
}

// Open the output file
std::ofstream outputFile(outputFileName);
if (!outputFile) {
 std::cerr << "Error opening output file: " << outputFileName << std::endl;
 return 1;
}

// Write the modified content to the output file
for (const std::string& modifiedLine : lines) {
 outputFile << modifiedLine << "\n";
}

outputFile.close();

std::cout << "File edited and saved successfully." << std::endl;

return 0;
}

```

----- Content of CPPDatabase.cpp-----

```

#include <iostream>
#include <fstream>
#include <set>
#include <string>

int main() {
 std::set<std::string> database;
 std::string filename = "database.txt";
 std::string line;

 // Load existing entries from the file
 std::ifstream infile(filename);
 if (infile.is_open()) {
 while (getline(infile, line)) {
 database.insert(line);
 }
 infile.close();
 }
}

```

```

// Get new entries from the user
while (true) {
 std::cout << "Enter a new entry (or '0' to quit): ";
 getline(std::cin, line);

 if (line == "0") {
 break;
 }

 // Check for duplicate entry
 if (database.find(line) != database.end()) {
 std::cout << "Duplicate entry. Entry not added to the database.\n";
 } else {
 // Add new entry to the set and the file
 database.insert(line);
 std::ofstream outfile(filename, std::ios_base::app);
 if (outfile.is_open()) {
 outfile << line << "\n";
 outfile.close();
 std::cout << "Entry added to the database.\n";
 } else {
 std::cout << "Unable to open file for writing.\n";
 }
 }
}

return 0;
}

```

----- Content of cpttoexe.cpp-----

```

#include <iostream>
#include <cstdlib>
#include <filesystem>

int main() {
 // Get the current working directory
 std::filesystem::path current_path = std::filesystem::current_path();

 // Iterate through each file in the current directory
 for (const auto& entry : std::filesystem::directory_iterator(current_path)) {
 std::filesystem::path file_path = entry.path();

 // Check if the file has a .cpp extension
 if (file_path.extension() == ".cpp") {
 std::string compile_command = "g++ " + file_path.string() + " -o " + file_path.stem().string();

 // Compile the .cpp file
 std::cout << "Compiling: " << file_path.string() << std::endl;
 std::system(compile_command.c_str());
 }
 }
}

```

```
 }
}
```

```
 return 0;
}
```

----- Content of CPP\_C\_Source\_Code\_Generator.cpp-----

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(){
 //FILE *p; p = fopen("SOLUTION.c", "w");
 char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
 '\n','\t','\\','\"','/','<','>','?','!','@','#','~','[','{','}',']','|','!','"','`','%','^','&','*','(',')','_','+','=','.,','A','B',
 'C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9'};
 int k, kprompt, fnum;
 k = sizeof(a) - 1;
 int soe = 0;
 puts("Set Of Elements");
 do{
 printf("\n\tElement %d = %c", soe, a[soe]);
 soe++;
 }while(soe <= k);
 printf("\n\tNumber Of Elements = %d", k + 1);
 printf("\n\tK = Number of Elements - 1");
 printf("\n\tTherefore k = %d", k);
 int noc, assert, go;
 printf("\n\nn = Number Of Cells Per File To Be Generated");
 printf("\n\nn\t(k+1)^n = Number of Files To Be Generated");
 printf("\n\tEnter 2 = Search For A Specific File\n\tEnter 1 = Generate Files\n\tEnter 0 =
Abort\n\n\t"); scanf("%d", &go);
 if(go == 1){
 assert = 2;
 }else if(go == 2){
 assert = 1;
 }else if(go == 0){
 assert = 0;
 }else
 assert = 3;
 //Get Files From Database and save to directory
 if(assert == 2){
 printf("\n\tPlease Enter a Value For n:\t");
 scanf("%d", &noc); printf("\n\tNumber Of FILE Cells To Be Generated = %d", noc);
 int n = noc; int row, col; int cell; int rdiv; unsigned long long id; id = 0;
 int nbr_comb = pow(k+1, n);
 for (row=0; row < nbr_comb; row++){
 char filename[500];
 sprintf(filename, "%dC%d.c", n, id);
```

```

//FILE *p; p = fopen("SOLUTION.txt","w");
FILE *p; p = fopen(filename,"w");
id++;
//fprintf(p,"\n\n\nFILE%d\n\n\n", id);
for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col); cell = (row/rdiv) % (k+1); fprintf(p,"%c",
a[cell]);
}

//if(id == nbr_comb){
// fprintf(p,"\n\n\t(k+1)^n = (%d + 1)^%d = %d", k, n, id);
//}

fclose(p);
printf("\n");
}

//Get One Specific File From Database and save to active Directory
if(assert == 1){

printf("\n\tPlease Enter a Value For n:\t");
scanf("%d", &noc);
int n = noc;
int nbr_comb = pow(k+1, n);
printf("\n\n\tPlease Enter file number wanted between 0 and %d Inclusively:\t",
nbr_comb-1);
scanf("%d", &fnum);
printf("\n\tNumber Of FILE Cells To Be Generated = %d", noc);
int row, col; int cell; int rdiv; unsigned long long id; id = fnum;

int sw = 1;
for (row=0; row < nbr_comb; row++){
 if(row == fnum){
 char filename[500];
 sprintf(filename, "DQUERY%dC%d.c", n, id);
//FILE *p; p = fopen("SOLUTION.txt","w");
FILE *p; p = fopen(filename,"w");
id++;
//fprintf(p,"\n\n\nFILE%d\n\n\n", id);
for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col); cell = (row/rdiv) % (k+1);

fprintf(p,"%c", a[cell]);
 sw = 0;
 }
 if(sw == 0)
 break;
}
}
}

```

```

 }

 if(assert == 0 || assert == 3){
 puts("exiting");
 }
 return 0;
}

```

----- Content of CPP\_Source\_Code\_Generator.cpp-----

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(){
 //FILE *p; p = fopen("SOLUTION.c","w");
 char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
 '\n','\t','\\','\"','/','<','>','?',';','@','#','^','!','{','}','[','}',',','!','"',$',','%','^','&','*','(',')','_','+','=','.',','A','B',
 'C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9'};
 int k, kprompt, fnum;
 k = sizeof(a)- 1;
 int soe = 0;
 puts("Set Of Elements");
 do{
 printf("\n\tElement %d = %c", soe, a[soe]);
 soe++;
 }while(soe <= k);
 printf("\n\tNumber Of Elements = %d", k + 1);
 printf("\n\tK = Number of Elements- 1");
 printf("\n\tTherefore k = %d", k);
 int noc, assert, go;
 printf("\n\tn = Number Of Cells Per File To Be Generated");
 printf("\n\t(k+1)^n = Number of Files To Be Generated");
 printf("\n\tEnter 2 = Search For A Specific File\n\tEnter 1 = Generate Files\n\tEnter 0 =
Abort\n\t"); scanf("%d", &go);
 if(go == 1){
 assert = 2;
 }else if(go == 2){
 assert = 1;
 }else if(go == 0){
 assert = 0;
 }else
 assert = 3;
 //Get Files From Database and save to directory
 if(assert == 2){
 printf("\n\tPlease Enter a Value For n:\t");
 scanf("%d", &noc); printf("\n\tNumber Of FILE Cells To Be Generated = %d", noc);
 }
}

```

```

int n = noc; int row, col; int cell; int rdiv; unsigned long long id; id = 0;
int nbr_comb = pow(k+1, n);
for (row=0; row < nbr_comb; row++){
 char filename[500];
 sprintf(filename, "%dC%d.cpp", n, id);
 //FILE *p; p = fopen("SOLUTION.txt", "w");
 FILE *p; p = fopen(filename, "w");
 id++;
 //fprintf(p, "\n\n\n\nFILE%d\n\n\n", id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col); cell = (row/rdiv) % (k+1); fprintf(p, "%c",
a[cell]);
 }
}

//if(id == nbr_comb){
// fprintf(p, "\n\n\n\t(k+1)^n = (%d + 1)^%d = %d", k, n, id);
//}

fclose(p);
printf("\n");
}

}

//Get One Specific File From Database and save to active Directory
if(assert == 1){

 printf("\n\tPlease Enter a Value For n:\t");
 scanf("%d", &noc);
 int n = noc;
 int nbr_comb = pow(k+1, n);
 printf("\n\n\tPlease Enter file number wanted between 0 and %d Inclusively:\t",
nbr_comb-1);
 scanf("%d", &fnum);
 printf("\n\tNumber Of FILE Cells To Be Generated = %d", noc);
 int row, col; int cell; int rdiv; unsigned long long id; id = fnum;

 int sw = 1;
 for (row=0; row < nbr_comb; row++){
 if(row == fnum){
 char filename[500];
 sprintf(filename, "DQUERY%dC%d.cpp", n, id);
 //FILE *p; p = fopen("SOLUTION.txt", "w");
 FILE *p; p = fopen(filename, "w");
 id++;
 //fprintf(p, "\n\n\n\nFILE%d\n\n\n", id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col); cell = (row/rdiv) % (k+1);
 fprintf(p, "%c", a[cell]);
 }
 sw = 0;
 }
 }
 if(sw == 0)
}

```

```
 break;
 }
}

}

if(assert == 0 || assert == 3){
 puts("exiting");
}
return 0;
}
```

----- Content of ctoexe.cpp-----

```
#include <iostream>
#include <cstdlib>
#include <filesystem>

int main() {
 // Get the current working directory
 std::filesystem::path current_path = std::filesystem::current_path();

 // Iterate through each file in the current directory
 for (const auto& entry : std::filesystem::directory_iterator(current_path)) {
 std::filesystem::path file_path = entry.path();

 // Check if the file has a .c extension
 if (file_path.extension() == ".c") {
 std::string compile_command = "gcc " + file_path.string() + "-o " + file_path.stem().string();

 // Compile the .c file
 std::cout << "Compiling: " << file_path.string() << std::endl;
 std::system(compile_command.c_str());
 }
 }

 return 0;
}
```

----- Content of C\_CPP\_Source\_Code\_Generator.c-----

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(){
```

```

//FILE *p; p = fopen("SOLUTION.c", "w");
char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
'\n','\t','\\','\"','/','<','>','?','!','@','#','~','[','{','}',']','|','!','"',"$",'%', '^','&','*','(',')','_','+','=','.','A','B',
'C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9'};
int k, kprompt, fnum;
k = sizeof(a)- 1;
int soe = 0;
puts("Set Of Elements");
do{
 printf("\n\tElement %d = %c", soe, a[soe]);
 soe++;
}while(soe <= k);
printf("\n\tNumber Of Elements = %d", k + 1);
printf("\n\tK = Number of Elements - 1");
printf("\n\tTherefore k = %d", k);
int noc, assert, go;
printf("\n\tn = Number Of Cells Per File To Be Generated");
printf("\n\n\t(k+1)^n = Number of Files To Be Generated");
printf("\n\tEnter 2 = Search For A Specific File\n\tEnter 1 = Generate Files\n\tEnter 0 =
Abort\n\n\t"); scanf("%d", &go);
if(go == 1){
 assert = 2;
}else if(go == 2){
 assert = 1;
}else if(go == 0){
 assert = 0;
}else
 assert = 3;
//Get Files From Database and save to directory
if(assert == 2){
 printf("\n\tPlease Enter a Value For n:\t");
 scanf("%d", &noc); printf("\n\tNumber Of FILE Cells To Be Generated = %d", noc);
 int n = noc; int row, col; int cell; int rdiv; int id; id = 0;
 int nbr_comb = pow(k+1, n);
 for (row=0; row < nbr_comb; row++){
 char filename[500];
 sprintf(filename, "%dC%d.cpp", n,id);
 //FILE *p; p = fopen("SOLUTION.txt", "w");
 FILE *p; p = fopen(filename, "w");
 id++;
 //fprintf(p,"n\nn\nFILE%d\nn\nn", id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col); cell = (row/rdiv) % (k+1); fprintf(p,"%c",
a[cell]);
 }
 }
 //if(id == nbr_comb){
 // fprintf(p,"n\nn\t(k+1)^n = (%d + 1)^%d = %d", k, n, id);
 //}
}

```

```

 fclose(p);
 printf("\n");
 }
}

//Get One Specific File From Database and save to active Directory
if(assert == 1){

 printf("\n\tPlease Enter a Value For n:\t");
 scanf("%d", &noc);
 int n = noc;
 int nbr_comb = pow(k+1, n);
 printf("\n\n\tPlease Enter file number wanted between 0 and %d Inclusively:\t",
nbr_comb-1);
 scanf("%d", &fnum);
 printf("\n\tNumber Of FILE Cells To Be Generated = %d", noc);
 int row, col; int cell; int rdiv; int id; id = fnum;

 int sw = 1;
 for (row=0; row < nbr_comb; row++){
 if(row == fnum){
 char filename[500];
 sprintf(filename, "DQUERY%dC%d.cpp", n,id);
 //FILE *p; p = fopen("SOLUTION.txt", "w");
 FILE *p; p = fopen(filename,"w");
 id++;
 //fprintf(p, "\n\n\n\nFILE%d\n\n\n", id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col); cell = (row/rdiv) % (k+1);
 fprintf(p,"%c", a[cell]);
 sw = 0;
 }
 if(sw == 0)
 break;
 }
 }
}

if(assert == 0 || assert == 3){
 puts("exiting");
}
return 0;
}

```

----- Content of C\_Source\_Code\_Generator.c-----

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(){
 //FILE *p; p = fopen("SOLUTION.c", "w");
 char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
 '\n','\t','\\','\\\\','/','<','>','?','!',';','@','#','^','_','[','{','}','_','_','!','"','$','%','^','&','*','(',')','_','+','=','.,A','B',
 'C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9'};
 int k, kprompt, fnum;
 k = sizeof(a)- 1;
 int soe = 0;
 puts("Set Of Elements");
 do{
 printf("\n\tElement %d = %c", soe, a[soe]);
 soe++;
 }while(soe <= k);
 printf("\n\tNumber Of Elements = %d", k + 1);
 printf("\n\tK = Number of Elements- 1");
 printf("\n\tTherefore k = %d", k);
 int noc, assert, go;
 printf("\n\nn = Number Of Cells Per File To Be Generated");
 printf("\n\nn\t(k+1)^n = Number of Files To Be Generated");
 printf("\n\tEnter 2 = Search For A Specific File\n\tEnter 1 = Generate Files\n\tEnter 0 =
Abort\n\n"); scanf("%d", &go);
 if(go == 1){
 assert = 2;
 }else if(go == 2){
 assert = 1;
 }else if(go == 0){
 assert = 0;
 }else
 assert = 3;
 //Get Files From Database and save to directory
 if(assert == 2){
 printf("\n\tPlease Enter a Value For n:\t");
 scanf("%d", &noc); printf("\n\tNumber Of FILE Cells To Be Generated = %d", noc);
 int n = noc; int row, col; int cell; int rdiv; int id; id = 0;
 int nbr_comb = pow(k+1, n);
 for (row=0; row < nbr_comb; row++){
 char filename[500];
 sprintf(filename, "%dC%d.c", n, id);
 //FILE *p; p = fopen("SOLUTION.txt", "w");
 FILE *p; p = fopen(filename, "w");
 id++;
 //fprintf(p, "\n\n\nFILE%d\n\n\n", id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col); cell = (row/rdiv) % (k+1); fprintf(p,"%c",
a[cell]);
 }
 }
 }
}

```

```

 //if(id == nbr_comb){
 // fprintf(p, "\n\n\t(k+1)^n = (%d + 1)^%d = %d", k, n, id);
 //}

 fclose(p);
 printf("\n");
 }

}

//Get One Specific File From Database and save to active Directory
if(assert == 1){

 printf("\n\tPlease Enter a Value For n:\t");
 scanf("%d", &noc);
 int n = noc;
 int nbr_comb = pow(k+1, n);
 printf("\n\n\tPlease Enter file number wanted between 0 and %d Inclusively:\t",
nbr_comb-1);
 scanf("%d", &fnum);
 printf("\n\tNumber Of FILE Cells To Be Generated = %d", noc);
 int row, col; int cell; int rdiv; int id; id = fnum;

 int sw = 1;
 for (row=0; row < nbr_comb; row++){
 if(row == fnum){
 char filename[500];
 sprintf(filename, "DQUERY%dC%d.c", n, id);
 //FILE *p; p = fopen("SOLUTION.txt", "w");
 FILE *p; p = fopen(filename, "w");
 id++;
 //fprintf(p, "\n\n\nFILE%d\n\n\n", id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col); cell = (row/rdiv) % (k+1);
 fprintf(p, "%c", a[cell]);
 sw = 0;
 }
 if(sw == 0)
 break;
 }
 }

}

if(assert == 0 || assert == 3){
 puts("exiting");
}

```

```
 return 0;
 }
```

## ----- Content of DaT-hts.cpp -----

//DaT-hts.cpp is an executable suitable for designing, manufacturing, and distributing, hardware, tools and software. That gives 9 workflows.

/\*

0 Design and Technology for Hardware, tools, software. The Design, manufacture and distribution of each.

```
1 Design 'Hardware'
2 Design 'Tool'
3 Design 'Software'
4 Manufacture 'Hardware'
5 Manufacture 'Tool'
6 Manufacture 'Software'
7 Distribute 'Hardware'
8 Distribute 'Tool'
9 Distribute 'Software'
*/
```

----- Content of database-texed.cpp -----

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
using namespace std;
//put together by Dominic Alexander Cooper
int main(){
 puts("(pt) == Universal Set = Computer");
 puts("(pt)i = sub-set of (pt)");
 puts("(pt)ia = element of sub-set of (pt)");
 puts("ti = ia");
 puts("t = i of (pt)ia ");
 puts("i = a of ((pt)ia)");
 puts("\nProgressive Abstract Numerical Solution");
 //Code Adapted by DAC from lyst on https://stackoverflow.com
 //k+1 = no. of elements
 //n = exponent = number of cells
 //the k and n values must perfectly fit the size of the set of elements in question
 FILE *p; p = fopen("PROGRESSIVE_ANS_px.RENAME.txt", "w");
 //char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
 '\n','\t','\\','\"','/','<','>','?','!','@','#','~','!','[','{','}',']','{','}',',','!','!','!','!',',','$','%','^','&','*','(',')','_','_',
 '+','-','!','A','B','C','D','E','F','G','H',
 'I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9'};
 int a[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,
 8,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,
 73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,
 05,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126};

```

```

130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,15
4,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,1
79,180,181,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,
204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,22
8,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,2
53,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,
278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,30
2,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,3
27,328,329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,346,347,348,349,350,351,
352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,373,374,375,37
6,377,378,379,380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,397,398,399,400,4
01,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,423,424,425,
426,427,428,429,430,431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,448,449,45
0,451,452,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,469,470,471,472,473,474,4
75,476,477,478,479,480,481,482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,499,
500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521,522,523,52
4,525,526,527,528,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,5
49,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,
574,575,576,577,578,579,580,581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597,59
8,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,618,619,620,621,622,6
23,624,625,626,627,628,629,630,631,632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,
648,649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665,666,667,668,669,670,671,67
2,673,674,675,676,677,678,679,680,681,682,683,684,685,686,687,688,689,690,691,692,693,694,695,696,6
97,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721,
722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,74
6,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,7
71,772,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,
796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,82
0,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,8
45,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,
870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,89
4,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,9
19,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,
944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,96
8,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,9
3,994,995,996,997,998,999};
 int pin;

```

```

 int pr, pc;
 printf("\nLet us begin\n\n");
 pc = 1000;
 for(pr = 0; pr < pc; pr++){
 printf("%d %d\n", pr, a[pr]);
 }
 char choice;
 cout << "Enter a to uniquely use the 1 to 1000 identifiers for the Default Library or c to create a
custom Library: ";
 cin >> choice;

 cout << "Enter the size of your array: ";

```

```

cin >> pin;

int array[pin], inn, position;
int generate;
cout << "\nDo you want manually create your array (0) or, do you want to generate your array for
a range of numbers (1): ";
cin >> generate;
int lbrange;
int ubrange;
//int i_lbrange, i_ubrange;
int r_check;
r_check = 0;

if(choice == 'c'){
 //cout << "Enter the size of your array: ";
 //cin >> pin;
 //int array[pin], inn, position;

 if(generate == 0){
 cout << "\n\n Enter the " << pin << " elements of your array: \n\n";
 for(inn = 0; inn < pin; inn++){
 cin >> array[inn];
 }
 }
 if(generate == 1){
 cout << "\nEnter the lower inclusive bound of the array range of numbers:
";
 cin >> lbrange;
 //i_lbrange = int(lbrange);
 cout << "\nEnter the upper inclusive bound of the array range of numbers:
";
 cin >> ubrange;
 //i_ubrange = ubrange;
 r_check = lbrange;
 //generate custom array from user input;
 for(inn = 0; inn < pin; inn++){
 // r_check = r_check + 1;
 array[inn] = r_check;
 printf("\n%d", r_check);
 r_check = r_check + 1;
 if (r_check > ubrange){
 break;
 }
 }
 }
}

//for(inn = 0; inn < pin; inn++){
// cin >> array[inn];
}

```

```

// }

//char a[] = {'1','2','3','4','p1','p2','p3','p4','p5','p6'};
//char b[] = {'a','b'};
//char c[] = {'0','1','2','3'};
//int n = 4; //int k = 3; //int n = x;
//int k = 100;

//int k = strlen(a)- 1;
int k;
if(choice == 'c'){
 k = pin- 1;
}
if(choice == 'a'){
 k = 1000- 1;
}
printf("\n\tk = %d", k);
int noc; printf("\n\tn = ");
scanf("%d", &noc);
printf("\n\tNumber Of FILE Cells = %d", noc);
int n = noc;
int row, col;
int cell;
int rdiv;
int id;
id = 0;
int rin;
int nbr_comb = pow(k+1, n);
int ai, ci, switch_c, switch_a;

if(choice == 'c'){

 switch_c = 0;
 cout << "\n\nEnter nth File System (Number of Cells Per File of the nth File System). Let
the value be equal to n for a progressionless session: ";
 cin >> ci;

 for(ci; n < ci + 1; n++){
 if(switch_c == 1){
 nbr_comb = pow(k + 1, n);
 row = 0;
 }
 for (row=0; row < nbr_comb; row++){
 id++; fprintf(p,"%d\t", id);
 }
 }
}

```

```

 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);

 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 if(col == 0){
 fprintf(p,"%d\n", a[rin]);
 }
 if(col != 0){
 fprintf(p,"%d ", a[rin]);
 }

 //fprintf(p, "%d", a[cell]);
 }
 //printf("\n");
 }
 switch_c = 1;
}
}

int check;
check = pin - 1000;

```

```

if(choice == 'a' && check == 0){

 switch_a = 0;
 cout << "\n\nEnter nth File System (Number of Cells Per File of the nth File System). Let
the value be equal to n for a progressionless session: ";
 cin >> ai;

 for(n; n < ai + 1; n++){
 if(switch_a == 1){
 nbr_comb = pow(k + 1, n);
 row = 0;
 }
 for (row=0; row < nbr_comb; row++){
 id++;
 fprintf(p," \n\n%dCF%d\n\n", n, id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);

 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 if(col == 0){
 fprintf(p,"%d", a[cell]);
 }
 if(col != 0){

```

```

 fprintf(p,"%d ", a[cell]);
 }
 //fprintf(p, "%d", a[cell]);
}
//printf("\n");
}
switch_a = 1;
}
}

//fprintf(p,"n\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
fclose(p);
//end of adaptation
return 0;
}

```

----- Content of Data.Strings\_Generator.cpp-----

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
using namespace std;
//put together by Dominic Alexander Cooper
int main(){
 //Code Adapted by DAC from lyst on https://stackoverflow.com
 //k+1 = no. of elements
 //n = exponent = number of cells
 //the k and n values must perfectly fit the size of the set of elements in question
 FILE *p; p = fopen("SOLUTION_RENAME_MAX.txt","w");
 char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
 '\n','\t','\\','\"','<','>','?','!','@','#','~','[','{','}',',' ','!','"',$','%','^','*','(',')','_','+','=','!','A','B',
 'C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9'};
 int pin;

 int pr, pc;
 printf("\nLet us begin\n\n");
 pc = 100;
 puts("26 = SPACE");
 puts("27 = NEWLINE");
 puts("28 = TAB");
 puts("");
 for(pr = 0; pr < pc; pr++){
 printf("%d %c\n", pr, a[pr]);
 }

 int cheque;
 cout << "Enter the size of your array: ";
 cin >> pin;
 cout << "Do you want to create a custom array? ";

```

```

cin >> cheque;
int array[pin], inn, position;
if(cheque == 1){
 cout << "\n\n Enter the " << pin << " elements of your array: \n\n";
 for(inn = 0; inn < pin; inn++){
 cin >> array[inn];
 }
}
//char a[] = {'1','2','3','4','p1','p2','p3','p4','p5','p6'};
//char b[] = {'a','b'};
//char c[] = {'0','1','2','3'};
//int n = 4; //int k = 3; //int n = x;
//int k = 100;

//int k = strlen(a)- 1;
int k;
k = pin- 1;
printf("\n\tnk = %d", k);
int noc; printf("\n\n\t n = ");
scanf("%d", &noc);
printf("\n\n\tNumber Of FILE Cells = %d", noc);
int n = noc;
int row, col;
int cell;
int rdiv;
unsigned long long id;
id = 0;
int rin;
int nbr_comb = pow(k+1, n);
if(cheque == 1){
 for (row=0; row < nbr_comb; row++){
 id++; fprintf(p,"%d ", (id- 1));
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);
 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 fprintf(p,"%c", a[rin]);
 }
 fprintf(p,"\\n");
 //printf("\\n");
 }
}
if(cheque == 0){
for (row=0; row < nbr_comb; row++){
 id++; fprintf(p,"%d ", (id- 1));
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 cell = (row/rdiv) % (k+1);
 fprintf(p,"%c", a[cell]);
 }
}
}

```

```

 fprintf(p, "\n");
 //printf("\n");
 }
}

//fprintf(p, "\n\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
fclose(p);
//end of adaptation
return 0;
}

```

----- Content of documentation-gen.py-----

```

import os
from fpdf import FPDF

def list_text_files():
 extensions = ['.txt', '.py', '.c', '.cpp', '.rb']
 # Filter files in the current directory based on specified extensions
 return [f for f in os.listdir() if any(f.endswith(ext) for ext in extensions)]

def read_file_content(filename):
 try:
 with open(filename, 'rb') as file:
 content = file.read().decode('utf-8', errors='ignore')
 return content
 except Exception as e:
 return str(e)

def create_txt(files):
 # Create a text file and write contents of each listed file
 with open("BTOS_Documentation.txt", "w", encoding='utf-8') as txt_file:
 for file in files:
 txt_file.write(f"----- Content of {file}-----\n")
 txt_file.write(read_file_content(file))
 txt_file.write("\n\n")

def create_pdf(files):
 pdf = FPDF(format='a5') # Create PDF object
 pdf.set_auto_page_break(auto=1, margin=8)
 pdf.add_page()
 pdf.set_font("Arial", size=10)

 cell_width = 130 # Adjusted width for content cells
 line_height = 6

 for file in files:
 pdf.cell(200, 10, f"----- Content of {file}-----", ln=1)
 content = read_file_content(file)

```

```

is_even_line = True # Toggle for alternating background colors

for line in content.split('\n'):
 words = line.split(' ')
 current_line = ""

 for word in words:
 next_line = f'{current_line}{word}'.strip()
 next_line_width = pdf.get_string_width(next_line)

 if next_line_width <= cell_width:
 current_line = next_line
 else:
 if is_even_line:
 pdf.set_fill_color(173, 216, 230) # Light blue RGB
 else:
 pdf.set_fill_color(255, 255, 0) # Yellow RGB
 pdf.multi_cell(cell_width, line_height, current_line, fill=True)
 is_even_line = not is_even_line
 current_line = word

 if current_line: # Output any remaining text in current_line
 if is_even_line:
 pdf.set_fill_color(173, 216, 230) # Light blue RGB
 else:
 pdf.set_fill_color(255, 255, 0) # Yellow RGB
 pdf.multi_cell(cell_width, line_height, current_line, fill=True)
 is_even_line = not is_even_line

pdf.add_page()

pdf.output("BTOS_Documentation.pdf", 'F') # Save PDF to file

if __name__ == "__main__":
 text_files = list_text_files()

 create_txt(text_files) # Generate the text file
 #create_pdf(text_files) # Generate the PDF file

 print("Documentation updated: Created 'BTOS_documentation.txt' and 'BTOS_documentation.pdf'")

----- Content of Encode.cpp-----
#include <iostream>
#include <fstream>
#include <vector>
#include <filesystem>
#include <string>
#include <unordered_map>

```

```

using namespace std;
namespace fs = std::filesystem;

vector<unordered_map<string, string>> readMappings() {
 vector<unordered_map<string, string>> mappings(4);
 for (int i = 1; i <= 4; ++i) {
 ifstream file(to_string(i) + ".txt");
 if (!file) {
 cerr << "Error opening file " << i << ".txt" << endl;
 exit(EXIT_FAILURE);
 }
 string line;
 while (getline(file, line)) {
 int spacePos = line.find(' ');
 if (spacePos != string::npos) {
 string numberStr = line.substr(0, spacePos);
 string str = line.substr(spacePos + 1);
 mappings[i-1][str] = numberStr;
 }
 }
 }
 return mappings;
}

int main() {
 auto mappings = readMappings();

 int start, end;
 cout << "Enter the range of files to read (e.g., 5 15 to read from 5.txt to 15.txt): ";
 cin >> start >> end;

 int maxFileNameNumber = -1;

 for (const auto& entry : fs::directory_iterator("./")) {
 string filename = entry.path().filename().string();
 if (filename.find(".txt") != string::npos) {
 int fileNameNumber = stoi(filename.substr(0, filename.find(".txt")));
 maxFileNameNumber = max(maxFileNameNumber, fileNameNumber);
 }
 }

 for (int i = start; i <= end; ++i) {
 ifstream inputFile("inputs/" + to_string(i) + ".txt");
 if (!inputFile) {
 cerr << "Could not open the file " + to_string(i) + ".txt" << endl;
 continue;
 }
 }
}

```

```

string line, content;
while (getline(inputFile, line)) {
 content += line + "\n";
}
inputFile.close();

string outputContent;
for (size_t i = 0; i < content.size(); ++i) {
 if (content[i] == '\n') {
 outputContent += "1.27 "; // Assuming 27 is the mapping for
newline character
 continue;
 }
 bool found = false;
 for (int len = 4; len >= 1;--len) {
 if (i + len - 1 < content.size()) {
 string substr = content.substr(i, len);
 if (mappings[len - 1].find(substr) != mappings[len -
1].end()) {
 int number = stoi(mappings[len - 1][substr]);
 outputContent += to_string(len) + "." +
to_string(number + 1) + " ";
 i += len - 1;
 found = true;
 break;
 }
 }
 }
 if (!found) {
 outputContent += "0 ";
 }
}

ofstream outputFile(to_string(maxFileNameNumber + 1) + ".txt");
if (outputFile) {
 outputFile << outputContent;
 outputFile.close();
 maxFileNameNumber++;
} else {
 cerr << "Could not create the file " + to_string(maxFileNameNumber + 1) + ".txt" << endl;
}
}

return 0;
}

```

----- Content of Encoded\_Image.py-----

```
from tkinter import Tk, Label, Button, Entry, filedialog
from PIL import Image
from PIL import ImageDraw
import json
import itertools
import math

class App:
 def __init__(self, master):
 self.master = master
 master.title("Hex Color Picker")

 self.label = Label(master, text="Choose Text File")
 self.label.pack()

 self.open_file_button = Button(master, text="Open File", command=self.open_file)
 self.open_file_button.pack()

 self.close_button = Button(master, text="Generate Image", command=self.generate_image)
 self.close_button.pack()

 def open_file(self):
 file_path = filedialog.askopenfilename()
 with open(file_path, 'r') as file:
 self.data = file.read().split()

 def generate_image(self):
 print("Generating ...")
 if hasattr(self, 'data'):
 self.label.config(text="Generating, please wait...")
 self.master.update_idletasks() # Force update of the GUI

 # Convert data to colors
 color_data = [self.data_to_color(datum) for datum in self.data]

 self.label.config(text="Data converted to colors. Creating image...")
 self.master.update_idletasks() # Force update of the GUI

 create_image(color_data, self.data)

 self.label.config(text="Image successfully generated.")
 self.master.quit()
 else:
 self.label.config(text="Please select a file first.")

 def data_to_color(self, datum):
 # Use hash to convert the string to a unique integer value
 unique_number = hash(datum) & 0xFFFF # Limit to 24 bits to match RGB
 return '#{0:06X}'.format(unique_number)
```

```

def create_image(color_data, data):
 #Count unique string encodings
 unique_encodings = set(data)
 num_unique_encodings = len(unique_encodings)
 total_encodings = len(data) # This will count all the encodings
 print(f"Number of unique encodings: {num_unique_encodings}")
 print(f"Total number of encodings: {total_encodings}") # Added this line
 print(f"If {total_encodings} = A square number, continue, else, reformat the encoded file accordingly..")

pixel_side_length = int(input("Enter the square side length of each pixel to be created.. "))
num_pixels_width = int(input("Enter the pixel width of the image, by number of (custom sized) pixels.. "))
num_pixels_height = num_pixels_width

img_width = pixel_side_length * num_pixels_width
img_height = img_width
Create an empty white canvas
image = Image.new('RGB', (img_width, img_height), color="white")

Create a drawing context for the image
draw = ImageDraw.Draw(image)

data_idx = 0 # Index to track data position
for i in range(num_pixels_height):
 for j in range(num_pixels_width):
 if data_idx < len(color_data):
 hex_value = color_data[data_idx]
 # Define the top-left and bottom-right coordinates for the rectangle
 top_left = (j * pixel_side_length, i * pixel_side_length)
 bottom_right = ((j + 1) * pixel_side_length, (i + 1) * pixel_side_length)
 # Draw the rectangle
 draw.rectangle([top_left, bottom_right], fill=hex_value)
 data_idx += 1

image.show()
image.save('output_image.png')

return image

```

```

print("Output is named 'output_image.png'")
root = Tk()
app = App(root)
root.mainloop()

```

----- Content of Engineering.py-----

```

import tkinter as tk
import random

```

```
from PIL import ImageGrab
import pickle
from tkinter import simpledialog
import os
from tkinter import Menu, Checkbutton, BooleanVar
from tkinter.colorchooser import askcolor
import logging
import time
import json
import os
import pyautogui
import keyboard
from tkinter import filedialog
from PIL import Image, ImageTk

current_col = 0
current_row = 0
resized_images = [] # Global variable to store resized images

Rest of your code...
Global variable for the image selection frame
image_selection_frame = None
selected_image = None # Global variable to hold the currently selected image
Global variable to keep track of the current mode
current_mode = "draw_char" # Possible values: "draw_char", "image_mode"

def toggle_mode():
 global current_mode, canvas
 if current_mode == "draw_char":
 current_mode = "image_mode"
 canvas.unbind("<Button-1>") # Unbind draw_char function
 canvas.bind("<Button-1>", on_canvas_click_for_image) # Bind image drawing function
 else:
 current_mode = "draw_char"
 canvas.unbind("<Button-1>") # Unbind image drawing function
 #canvas.bind("<Button-1>", lambda event: draw_char(event.x // square_size, event.y // square_size))
Bind draw_char function
 canvas.bind("<Button-1>", lambda event: draw_char(event.y // square_size, event.x // square_size))

def select_image(img):
 global selected_image
 selected_image = img
 # You can add additional logic here, e.g., updating the UI to indicate the selected image

def resize_and_add_image(file_path, size):
 try:
 with Image.open(file_path) as img:
 # Resize and add to the list
 #resized_img = img.resize(size, Image.Resampling.LANCZOS) # For Pillow versions 8.0.0 and later
```

```

resized_img = img.resize(size, Image.LANCZOS) # For older versions of Pillow
resized_img = img.resize(size, Image.LANCZOS)
resized_images.append(resized_img)
update_image_selection_area()
except Exception as e:
 print(f"Error loading image: {e}")

"""
def on_canvas_click(event):
 # Calculate the sub-square coordinates
 col = event.x // square_size
 row = event.y // square_size

 if selected_image:
 draw_image_on_canvas(row, col, selected_image)
"""

def on_canvas_click_for_image(event):
 global selected_image
 if selected_image:
 col = event.x // square_size
 row = event.y // square_size
 draw_image_on_canvas(row, col, selected_image)

def draw_image_on_canvas(row, col, image):
 x = col * square_size
 y = row * square_size

 resized_image = image.resize((square_size, square_size), Image.LANCZOS)
 tk_image = ImageTk.PhotoImage(resized_image)

 canvas.create_image(x, y, image=tk_image, anchor='nw')

 if not hasattr(canvas, 'images'):
 canvas.images = []
 canvas.images.append(tk_image) # Keep a reference

"""

Function to update the image selection area
def update_image_selection_area():
 for img in resized_images:
 # Convert PIL image to Tkinter PhotoImage
 tk_image = ImageTk.PhotoImage(img)
 btn = tk.Button(image_selection_frame, image=tk_image, command=lambda img=img:
select_image(img))
 btn.image = tk_image # Keep a reference to avoid garbage collection
 btn.pack(side='left')
"""


```

```

def update_image_selection_area():
 global image_selection_frame

 # Clear existing buttons in the frame
 for widget in image_selection_frame.winfo_children():
 widget.destroy()

 # Create buttons for each image
 for img in resized_images:
 tk_image = ImageTk.PhotoImage(img)
 btn = tk.Button(image_selection_frame, image=tk_image, command=lambda img=img:
select_image(img))
 btn.image = tk_image # Keep a reference to avoid garbage collection
 btn.pack(side='left')

def select_and_add_images():

 # Define a target size for the images
 wid = int(input("Target Width: "))
 hei = int(input("Target Height: "))
 target_size = (wid, hei) # You can change this size as needed

 file_paths = filedialog.askopenfilenames(filetypes=[("PNG files", "*.png")]) # Enable multi-file selection
 if file_paths:
 for file_path in file_paths:
 # Assuming you have a predefined target size for the images
 resize_and_add_image(file_path, target_size)

def resize_and_add_image(file_path, size):
 try:
 with Image.open(file_path) as img:
 # Use Image.Resampling.LANCZOS for Pillow versions 8.0.0 and later
 resized_img = img.resize(size, Image.Resampling.LANCZOS)
 # For older versions of Pillow, use Image.LANCZOS
 # resized_img = img.resize(size, Image.LANCZOS)
 resized_images.append(resized_img)
 update_image_selection_area() # Update the display area with new images
 except Exception as e:
 print(f"Error loading image: {e}")

def select_and_resize_image():
 # Ask the user to select an image file
 file_path = filedialog.askopenfilename(filetypes=[("PNG files", "*.png")])
 if not file_path:
 return # User cancelled the dialog

 # Prompt the user for the new size

```

```

new_size = simpledialog.askstring("Resize Image", "Enter new size (width,height):")
if not new_size:
 return # User cancelled the dialog

try:
 # Parse the size input and resize the image
 width, height = map(int, new_size.split(','))
 resize_image(file_path, (width, height))
except Exception as e:
 tk.messagebox.showerror("Error", f"An error occurred: {e}")

def resize_image(input_path, size):
 try:
 # Construct a new file name based on the original path
 dir_name, file_name = os.path.split(input_path)
 name, ext = os.path.splitext(file_name)
 output_path = os.path.join(dir_name, f"{name}_resized{ext}")

 # Open, resize, and save the image
 with Image.open(input_path) as img:
 # Use Image.Resampling.LANCZOS for Pillow versions 8.0.0 and later
 img = img.resize(size, Image.Resampling.LANCZOS)
 # For older versions of Pillow, use Image.LANCZOS
 # img = img.resize(size, Image.LANCZOS)
 img.save(output_path)

 tk.messagebox.showinfo("Success", f"Image saved successfully to {output_path}")
 except Exception as e:
 tk.messagebox.showerror("Error", f"An error occurred: {e}")

def get_random_color():
 r = lambda: random.randint(0,255)
 return '#%02X%02X%02X' % (r(),r(),r())

def get_random_char():
 random_int = random.randint(0x0021, 0x007E)
 return chr(random_int)

def draw_grid():
 for i in range(grid_size):
 for j in range(grid_size):
 color = get_random_color()
 square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size, (i+1)*square_size, fill=color)
 char = get_random_char()
 text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2, text=char, font=("Arial", 18), anchor="center")

```

```

def draw_grid_IDE():
 global col
 try:
 for i in range(grid_size):
 for j in range(grid_size):
 color = "blue"
 square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size,
(i+1)*square_size, fill=color, outline = "white")
 if typing_mode:
 # In the function where you create the canvas...
 canvas.bind("<Key>", on_key_press)
 canvas.focus_set()
 except:
 print("Submit an app mode ...")

def toggle_color_mode():
 """Toggles between manual color mode and random color mode"""
 global color_mode_var
 color_mode_var.set(not color_mode_var.get())

font_size = 9

def draw_char1(canvas, char, row, col, font_size, color):
 global cell_size, font_color, square_size, grid_size, last_drawn, note, char_note

 ...
 def draw_char1(canvas, char, row, col, font_size):

Draws a character on the Tkinter canvas at the specified row and column with the specified font size.

Parameters:
 canvas (tk.Canvas): The Tkinter canvas to draw on.
 char (str): The character to draw.
 row (int): The row to draw the character at.
 col (int): The column to draw the character at.
 font_size (int): The font size to use for the character.

 ...

try:
 x = col * square_size + square_size / 2
 y = row * square_size + square_size / 2

 print(f"Drawing char '{char}' at ({x}, {y})" # Debug print to check coordinates
 canvas.create_rectangle(col * square_size, row * square_size, (col+1) * square_size, (row+1) *
square_size, fill=color)
 canvas.create_text(x, y, text=char, fill=font_color, font=('Calibri', int(font_size)))

```

```

note = char_note.get()
Log the color and character info
logging.info(f"Position:{x};{y},Xp:{x}',Yp:{y}',Character:{char}',Square color:{color}',Font
color:{font_color}',Note:{note}'")
Log the color and character info in a text file
if(note == ""):
 note = "emptyNote"
with open("color_log.txt", "a", encoding='utf-8') as login:
 print(f"{x},{y},{char},{color},{font_color},{note}", file=login)

except Exception as e:
 print(f"Something Went Wrong... Error: {e}")

def draw_char(i=None, j=None):
 try:
 """Draws a character from input field in a specific square"""
 global char_count, IDE_mode, typing_mode, last_drawn, font_color
 if(mode == 'IDE'):
 IDE_mode = True
 # Check if last_drawn is defined, otherwise define it
 if 'last_drawn' not in globals():
 last_drawn = []
 # If i, j are not provided, calculate them based on char_count
 if i is None or j is None:
 i, j = divmod(char_count, grid_size)
 # Handle out-of-grid situations
 if i >= grid_size or j >= grid_size:
 print('Out of grid!')
 return
 # Generate a random color if not in typing mode, white otherwise
 ##color = get_random_color() if not typing_mode else "white"

 # Check color mode
 if color_mode_var.get():
 # Manual color mode
 # Show a color picker and get the chosen color for the square
 color = askcolor(title="Choose square color")[1]

 # Ask for the font color
 color_result = askcolor(title="Choose font color")
 if color_result is not None:
 font_color = color_result[1]
 else:
 # Handle the case when the user cancelled the color selection
 font_color = "#000000" # default to black, for example

```

```

else:
 # Random color mode
 color = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
 color1 = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
 font_color = color1 "##000000"

square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size, (i+1)*square_size,
fill=color)
char = char_entry.get()[:1]
note = char_note.get()
text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2, text=char,
font=("Arial", font_size), fill=font_color, anchor="center")

...
Log the color and character info
logging.info(f"Position:{i};{j},Xp:{i},Yp:{j},Character:{char},Square color:{color},Font
color:{font_color}")
Log the color and character info in a text file
with open("color_log.txt", "a") as login:
 print(f"{i},{j},{char},{color},{font_color}", file=login)
...
Log the color and character info
logging.info(f"Position:{i};{j},Xp:{i},Yp:{j},Character:{char},Square color:{color},Font
color:{font_color},Note:{note}")
Log the color and character info in a text file
if(note == ""):
 note = "emptyNote"
with open("color_log.txt", "a", encoding='utf-8') as login:
 print(f"{i},{j},{char},{color},{font_color},{note}", file=login)

if len(char_entry.get()) > 0: # Check if there's more than one character
 char_entry.delete(0, 1) # Delete the first character

last_drawn.append((square, text))

if not IDE_mode:
 char_count = (char_count + 1) % (grid_size * grid_size)
 if char_count == 0: # If we've filled the canvas, clear it
 canvas.delete('all')

 return square, text
except:
 print("Submit an app mode ...")

def adjust_grid_and_font():

```

```

global grid_size, square_size, font_size, canvas_width, current_row, current_col

n_g_s = simpledialog.askstring("Change Grid Size (Grid Width)", "Enter New Grid Size (+Integer):")
if(n_g_s != ""):
 new_grid_size = n_g_s
n_f_s = simpledialog.askstring("Change Font Size (Default = 9)", "Enter New Font Size (+Integer):")
if(n_f_s != ""):
 new_font_size = n_f_s
 # Update the global variables
if(new_grid_size==None and new_font_size==None):
 grid_size = 10
 font_size = 9
if((not new_grid_size==None) and (not new_font_size==None)):
 try:
 grid_size = int(new_grid_size)
 square_size = canvas_width / grid_size
 font_size = int(new_font_size)
 except:
 print("positive integers please")

 ...
Get new values from input fields
new_grid_size = grid_size_entry.get()
new_font_size = font_size_entry.get()

Update the global variables
grid_size = int(new_grid_size)
square_size = canvas_width // grid_size
font_size = int(new_font_size)
"""

Redraw the grid
refresh_canvas()
qit = input("Are you in typing mode, or want to enter typing mode? 1 for (No), 2 for (yes): ")
if(qit == '2'):
 current_row = 0
 current_col = 0
 show_typing_mode_menu()

def refresh_canvas():
 """Clears the canvas and resets the char_count"""
 global char_count
 canvas.delete('all')
 char_count = 0
 draw_grid_IDE()
 last_drawn.clear()

def undo_last():
 """Undoes the last drawing operation"""
 if last_drawn:

```

```
square, text = last_drawn.pop()
canvas.delete(square)
canvas.delete(text)

def update_canvas():
 if not paused.get():
 canvas.delete('all')
 draw_grid()
 root.after(8000, update_canvas)

def toggle_typing_mode():
 global typing_mode, canvas

 typing_mode = not typing_mode
 if typing_mode:
 canvas.focus_set() # Set focus to the canvas for keyboard input

def on_canvas_click(event):
 try:
 global mode, typing_mode
 if mode == 'IDE' and not typing_mode:
 j = event.x // square_size
 i = event.y // square_size
 draw_char(i, j)
 else:
 paused.set(not paused.get())
 except:
 print("Submit an app mode ...")

def on_key_press(event):
 global char_count, typing_mode
 if typing_mode:
 char_entry.delete(0, 'end') # Clear the entry box
 try:
 utf8_char = event.char.encode('utf-8').decode('utf-8')
 char_entry.insert(0, utf8_char) # Insert the typed character
 except UnicodeDecodeError:
 print("Non UTF-8 character detected")
 return
 draw_char() # Draw the character
 char_count += 1 # Increment the count

 # If char_count exceeds the total number of squares in the grid, reset it
 if char_count >= grid_size ** 2:
 char_count = 0
```

```
def submit_mode():

 global char_count, mode, typing_mode

 mode = mode_entry.get().upper()

 ...

 if mode in ['editor']:

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 draw_grid()
 char_label.pack()
 char_entry.pack()
 char_button.pack()

 ...

 # Draw the initial grid
 if(mode != 'IDE'):
 print("A valid mode ...")

 ...

 if(mode == 'normal'):

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 draw_grid()
 create_menu_1()

 if mode == 'normal':
 # Schedule the first update
 root.after(8000, update_canvas)
 ...

if mode == 'IDE':

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
```

```
submit_button.pack_forget()

char_count = 0
draw_grid_IDE()

control_frame.pack()
char_label.pack()
char_entry.pack()
#grid_size_label.pack()
#grid_size_entry.pack()
#font_size_label.pack()
#font_size_entry.pack()
#adjust_button.pack()
char_note_label.pack()
char_note.pack()
#refresh_button.pack(side="left")
#undo_button.pack(side="left")
#save_button = tk.Button(root, text='Save', command=save_canvas, bg="white", padx=5, pady=0)
#save_button.pack(side="left")
#save_Sbutton.pack(side="left")
#load_button.pack(side="left")
create_menu()

return mode
```

```
def save_canvas():
 global IC_value, xSwitch, ing
 xSwitch = 1
 ing = 1
 if(ing == 0):
 ing = 2
 ing = 1
 if(xSwitch == 1 and ing == 1):
 IC_value = 0
 ing = 0
 """Save the current state of the canvas to a .png file"""
 #filename = simpledialog.askstring("Save Canvas", "Enter filename:")
 filename = f"ImageCanvas{IC_value}_rename"

if filename: # Only save the canvas if a filename was entered
 # Get the bounding box of the canvas
 x = root.winfo_rootx() + canvas.winfo_x()
 y = root.winfo_rooty() + canvas.winfo_y()
 x1 = x + canvas.winfo_width()
 y1 = y + canvas.winfo_height()
 time.sleep(3)
 # Grab the image, crop it to the bounding box, and save it
```

```
ImageGrab.grab().crop((x, y, x1, y1)).save(filename + ".png")\n\n\ndef save_session():\n """Save the current session to a file using pickle"""\n global last_drawn\n # Ask the user to enter a filename\n filename = simpledialog.askstring("Save Session", "Enter filename:")\n if filename: # Only save the session if a filename was entered\n session_data = [(canvas.coords(square), canvas.itemcget(square, "fill"),\n canvas.coords(text), canvas.itemcget(text, "text"), canvas.itemcget(text, "fill"))\n for square, text in last_drawn]\n with open(filename + ".pkl", "wb") as f:\n pickle.dump(session_data, f)\n\n\ndef load_session():\n """Load a saved session from a file using pickle"""\n global last_drawn\n # Ask the user to enter a filename\n filename = simpledialog.askstring("Load Session", "Enter filename:")\n if filename: # Only try to load a session if a filename was entered\n try:\n with open(filename + ".pkl", "rb") as f:\n session_data = pickle.load(f)\n # Clear the canvas and redraw all elements from the loaded session\n canvas.delete('all')\n draw_grid_IDE()\n last_drawn = []\n for square_coords, square_fill, text_coords, text_content, text_fill in session_data:\n square = canvas.create_rectangle(*square_coords, fill=square_fill)\n text = canvas.create_text(*text_coords, text=text_content, fill=text_fill, font=("Arial", font_size),\n anchor="center")\n last_drawn.append((square, text))\n except FileNotFoundError:\n print(f"No session named '{filename}' found.")\n\n#def exit_app():\n# exit()\n\n\ndef show_about():\n about_window = tk.Toplevel(root)\n about_window.title("About")\n about_msg = "This is a program created to learn and experiment with Tkinter. In IDE mode, you can draw characters on a grid (By typing or copying and pasting then clicking the squares you want to draw each character on), adjust the grid and font size, save and load sessions, and more, like saving the grid as an image (For example). C(num) was written with the aid of Chat GPT. Enjoy!\n\nNote: the workflow ... IDE to color_log.txt (Compiled by compile#.exe or compile#.py);\\nindex_1.txt contains fields of study or research and development and\\nindex_2.txt contains fields of study or research and development also ...\\n\nBertotools Digital"\n
```

```
tk.Message(about_window, text=about_msg, width=500).pack()
tk.Button(about_window, text="OK", command=about_window.destroy).pack()

def compileGOSmX():
 MAX_LINE_LENGTH = 256
 MAX_TOPICS = 256

 global index_1, index_2

 index_1 = [
 "Axiom",
 "Theorem",
 "Lemma",
 "Proposition",
 "Corollary",
 "Conjecture",
 "Proof",
 "Premise",
 "Conclusion",
 "Hypothesis",
 "Counterexample",
 "Direct Proof",
 "Indirect Proof",
 "Proof by Contradiction (Reductio ad absurdum)",
 "Proof by Induction",
 "Proof by Contrapositive",
 "Deductive Reasoning",
 "Inference",
 "Assumption",
 "Statement",
 "Postulate",
 "Proof by Exhaustion",
 "Syllogism",
 "Constructive Proof",
 "Non-Constructive Proof",
 "Trivial Proof",
 "Vacuous Proof",
 "Biconditional",
 "Condition",
 "Sufficiency",
 "Necessity",
 "Quantifier",
 "Universal Quantifier",
 "Existential Quantifier",
 "Bound Variable",
 "Free Variable",
 "Predicate",
 "Propositional Logic",
 "Modus Ponens",
 "Modus Tollens",
```

"Discrete Mathematics",  
"Set Theory",  
"Function",  
"Bijection",  
"Injection",  
"Surjection",  
"Equivalence Relation",  
"Partial Order",  
"Total Order",  
"Well-Order",  
"Reflexivity",  
"Symmetry",  
"Transitivity",  
"Antisymmetry",  
"Completeness",  
"Compactness",  
"Connectedness",  
"Convergence",  
"Divergence",  
"Limit",  
"Sequence",  
"Series",  
"Monotonicity",  
"Cauchy Sequence",  
"Infinite Set",  
"Finite Set",  
"Cardinality",  
"Countable Set",  
"Uncountable Set",  
"Subset",  
"Superset",  
"Intersection",  
"Union",  
"Empty Set",  
"Power Set",  
"Cartesian Product",  
"Equivalence Class",  
"Partition",  
"Field",  
"Ring",  
"Group",  
"Abelian Group",  
"Non-abelian Group",  
"Matrix",  
"Vector Space",  
"Linear Transformation",  
"Eigenvalue",  
"Eigenvector",  
"Norm",  
"Inner Product",

"Orthogonality",  
"Basis",  
"Dimension",  
"Rank",  
"Nullity",  
"Determinant",  
"Graph Theory",  
"Vertex",  
"Edge",  
"Connectivity",  
"Cycle",  
"Path",  
"Degree",  
"Subgraph",  
"Tree",  
"Forest",  
"Planar Graph",  
"Bipartite Graph",  
"Directed Graph (Digraph)",  
"Eulerian Graph",  
"Hamiltonian Graph",  
"Adjacency Matrix",  
"Incidence Matrix",  
"Isomorphism",  
"Homeomorphism",  
"Topology",  
"Open Set",  
"Closed Set",  
"Boundary",  
"Compact Space",  
"Hausdorff Space",  
"Continuity",  
"Differential",  
"Derivative",  
"Integral",  
"Partial Derivative",  
"Multivariable Calculus",  
"Laplace Transform",  
"Fourier Transform",  
"Taylor Series",  
"Maclaurin Series",  
"Conic Sections",  
"Hyperbola",  
"Ellipse",  
"Parabola",  
"Asymptote",  
"Limits at Infinity",  
"Complex Number",  
"Imaginary Unit",  
"Real Number",

"Rational Number",  
"Irrational Number",  
"Prime Number",  
"Composite Number",  
"GCD (Greatest Common Divisor)",  
"LCM (Least Common Multiple)",  
"Permutation",  
"Combination",  
"Probability",  
"Statistics",  
"Expected Value",  
"Variance",  
"Standard Deviation",  
"Normal Distribution",  
"Poisson Distribution",  
"Binomial Distribution",  
"Hypothesis Testing",  
"Regression",  
"Correlation",  
"Matrix Algebra",  
"Linear Algebra",  
"Vector Calculus",  
"Optimization",  
"Algorithm",  
"Computational Complexity",  
"Big O Notation",  
"Pigeonhole Principle",  
"Principle of Inclusion-Exclusion",  
"Turing Machine",  
"Computability",  
"Unsolvability",  
"Parity",  
"Diophantine Equations",  
"Cryptography",  
"Fermat's Last Theorem",  
"Pythagorean Theorem",  
"Triangle Inequality",  
"Trigonometric Functions",  
"Trigonometric Identities",  
"Polar Coordinates",  
"Euler's Formula",  
"Riemann Zeta Function",  
"P vs NP Problem",  
"NP-complete Problem",  
"Stochastic Process",  
"Markov Chain",  
"Random Variable",  
"Conditional Probability",  
"Bayes' Theorem",  
"Monte Carlo Method",

"Fractal",  
"Chaos Theory",  
"Game Theory",  
"Nash Equilibrium",  
"Zero-Sum Game",  
"Non-Zero-Sum Game",  
"Linear Programming",  
"Nonlinear Programming",  
"Quadratic Programming",  
"Dynamic Programming",  
"Integer Programming",  
"Graph Coloring",  
"Network Flow",  
"Spanning Tree",  
"Bellman-Ford Algorithm",  
"Dijkstra's Algorithm",  
"Kruskal's Algorithm",  
"Prim's Algorithm",  
"Floyd-Warshall Algorithm",  
"Euler's Method",  
"Runge-Kutta Method",  
"Numerical Integration",  
"Numerical Differentiation",  
"Bisection Method",  
"Newton's Method",  
"Secant Method",  
"Fixed Point Iteration",  
"Linear Interpolation",  
"Polynomial Interpolation",  
"Lagrange Interpolation",  
"Splines",  
"Fourier Series",  
"Laplace's Equation",  
"Heat Equation",  
"Wave Equation",  
"Schrodinger Equation",  
"Ordinary Differential Equation (ODE)",  
"Partial Differential Equation (PDE)",  
"Boundary Value Problem",  
"Initial Value Problem",  
"Green's Theorem",  
"Stoke's Theorem",  
"Divergence Theorem",  
"Curl",  
"Gradient",  
"Divergence",  
"Tensor",  
"Manifold",  
"Topological Space",  
"Measure Theory",

"Lebesgue Integral",  
"Borel Set",  
"Hilbert Space",  
"Banach Space",  
"Category Theory",  
"Functor",  
"Natural Transformation",  
"Sheaf",  
"Homotopy",  
"Homology",  
"Cohomology",  
"Galois Theory",  
"Algebraic Geometry",  
"Topological K-Theory",  
"Knot Theory",  
"Lattice Theory"  
]

index\_2 = [  
    "Biochemistry",  
    "Biophysics",  
    "Molecular biology",  
    "Genetics",  
    "Immunology",  
    "Cell biology",  
    "Microbiology",  
    "Neuroscience",  
    "Pharmacology",  
    "Bioinformatics",  
    "Biotechnology",  
    "Proteomics",  
    "Genomics",  
    "Structural biology",  
    "Virology",  
    "Systems biology",  
    "Developmental biology",  
    "Evolutionary biology",  
    "Synthetic biology",  
    "Metabolomics",  
    "Epigenetics",  
    "Tissue engineering",  
    "Nanotechnology",  
    "Materials science",  
    "Quantum physics",  
    "Condensed matter physics",  
    "Particle physics",  
    "Astrophysics",  
    "Cosmology",  
    "Optics",  
    "Atomic and molecular physics",

"Fluid mechanics",  
"Thermodynamics",  
"Environmental science",  
"Climate science",  
"Geology",  
"Oceanography",  
"Atmospheric science",  
"Ecology",  
"Conservation biology",  
"Botany",  
"Zoology",  
"Entomology",  
"Marine biology",  
"Paleontology",  
"Anthropology",  
"Archaeology",  
"Psychology",  
"Cognitive science",  
"Social psychology",  
"Linguistics",  
"Artificial intelligence",  
"Machine learning",  
"Computer vision",  
"Natural language processing",  
"Human-computer interaction",  
"Robotics",  
"Computer graphics",  
"Data science",  
"Mathematical modeling",  
"Mathematical physics",  
"Number theory",  
"Algebraic geometry",  
"Differential equations",  
"Computational physics",  
"Mathematical biology",  
"Operations research",  
"Biostatistics",  
"Epidemiology",  
"Cancer research",  
"Diabetes research",  
"Heart disease research",  
"Infectious diseases research",  
"Immunotherapy",  
"Stem cell research",  
"Gene therapy",  
"Drug discovery",  
"Precision medicine",  
"Health informatics",  
"Renewable energy",  
"Energy storage",

"Sustainable materials",  
"Environmental engineering",  
"Water management",  
"Transportation engineering",  
"Civil engineering",  
"Chemical engineering",  
"Aerospace engineering",  
"Biomedical engineering",  
"Electrical engineering",  
"Mechanical engineering",  
"Robotics engineering",  
"Quantum computing",  
"Cryptography",  
"Cybersecurity",  
"Network engineering",  
"Telecommunications",  
"Human genetics",  
"Forensic science",  
"Space exploration and research",  
"Planetary science",  
"Astrobiology",  
"Astrochemistry",  
"Astrogeology",  
"Astroinformatics",  
"Exoplanet research",  
"Stellar evolution",  
"Galactic astronomy",  
"Observational astronomy",  
"Computational astrophysics",  
"Quantum chemistry",  
"Computational chemistry",  
"Organic chemistry",  
"Inorganic chemistry",  
"Physical chemistry",  
"Environmental chemistry",  
"Analytical chemistry",  
"Agricultural science",  
"Food science",  
"Nutritional science",  
"Exercise physiology",  
"Sports science",  
"Biomechanics",  
"Plant physiology",  
"Plant genetics",  
"Plant pathology",  
"Soil science",  
"Hydrology",  
"Geochemistry",  
"Geophysics",  
"Geomorphology",

"Remote sensing",  
"Geotechnical engineering",  
"Petroleum engineering",  
"Aerospace materials",  
"Nanomaterials",  
"Polymer science",  
"Computational materials science",  
"Photonics",  
"Physical optics",  
"Quantum optics",  
"Neuroengineering",  
"Brain imaging",  
"Cognitive neuroscience",  
"Neuroinformatics",  
"Psychophysics",  
"Developmental psychology",  
"Personality psychology",  
"Clinical psychology",  
"Industrial-organizational psychology",  
"Educational psychology",  
"Psycholinguistics",  
"Human genetics",  
"Evolutionary genetics",  
"Population genetics",  
"Genetic engineering",  
"Genetic counseling",  
"Epigenomics",  
"Cardiovascular research",  
"Respiratory research",  
"Gastroenterology research",  
"Endocrinology research",  
"Nephrology research",  
"Hematology research",  
"Ophthalmology research",  
"Orthopedic research",  
"Dermatology research",  
"Veterinary medicine",  
"Animal behavior",  
"Conservation ecology",  
"Wildlife biology",  
"Environmental microbiology",  
"Agricultural economics",  
"Development economics",  
"Behavioral economics",  
"Econometrics",  
"Financial mathematics",  
"Operations management",  
"Supply chain management",  
"Industrial engineering",  
"Human-computer interaction",

"Virtual reality",  
"Augmented reality",  
"Data mining",  
"Text mining",  
"Big data analytics",  
"Computational linguistics",  
"Quantum information science",  
"Quantum cryptography",  
"Biometrics",  
"Information retrieval",  
"Software engineering",  
"Computer networks",  
"Embedded systems",  
"Human-robot interaction",  
"Control systems",  
"Biopharmaceuticals",  
"Drug delivery systems",  
"Clinical trials",  
"Regenerative medicine",  
"Agricultural biotechnology",  
"Plant breeding",  
"Animal breeding",  
"Food technology",  
"Sensory science",  
"Poultry science",  
"Aquaculture",  
"Marine ecology",  
"Limnology",  
"Population ecology",  
"Landscape ecology",  
"Evolutionary ecology",  
"Environmental toxicology",  
"Environmental chemistry",  
"Environmental microbiology",  
"Ecotoxicology",  
"Green chemistry",  
"Space physics",  
"Space weather",  
"Astrostatistics",  
"Computational fluid dynamics",  
"Mathematical optimization",  
"Operations research",  
"Human genetics",  
"Functional genomics",  
"Molecular genetics",  
"Cancer genetics",  
"Psychiatric genetics",  
"Population genomics",  
"Bioengineering",  
"Biomaterials",

```
"Biomechatronics",
"Cardiovascular engineering",
"Neural engineering",
"Rehabilitation engineering",
"Genetic engineering",
"Environmental engineering",
"Water resources engineering",
"Structural engineering",
"Robotics engineering",
"Quantum information theory",
"Quantum simulation",
"Quantum sensing",
"Geographical information systems (GIS)",
"Urban planning",
"Renewable energy systems",
"Solar cell technology",
"Wind energy research",
"Energy policy and economics",
"Computational neuroscience",
"Neurobiology",
"Cognitive neuroscience",
"Systems neuroscience",
"Human-robot interaction",
"Evolutionary psychology",
"Social network analysis"
]
```

```
def square_print_topic_indices(r_value, g_value, b_value, file):
 index1 = r_value % len(index_1)
 index2 = g_value % len(index_1)
 index3 = b_value % len(index_1)
 ...
 file.write(f"\t\t\t{topics[index1]}\n")
 file.write(f"\t\t\t{topics[index2]}\n")
 file.write(f"\t\t\t{topics[index3]}\n")
 ...
 file.write(f"\t\t\t{index_1[index1]}\n")
 file.write(f"\t\t\t{index_1[index2]}\n")
 file.write(f"\t\t\t{index_1[index3]}\n")
```

```
squareRatio = str(index1) + ":" + str(index2) + ":" + str(index3)
return squareRatio
```

```
def font_print_topic_indices(r_value, g_value, b_value, file):
 index1_ = r_value % len(index_2)
 index2_ = g_value % len(index_2)
 index3_ = b_value % len(index_2)
 file.write(f"\t\t\t{index_2[index1_]}\n")
```

```

file.write(f"\t\t\t{index_2[index2_]}\n")
file.write(f"\t\t\t{index_2[index3_]}\n")

fontRatio = str(index1_) + ":" + str(index2_) + ":" + str(index3_)
return fontRatio

def get_index(i, j, grid_size):
 return i * grid_size + j

def extract_rgb_values(color):
 if len(color) != 7 or color[0] != '#':
 raise ValueError('Input should be a hex color code in the format "#RRGGBB"')
 try:
 red = int(color[1:3], 16)
 green = int(color[3:5], 16)
 blue = int(color[5:7], 16)
 return red, green, blue
 except ValueError:
 raise ValueError('Invalid color code. RGB values should be hex digits (0-9, A-F)')

def GO():
 log_file = "color_log.txt"
 ## topic_file = "index_1.txt"
 ## topic2_file = "index_2.txt" # Corrected this line to read from a different file
 gridWidth = int(input("Enter gridWidth: "))
 # Read topic files to get the topics and topic2s
 ## with open(topic_file, "r", encoding='utf8') as file:
 ## topics = [line.strip() for line in file]
 ##
 ## with open(topic2_file, "r", encoding='utf8') as file:
 ## topic2s = [line.strip() for line in file]

 numTopics = len(index_1)
 numTopic2s = len(index_2)

 # Read color log file
 with open(log_file, "r") as file:
 lines = file.readlines()

 lineNumber = 0
 # Save output to a file
 output_file = "output.txt"
 with open(output_file, "w") as file:
 print()
 for line in lines:
 parts = line.strip().split(",")
 if len(parts) >= 5:

```

```

xcoor = int(parts[1])
ycoor = int(parts[0])
index_ = (gridWidth * ycoor) + xcoor
character = parts[2]
squareColor = parts[3]
fontColor = parts[4]
notes = parts[5]

Extract RGB values from squareColor
red, green, blue = extract_rgb_values(squareColor)

Extract RGB values from fontColor
fontRed, fontGreen, fontBlue = extract_rgb_values(fontColor)

Calculate the grid size
grid_size = int(parts[0])

Select topics based on RGB values from squareColor
topicIndices = [(red + get_index(i, j, grid_size)) % numTopics for i in range(grid_size) for j in range(grid_size)]

lineNumber += 1

with open(output_file, "a") as file:
 file.write(f"\nCompiling Line/s: {lineNumber}\n\n")
 file.write(f"Line({lineNumber})\n\n")
 file.write(f"0\tLineNumber: {lineNumber}\ttx-coordinate: {xcoor}\ty-coordinate: {ycoor}\t"
Character: {character}\tRGB: {squareColor}\tRGB: {fontColor}\t Note: {notes}\t GridDimensions: "
{gridWidth}x{gridWidth}\n\n")
 #print(topicIndices)
 file.write(f"Line/s({lineNumber}) Output:\n\n")
 file.write(f"#define Line({lineNumber}){\n\n")
 # Inside the main function
 file.write(f"\ti_0({squareColor}){\n\n")
 file.write(f"\t{character}\n")
 file.write(f"\t\tR: {red}, G: {green}, B: {blue})\n")
 s_ratio = square_print_topic_indices(red, green, blue, file)
 #square_print_topic_indices(red, green, blue, topics, file) # Modify this line
 file.write("\t\t)\n")
 file.write("\t;\n\n")
 file.write(f"\ti_1({fontColor}){\n\n")
 file.write(f"\t{character}\n")
 file.write(f"\t\tR: {fontRed}, G: {fontGreen}, B: {fontBlue})\n")
 f_ratio = font_print_topic_indices(fontRed, fontGreen, fontBlue, file)
 #font_print_topic_indices(fontRed, fontGreen, fontBlue, topic2s, file) # And this line
 file.write("\t\t)\n")
 file.write(f"\treturn squareRatio({s_ratio}), fontRatio({f_ratio}), squareIndex({index_})\n")
 #file.write(f"\treturn fontRatio({f_ratio})\n")
 file.write("}\n\n")

```

```
GO()

def create_menu():
 global control_frame
def home():
 canvas.pack_forget()
 menubar.destroy()
 control_frame.pack()
 char_entry.pack_forget()
 char_label.pack_forget()
 #grid_size_label.pack_forget()
 #grid_size_entry.pack_forget()
 #font_size_label.pack_forget()
 #font_size_entry.pack_forget()
 #adjust_button.pack_forget()
 char_note.pack_forget()
 char_note_label.pack_forget()

 root.destroy()
 switch = 1
 setup_mode(switch)

Create a menubar
menubar = tk.Menu(root)

Create an options menu and add it to the menubar
options_menu = tk.Menu(menubar, tearoff=0)
#menubar.add_cascade(label="Options", menu=options_menu)

Create a dropdown menu and add it to the menubar
dropdown = tk.Menu(menubar, tearoff=0)
menubar.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"

Add commands to dropdown menu

Add checkbutton to options menu
dropdown.add_command(label="Toggle Manual/Random Colour", command=manual_colour)

dropdown.add_command(label="Refresh", command=refresh_canvas)
dropdown.add_command(label="Undo", command=undo_last)
dropdown.add_command(label="Save Session", command=save_session)
dropdown.add_command(label="Load Session", command=load_session)
dropdown.add_command(label="Save as Image", command=save_canvas)
dropdown.add_command(label="Compile Canvas", command=compileGOSmX)
Add the new command for resizing images
dropdown.add_command(label="Select and Resize Image", command=select_and_resize_image)
Add the command for selecting images
```

```
dropdown.add_command(label="Select Images", command=select_and_add_images)
Add the toggle mode command
dropdown.add_command(label="Toggle Draw Chars/Images", command=toggle_mode)
#dropdown.add_command(label="Exit", command=exit_app)
#dropdown.add_command(label="Toggle Type/Click Mode", command=toggle_typing_mode)
dropdown.add_command(label="About Program", command=show_about)
dropdown.add_command(label="Go Home", command=home)
dropdown.add_command(label="Edit Grid Dimensions", command=adjust_grid_and_font)
dropdown.add_command(label="Typing Mode", command=show_typing_mode_menu)

Set the menubar
root.config(menu=menubar)

def create_menu_1():
 global control_frame
 def home1():
 canvas.pack_forget()
 menubar1.destroy()
 root.destroy()
 switch = 1
 setup_mode(switch)

 # Create a menubar
 menubar1 = tk.Menu(root)

 # Create an options menu and add it to the menubar
 options_menu1 = tk.Menu(menubar1, tearoff=0)
 #menubar.add_cascade(label="Options", menu=options_menu)

 # Create a dropdown menu and add it to the menubar
 dropdown = tk.Menu(menubar1, tearoff=0)
 menubar1.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"

 # Add commands to dropdown menu

 # Add checkbutton to options menu
 dropdown.add_command(label="About Program", command=show_about)
 dropdown.add_command(label="Change Mode", command=home1)

 # Set the menubar
 root.config(menu=menubar1)

def manual_colour():
 # Toggle the value of color_mode_var
 current_value = color_mode_var.get()
```

```
color_mode_var.set(not current_value)

"""

Create a tkinter window
root = tk.Tk()
root.title("General Operating System")
root.iconbitmap("logo.ico")
"""

def show_typing_mode_menu():
 global color_
 while True:
 print("1. Use default UTF-8 mapping")
 print("4. Exit")
 choice = input("Choose an option: ")

 if choice == "1":
 # Use default UTF-8 mapping
 print("Using default UTF-8 mapping.")
 color_ = input("Use Default Colour Scheme (1): ")
 if(color_ == '1'):
 # Initialize bg_color and font_color with default values at the global scope
 bg_color = "#FFFFFF" # Default white background color
 font_color = "#000000" # Default black font color
 break
 else:
 print("Invalid choice. Please try again.")

 canvas.bind("<Key>", type_character)
 canvas.focus_set()

def set_background_color():
 global color_
 #bg_color = tk.colorchooser.askcolor()[1]
 color = input("Enter background color (#FFFFFF for example):")

def set_font_color():
 global font_color, font_size
 #font_color = tk.colorchooser.askcolor()[1]
 font_color = input("Enter background color (#000000 for example): ")
 font_size = input("Enter font_size: ")

def type_character(event):
 global current_row, current_col, col, font_size, font_color, color, canvas_width, square_size
```

```
color = "#FFFFFF"
font_color = "#000000"
char = event.char
if char: # Ignore non-character events
 try:
 draw_char1(canvas, char, current_row, current_col, font_size, color)
 current_col += 1
 if (current_col >= canvas_width/square_size):
 current_col = 0
 current_row += 1
 if(current_row >= canvas_width/square_size):
 current_col = 0
 current_row = 0
 except Exception as e:
 print("Something Went Wrong...")
 print(e)
```

```
def setup_mode(sw):
 global mode_entry, mode_label, square_size, grid_size, char_count, last_drawn
 global root, color_mode_var, logging, paused, canvas_width, canvas_height, canvas
 global grid_size_label, grid_size_entry, font_size_label, font_size_entry, adjust_button
 global save_Sbutton, load_button, char_label, char_entry, char_note_label, char_note, char_button
 global refresh_button, undo_button, control_frame, submit_button, col, color_
 global image_selection_frame # Use the global variable

 current_row = 0
 current_col = 0
 color_ = 1

 # Define the size of the squares and the grid
 #square_size = 70
 square_size = int(input("Enter sub-square size: "))
 #grid_size = 10
 grid_size = int(input("Enter grid size by the number of sub-squares: "))
 char_count = 0
 last_drawn = []

 window_height = square_size * grid_size + (square_size * 2)
 window_width = square_size * grid_size

 if(sw == 1):
```

```
Create a tkinter window
root = tk.Tk()
Define the frame for displaying the image selection
image_selection_frame = tk.Frame(root)
image_selection_frame.pack(side='top') # Adjust the side as per your layout
root.title("C")
#root.iconbitmap("logo.ico")
sw = 0
switch = 0

#####
Get screen width and height
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

Calculate position of the window
position_top = int(screen_height / 2 - window_height / 2)
position_right = int(screen_width / 2 - window_width / 2)

Set the dimensions of the window and where it is placed
root.geometry(f'{window_width}x{window_height}+{position_right}+{position_top}')

#####
Create a BooleanVar for the color mode and set it to False initially
color_mode_var = BooleanVar(value=False)

Setup logging
logging.basicConfig(filename='logging.txt', level=logging.INFO)

paused = tk.BooleanVar()
paused.set(False)

canvas_width = grid_size * square_size
canvas_height = grid_size * square_size
canvas = tk.Canvas(root, width=canvas_width, height=canvas_height)
col = canvas_width
canvas.pack()
canvas.bind("<Button-1>", on_canvas_click)

In the function where you create the canvas...
canvas.bind("<Key>", on_key_press)
canvas.focus_set()

Create input fields and buttons for grid and font size input
grid_size_label = tk.Label(root, text="Grid size:")
grid_size_entry = tk.Entry(root, width=5)
font_size_label = tk.Label(root, text="Font size:")
```

```

font_size_entry = tk.Entry(root, width=5)
adjust_button = tk.Button(root, text='Adjust Grid & Font', command=adjust_grid_and_font)

Create the save and load buttons
save_Sbutton = tk.Button(root, text='Save Session', command=save_session, bg="white", padx=5,
pady=0)
load_button = tk.Button(root, text='Load Session', command=load_session, bg="white", padx=5, pady=0)

Create input fields and buttons for mode and character input
mode_label = tk.Label(root, text="Enter ... 'IDE' for an\nInteractive Development Environment\nfor
Learning anything!")
mode_entry = tk.Entry(root)
submit_button = tk.Button(root, text='LEARN', command=submit_mode)

"""

filename = simpledialog.askstring("Save Canvas", "Enter filename:")
if filename:
"""

char_label = tk.Label(root, text="Enter symbol/s to draw: ")
char_entry = tk.Entry(root, width=12)
char_note_label = tk.Label(root, text="Enter Note: ")
char_note = tk.Entry(root, width=12)
#char_button = tk.Button(root, text='Draw Character', command=draw_char)

refresh_button = tk.Button(root, text='Refresh', command=refresh_canvas, bg="white", padx=5, pady=0)
undo_button = tk.Button(root, text='Undo', command=undo_last, bg="white", padx=5, pady=0)

control_frame = tk.Frame(root)

mode_label.pack()
mode_entry.pack()
submit_button.pack()
#create_menu(root)

Run the tkinter main loop
root.mainloop()

global switch
switch = 1

setup_mode(switch)

```

----- Content of Enumeration\_of\_DSG.cpp-----

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

```

```

#include <iostream>
using namespace std;
//put together by Dominic Alexander Cooper
int main(){
 puts("(pt) == Universal Set = Computer");
 puts("(pt)i = sub-set of (pt)");
 puts("(pt)ia = element of sub-set of (pt)");
 puts("ti = ia");
 puts("t = i of (pt)ia ");
 puts("i = a of ((pt)ia)");
 puts("\nProgressive Abstract Numerical Solution");
 //Code Adapted by DAC from lyst on https://stackoverflow.com
 //k+1 = no. of elements
 //n = exponent = number of cells
 //the k and n values must perfectly fit the size of the set of elements in question
 FILE *p; p = fopen("PROGRESSIVE_ANS_px_RENAME.txt","w");
 //char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
 '\n','\t','\\','\"','/','<','>','?','!','@','#','~','[','{','}','`','!','!','!','!','!','!','!','!','!','!','!','!','!','!','!','!','_','_',
 '+','=,'!','A','B','C','D','E','F','G','H',
 'I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9'};
 int a[] =
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,3
8,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,
73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,1
05,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,129,
130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,15
4,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,1
79,180,181,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,
204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,22
8,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,2
53,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,
278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,30
2,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,3
27,328,329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,346,347,348,349,350,351,
352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,373,374,375,37
6,377,378,379,380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,397,398,399,400,4
01,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,423,424,425,
426,427,428,429,430,431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,448,449,45
0,451,452,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,469,470,471,472,473,474,4
75,476,477,478,479,480,481,482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,499,
500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521,522,523,52
4,525,526,527,528,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,5
49,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,
574,575,576,577,578,579,580,581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597,59
8,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,618,619,620,621,622,6
23,624,625,626,627,628,629,630,631,632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,
648,649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665,666,667,668,669,670,671,67
2,673,674,675,676,677,678,679,680,681,682,683,684,685,686,687,688,689,690,691,692,693,694,695,696,6
97,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721,
722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,74
```

```

6,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,7
71,772,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,
796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,82
0,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,8
45,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,
870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,89
4,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,9
19,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,
944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,96
8,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,9
93,994,995,996,997,998,999;

int pin;

int pr, pc;
printf("\nLet us begin\n\n");
pc = 1000;
for(pr = 0; pr < pc; pr++){
 printf("%d %d\n", pr, a[pr]);
}
char choice;
cout << "Enter a to uniquely use the 1 to 1000 identifiers for the Default Library or c to create a
custom Library: ";
cin >> choice;

cout << "Enter the size of your array: ";
cin >> pin;

int array[pin], inn, position;
int generate;
cout << "\nDo you want manually create your array (0) or, do you want to generate your array for
a range of numbers (1): ";
cin >> generate;
int lbrange;
int ubrange;
//int i_lbrange, i_ubrange;
int r_check;
r_check = 0;

if(choice == 'c'){
 //cout << "Enter the size of your array: ";
 //cin >> pin;
 //int array[pin], inn, position;

 if(generate == 0){
 cout << "\n\n Enter the " << pin << " elements of your array: \n\n";
 for(inn = 0; inn < pin; inn++){
 cin >> array[inn];
 }
 }
}

```

```

 }
 if(generate == 1){
 cout << "\nEnter the lower inclusive bound of the array range of numbers:
";
 cin >> lrange;
 //i_lrange = int(lrange);
 cout << "\nEnter the upper inclusive bound of the array range of numbers:
";
 cin >> ubrange;
 //i_ubrange = ubrange;
 r_check = lrange;
 //generate custom array from user input;
 for(inn = 0; inn < pin; inn++){
 //
 r_check = r_check + 1;
 array[inn] = r_check;
 printf("\n%d", r_check);
 r_check = r_check + 1;
 if (r_check > ubrange){
 break;
 }
 }
 }

 //
 for(inn = 0; inn < pin; inn++){
 cin >> array[inn];
 }
 }

//char a[] = {'1','2','3','4','p1','p2','p3','p4','p5','p6'};
//char b[] = {'a','b'};
//char c[] = {'0','1','2','3'};
//int n = 4; //int k = 3; //int n = x;
//int k = 100;

//int k = strlen(a)- 1;
int k;
if(choice == 'c'){
 k = pin- 1;
}
if(choice == 'a'){
 k = 1000- 1;
}
printf("\n\tnk = %d", k);
int noc; printf("\n\tn = ");
scanf("%d", &noc);
printf("\n\tNumber Of FILE Cells = %d", noc);
int n = noc;
int row, col;
int cell;
int rdiv;

```

```

int id;
id = 0;
int rin;
int nbr_comb = pow(k+1, n);
int ai, ci, switch_c, switch_a;

if(choice == 'c'){

 switch_c = 0;
 cout << "\n\nEnter nth File System (Number of Cells Per File of the nth File System). Let
the value be equal to n for a progressionless session: ";
 cin >> ci;

 for(ci; n < ci + 1; n++){

 if(switch_c == 1){
 nbr_comb = pow(k + 1, n);
 row = 0;
 }

 for (row=0; row < nbr_comb; row++){
 id++;
 fprintf(p, "\n\n%d\t%d\n", n,id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);

 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 if(col == 0){
 fprintf(p,"%d", a[rin]);
 }
 if(col != 0){
 fprintf(p,"%d ", a[rin]);
 }
 //fprintf(p, "%d", a[cell]);
 }
 //printf("\n");
 }
 switch_c = 1;
 }
}

int check;
check = pin- 1000;

```

```

if(choice == 'a' && check == 0){

 switch_a = 0;
 cout << "\n\nEnter nth File System (Number of Cells Per File of the nth File System). Let
the value be equal to n for a progressionless session: ";
 cin >> ai;

 for(n; n < ai + 1; n++){
 if(switch_a == 1){
 nbr_comb = pow(k + 1, n);
 row = 0;
 }
 for (row=0; row < nbr_comb; row++){
 id++;
 fprintf(p, "\n\n%dCF%d\n", n, id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p, "%c", a[cell]);

 cell = (row/rdiv) % (k+1);
 //rin = array[cell];
 if(col == 0){
 fprintf(p, "%d", a[cell]);
 }
 if(col != 0){
 fprintf(p, "%d ", a[cell]);
 }
 //fprintf(p, "%d", a[cell]);
 }
 //printf("\n");
 }
 switch_a = 1;
 }

 //fprintf(p, "\n\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
 fclose(p);
 //end of adaptation
 return 0;
}

```

----- Content of ETA.py-----

```

import os
import subprocess

def main_menu():
 print("\nEnhanced Terminal Application")
 print("1. Execute Command")
 print("2. File System Operations")

```

```
print("9. Exit")
choice = input("Enter choice: ")
return choice

def execute_command():
 command = input("Enter command: ")
 try:
 result = subprocess.run(command, shell=True, capture_output=True, text=True)
 print(result.stdout)
 except Exception as e:
 print(f"An error occurred: {e}")

def file_system_operations():
 print("File System Operations:")
 print("1. List Directory Contents")
 print("2. Change Directory")
 print("3. Create a Directory")
 print("4. Delete a Directory")
 print("9. Return to Main Menu")
 choice = input("Enter choice: ")
 return choice

def list_directory_contents():
 path = input("Enter directory path: ")
 try:
 contents = os.listdir(path)
 for item in contents:
 print(item)
 except FileNotFoundError:
 print("Directory not found.")

def change_directory():
 path = input("Enter path to change to: ")
 try:
 os.chdir(path)
 print(f"Changed to directory: {os.getcwd()}")
 except FileNotFoundError:
 print("Directory not found.")

def create_directory():
 path = input("Enter directory path to create: ")
 try:
 os.makedirs(path, exist_ok=True)
 print(f"Directory created: {path}")
 except OSError as e:
 print(f"An error occurred: {e}")

def delete_directory():
 path = input("Enter directory path to delete: ")
 try:
```

```

os.rmdir(path)
print(f"Directory deleted: {path}")
except OSError as e:
 print(f"An error occurred: {e}")

if __name__ == "__main__":
 while True:
 user_choice = main_menu()
 if user_choice == "1":
 execute_command()
 elif user_choice == "2":
 while True:
 fs_choice = file_system_operations()
 if fs_choice == "1":
 list_directory_contents()
 elif fs_choice == "2":
 change_directory()
 elif fs_choice == "3":
 create_directory()
 elif fs_choice == "4":
 delete_directory()
 elif fs_choice == "9":
 break
 break
 elif user_choice == "9":
 break

```

----- Content of extract1.py-----

```

import os
import re

def extract_characters_from_file(file_path):
 with open(file_path, 'r', encoding='utf-8') as file:
 return re.findall(r'\b[U\+0-9A-F]{4,5}\b', file.read())

def write_characters_to_file(characters, output_file_path):
 with open(output_file_path, 'w', encoding='utf-8') as file:
 for char in characters:
 # Convert Unicode code point to character
 file.write(chr(int(char[2:], 16)) + '\n')

def main():
 directory = '.' # Specify the directory where your files are located
 output_file_path = 'charSF1.txt'
 all_characters = []

 for filename in os.listdir(directory):
 if filename.startswith('Unihan_Readings') and filename.endswith('.txt'):
 file_path = os.path.join(directory, filename)
 characters = extract_characters_from_file(file_path)

```

```

all_characters.extend(characters)

write_characters_to_file(set(all_characters), output_file_path)
print("Characters extracted and saved to", output_file_path)

if __name__ == "__main__":
 main()

----- Content of extract2.py-----
import os
import re

def extract_characters_from_file(file_path):
 with open(file_path, 'r', encoding='utf-8') as file:
 return re.findall(r'\bU+[0-9A-F]{4,5}\b', file.read())

def write_characters_to_file(characters, output_file_path):
 with open(output_file_path, 'w', encoding='utf-8') as file:
 for char in characters:
 file.write(chr(int(char[2:], 16)) + '\n')

def main():
 directory = '.' # Specify the directory where your files are located
 output_file_path = 'charSF2.txt'
 all_characters = set()
 files_to_process = ['Unihan_Readings', 'Unihan_DictionaryLikeData', 'Unihan_IRGSources'] # Add more
 file names as needed

 for filename in os.listdir(directory):
 for file_prefix in files_to_process:
 if filename.startswith(file_prefix) and filename.endswith('.txt'):
 file_path = os.path.join(directory, filename)
 characters = extract_characters_from_file(file_path)
 all_characters.update(characters)

 write_characters_to_file(all_characters, output_file_path)
 print("Characters extracted and saved to", output_file_path)

if __name__ == "__main__":
 main()

```

```

----- Content of FIdentifier.cpp-----
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>

```

```

int main(){
//char i[4] = {""};
int j;
int tmax;
puts("Note: Program Output--> FIdentified.txt--> In current Directory\n");
printf("\nStatement/Word Character Length = ");
scanf("%d", &tmax);
j = tmax;
printf("\n\t%d it is", j);
char ui[tmax];
printf("\n\nInput String (Max Length = 4): ");
int x;
puts("\n");
scanf("%s", &ui);
printf("Your Input:\n\t");
for(x = 0; x < j; x++){
 printf("%c", ui[x]);
}
/*
fn(v,p) = (v(a + i)(p(n- k)) + ... + v(0))- (p(n- l) + ... + p(0));
i = {1,2,3,4 ...};
k = {1,2,3,4 ...};
l = {1,2,3,4 ...};
a = 0;
p(n- k) = e^(n-k);
e = number of potential events in the set of potential events;
v(0) = NULL;
p(0) = NULL;
(v1)(NULL) = v1;
NULL + NULL = 0;
n = number of events of the statement or of the word;
Read from left to right;

```

God

→

```

//puts("How many characters long is your input: ");
//scanf("%d", &j);
int e,kmax,item,v,a1,i,p,n,k,l,pros,psum,image;
long N;
e = 100;
a1 = 0;
k = 1;
n = j;
kmax = n - k;
int pr,f;
f = 0;
N = 0;
if(n <= 4 && n > 0){
 i = a1;
 item = a1 + i;
 while(f < n){
 for(image = 0; image < e; image++){
 if(ui[item] == a[image]){
 N = N + ((image + 1)*(pow(e,n - k)));
 image = e;
 k++;
 item++;
 }
 }
 i++;
 }
 f++;
}

image = 0;
}

//N = N + v;
int iter;
for(iter = 0; iter < kmax; ++iter){
 N = N - (pow(e,kmax - iter));
}
FILE *pi;
pi = fopen("Fidentified.txt","w");

fprintf(pi,"%dCF%d\n\t",n,N);
fprintf(pi,"%s", ui);
printf("\n\nFile Identified:\n\t%dCF%d\n\n",n,N);

fclose(pi);

system("pause");

return 0;

```

```
}
```

----- Content of File\_Explorer.py-----

```
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog
import os

def open_directory():
 folder_selected = filedialog.askdirectory()
 if folder_selected:
 update_file_list(folder_selected)

def update_file_list(folder_selected):
 for widget in file_canvas_frame.winfo_children():
 widget.destroy()

 for file_name in os.listdir(folder_selected):
 btn = ttk.Button(file_canvas_frame, text=file_name, command=lambda name=file_name:
open_file_or_folder(folder_selected, name))
 btn.pack(side=tk.TOP, anchor=tk.W, padx=5, pady=5)

 file_canvas_frame.update_idletasks()
 file_canvas.config(scrollregion=file_canvas.bbox("all"))

def open_file_or_folder(folder, file_name):
 path = os.path.join(folder, file_name)
 if os.path.isdir(path):
 update_file_list(path)
 else:
 os.system(f'start "{path}"')

root = tk.Tk()
root.title("Simple File Explorer")

frame = tk.Frame(root, padx=10, pady=10)
frame.pack(padx=20, pady=20, fill=tk.BOTH, expand=True)

file_frame = tk.Frame(frame)
file_frame.pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)

file_canvas = tk.Canvas(file_frame)
file_canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

file_scrollbar = ttk.Scrollbar(file_frame, orient="vertical", command=file_canvas.yview)
file_scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

file_canvas.configure(yscrollcommand=file_scrollbar.set)
```

```
file_canvas_frame = tk.Frame(file_canvas)
file_canvas.create_window((0,0), window=file_canvas_frame, anchor="nw")

button_open_directory = ttk.Button(frame, text="Open Directory", command=open_directory)
button_open_directory.pack(side=tk.LEFT, padx=10, pady=10)

root.mainloop()
```

----- Content of find\_square\_number.py-----

```
def closest_square(n):
 """
 Returns the closest square number to the given integer n.
 """

 # Calculate the square root of the number
 root = n ** 0.5

 # Find the two closest integers to the square root
 lower_integer = int(root)
 upper_integer = lower_integer + 1

 # Compare the distance of their squares to the original number
 if abs(n - lower_integer**2) <= abs(n - upper_integer**2):
 return lower_integer**2
 else:
 return upper_integer**2

def main():
 # Prompt the user for an integer
 num = int(input("Enter an integer for the closest square number to it:"))

 # Find and print the closest square number
 print(f"The closest square number to {num} is {closest_square(num)}.")

if __name__ == "__main__":
 main()
```

----- Content of Finite-Automata.py-----

#Computational Mechanics of Finite Elements Using Finite Automata

```
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import matplotlib.animation as animation
import numpy as np
import random

class MealyMachine:
 def __init__(self, state_table, output_table, initial_state):
 self.state_table = state_table
```

```

 self.output_table = output_table
 self.current_state = initial_state

def transition(self, input_val):
 self.current_state = self.state_table[self.current_state][input_val]
 return self.output_table[self.current_state][input_val]

state_table = {
 'State1': {'input1': 'State2', 'input2': 'State3', 'input3': 'State4', 'input4': 'State1'},
 'State2': {'input1': 'State3', 'input2': 'State4', 'input3': 'State1', 'input4': 'State2'},
 'State3': {'input1': 'State4', 'input2': 'State1', 'input3': 'State2', 'input4': 'State3'},
 'State4': {'input1': 'State1', 'input2': 'State2', 'input3': 'State3', 'input4': 'State4'},
}

Pad the hex color with zeros to ensure it has six digits
hex_color1 = input("Enter Hex Font Color, e.g. (#000000): ")
hex_color2 = input("Enter Hex Square Color, e.g. (#0E0000): ")

output_table = {
 'State1': {'input1': (0, 1, hex_color1), 'input2': (0,-1, 'blue'), 'input3': (1, 0, 'green'), 'input4': (-1, 0, 'yellow')},
 'State2': {'input1': (0, 1, 'yellow'), 'input2': (0,-1, 'green'), 'input3': (1, 0, 'blue'), 'input4': (-1, 0, 'red')},
 'State3': {'input1': (0, 1, 'blue'), 'input2': (0,-1, 'yellow'), 'input3': (1, 0, 'red'), 'input4': (-1, 0, 'green')},
 'State4': {'input1': (0, 1, 'green'), 'input2': (0,-1, 'red'), 'input3': (1, 0, 'yellow'), 'input4': (-1, 0, hex_color2)},
}

Initialize plot
fig, ax = plt.subplots()

n = int(input("Enter GridWidth: ")) # Number of squares in one dimension
squares = []
machines = []
for i in range(n):
 for j in range(n):
 square = Rectangle((i*2, j*2), 1, 1, fill=None, edgecolor='r')
 ax.add_patch(square)
 squares.append(square)
 machines.append(MealyMachine(state_table, output_table, np.random.choice(['State1', 'State2', 'State3', 'State4'])))

ax.set_xlim(-2, 2*n)
ax.set_ylim(-2, 2*n)
ax.set_aspect('equal') # ensures that the square remains a square

in1 = input("Enter word 1: ")
in2 = input("Enter word 2: ")
in3 = input("Enter word 3: ")
in4 = input("Enter word 4: ")

```

```

inputs = np.random.choice(['input1', 'input2', 'input3', 'input4'], size=20) # Inputs for 20 iterations

def update(num, squares, machines):
 input_val = inputs[num]
 for square, machine in zip(squares, machines):
 dx, dy, color = machine.transition(input_val)

 # Check if the square is about to move outside of the frame
 new_x = square.get_x() + dx
 new_y = square.get_y() + dy
 if new_x < 0 or new_x > 2*n- 1 or new_y < 0 or new_y > 2*n- 1:
 continue

 square.set_x(new_x)
 square.set_y(new_y)
 square.set_edgecolor(color)

ani = animation.FuncAnimation(fig, update, frames=len(inputs), fargs=[squares, machines])

ani.save('animation.gif', writer='imagemagick', fps=1)

plt.show()

```

----- Content of Finite\_Automata.py-----

```

import re

def read_file(filename):
 with open(filename, 'r') as file:
 return [line.strip() for line in file]

class FSM:
 def __init__(self, transitions, start_state, accept_states):
 self.transitions = transitions
 self.current_state = start_state
 self.accept_states = accept_states

 def transition(self, input_conditions):
 for transition, next_state in self.transitions.get(self.current_state, {}).items():
 if eval(transition.format(**input_conditions)):
 self.current_state = next_state
 return

```

```

def is_accept_state(self):
 return self.current_state in self.accept_states

def evaluate_logic(logic_list, words_dict):
 binary_results = []
 for logic, truth_value, binary_id in logic_list:
 # Replace numbers in logic with corresponding words
 for num, word in words_dict.items():
 logic = logic.replace(str(num), f'"{word}"')
 # Evaluate the logic expression
 eval_result = eval(logic)
 # Compare the evaluation result to the desired truth value
 result = '1' if eval_result == (truth_value == '1') else '0'
 binary_results.append(result)
 return ''.join(binary_results)

Parse logic.txt to create a list of tuples containing logic expressions, truth values, and binary identifiers
logic_file = read_file('logic.txt')
logic_list = []
for line in logic_file:
 parts = line.rsplit(' ', 2)
 if len(parts) == 3:
 logic_list.append(tuple(parts))
 else:
 print(f"Error: Unable to parse line: {line}")

Parse words.txt to create a dictionary mapping binary digits to words
words_file = read_file('words.txt')
words_dict = {line.split()[0]: line.split()[1] for line in words_file}

Evaluate logic for each logic expression and compare against truth value
result_string = evaluate_logic(logic_list, words_dict)
print(result_string)

```

----- Content of Generate-PDF-LaTeX.py-----

```

import os
import subprocess

def is_valid_content_file(file_name):
 valid_extensions = ['.txt', '.tex', '.png']
 return os.path.splitext(file_name)[1] in valid_extensions

def sort_files(files):
 return sorted(files, key=lambda x: int(os.path.splitext(os.path.basename(x))[0]))

def process_text_file(file_path):
 with open(file_path, 'r') as file:
 content = file.read()

```

```

return "\\paragraph{} " + content

def process_latex_file(file_path):
 with open(file_path, 'r') as file:
 content = file.read()
 return content

def process_image_file(file_path):
 # Replace single backslashes with forward slashes
 formatted_path = file_path.replace('\\', '/')
 # No single quotes around the file path
 return
f"\\begin{{figure}}[h!]\n\\centering\n\\includegraphics[width=0.8\\textwidth]{{{formatted_path}}}\n\\end{{figure}}"

def process_file(file_path):
 extension = os.path.splitext(file_path)[1]
 if extension == '.txt':
 return process_text_file(file_path)
 elif extension == '.tex':
 return process_latex_file(file_path)
 elif extension == '.png':
 return process_image_file(file_path)
 else:
 return ""

def create_latex_document(directory):
 all_files = os.listdir(directory)
 content_files = [f for f in all_files if is_valid_content_file(f) and os.path.isfile(os.path.join(directory, f))]
 sorted_files = sort_files(content_files)
 document =
"\\documentclass[a5paper]{book}\n\\usepackage[utf8]{inputenc}\n\\usepackage{graphicx}\n\\begin{document}\n"
 for file in sorted_files:
 file_path = os.path.join(directory, file)
 document += process_file(file_path) + "\n\n"
 document += "\\end{document}"
 return document

def write_to_file(file_name, content):
 with open(file_name, 'w') as file:
 file.write(content)

def compile_pdf(latex_file):
 pdflatex_command = 'C:/texlive/2023/bin/windows/pdflatex.exe' # or the full path to pdflatex if it's not
 in your PATH
 try:
 subprocess.run([pdflatex_command, latex_file])
 except Exception as e:

```

```

print(f"An error occurred: {e}")

def main():
 directory = 'D:/PhD/TextbookMathematics' # Update with the path to your content directory
 docu = 'D:/PhD/TextbookMathematics'
 latex_document = create_latex_document(docu)
 latex_file_name = 'D:/PhD/TextbookMathematics/1000000.tex'
 write_to_file(latex_file_name, latex_document)
 compile_pdf(latex_file_name)

if __name__ == "__main__":
 main()

```

----- Content of generate\_setupCommandMatrix.py-----

```

Create the 'instructionSet.txt' file
with open('instructionSet.txt', 'w') as f:
 # Lines 1 to 13107: string_var# = input('Enter string_var: ')
 for i in range(13107):
 f.write(f"string_var{i} = input('Enter string_var: ')\n")

 # Lines 13108 to 26214: var_given# = input('Enter the variable name to be used: ')
 for i in range(13107, 26214):
 f.write(f"var_given{i} = input('Enter the variable name to be used: ')\n")

 # Lines 26215 to 39321: var_glob# = globals()[var_given#]
 for i in range(26214, 39321):
 f.write(f"var_glob{i} = globals().get(var_given{i}, 'Variable not found') # Error handling: 'Variable not
found'\n")

 # Lines 39322 to 52428: exec(var_glob#)
 for i in range(39321, 52428):
 f.write(f"if isinstance(var_glob{i}, str): exec(var_glob{i}) # Error handling: Execute only if it's a string\n")

 # Lines 52429 to 65535: eval(var_glob#)
 for i in range(52428, 65535):
 f.write(f"if isinstance(var_glob{i}, str): result = eval(var_glob{i}) # Error handling: Evaluate only if it's a
string\n")

 # Line 65536: Hyperlink
 f.write("webbrowser.open('https://example.com') # Opens hyperlink to this site\n")

Note: For logging, you can add a line to write the executed or evaluated command to a log file.
Note: For user authentication, you can add a line to check user credentials before executing or evaluating
a command.

```

----- Content of Generate\_valid\_fileLengths.py-----

```

import math

```

```

count = 1
e = 12
n = 4
s = 2
p = 0

def is_square(n):
 if(n//2 == 0 or n//3 == 0 or n//4 == 0 or math.sqrt(n).is_integer()):
 return True
 return False

#h = int(input("Enter the character length of your file: "))
while count <= 1000000:

 b = ((e // n == 1) or (e / n == 3) or (e // n == 2)) or (e // n != 0) is True

 c1 = int(e/4)
 c2 = (c1)
 c3 = e- c1
 c4 = int(c3/3)
 c5 = e- (c1 + c4)
 c6 = int(c5/2)
 c7 = e- (c1 + c4 + c6)
 c8 = int(c7/1)
 c9 = c8 + c6 + c4 + c1

 ht = is_square(c9)
 hg = is_square(c9***(1/2))
 nulle = e
 v1 = (((e/n) + 1)***(1/2))- ((e/n) + 1)***(1/2)
 #vx = ((v1 == 0) or (v1 == 1) or (v1 == 2) or (v1 == 3)) is True
 ex = c9***(1/2)*4
 if(b and (ht and hg)):
 print(f"The square {c9} of hash length {c9***(1/2)} corresponding to file length {ex- 0- 3}, {ex- 1- 3}, {ex- 2- 3} or {ex- 3- 3} including null ... is hashable")

 count += 1
 e += 1

```

----- Content of gfa.py-----

```

class FiniteAutomaton:
 def __init__(self, states, alphabet, transitions, start_state, accept_states):
 self.states = states
 self.alphabet = alphabet
 self.transitions = transitions
 self.start_state = start_state
 self.accept_states = accept_states

```

```

self.current_state = start_state

def define_transitions(self):
 for state in self.states:
 self.transitions[state] = {}
 for symbol in self.alphabet:
 while True:
 next_state = input(f"From state '{state}' on input '{symbol}', enter next state (e.g., q0, q1,...): ")
 if next_state in self.states:
 self.transitions[state][symbol] = next_state
 break
 else:
 print("Invalid state entered. Please try again, use format 'q<number>'.")

def process_input(self, input_string):
 # Adjusting for double-barrelled input strings
 if len(input_string) % 2 != 0 or len(input_string) > 24:
 return "Input string is not properly formatted or too long (maximum 12 symbols)"

 symbols = [input_string[i:i+2] for i in range(0, len(input_string), 2)] # Splitting into symbols

 for symbol in symbols:
 if symbol not in self.alphabet:
 return "Invalid input symbol"
 if symbol in self.transitions[self.current_state]:
 next_state = self.transitions[self.current_state][symbol]
 self.current_state = next_state
 else:
 return "Input string rejected due to missing transition"

 return "Input string accepted" if self.current_state in self.accept_states else "Input string rejected"

while True:
 yi = input("Continue? 1 (for yes) 2 (for no): ")
 if yi == '1':
 while True:
 try:
 u_states = int(input("Enter number of states (1 to 12): "))
 if not 1 <= u_states <= 12:
 raise ValueError
 break
 except ValueError:
 print("Invalid input. Please enter a number between 1 and 12.")

 while True:
 try:
 low_a = int(input("Enter lower range of the alphabet (0 to 11): "))
 high_a = int(input("Enter higher range of the alphabet (0 to 11): "))
 if not 0 <= low_a <= 11 or not 0 <= high_a <= 11 or low_a > high_a:
 raise ValueError
 except ValueError:

```

```

 break
 except ValueError:
 print("Invalid range. Please enter numbers between 0 and 11 with lower <= higher.")

Alphabet is defined to include symbols as two-digit strings
u_alphabet = [f"{i:02}" for i in range(low_a, high_a + 1)]
states = [f"q{i}" for i in range(u_states)]

my_fa = FiniteAutomaton(states, u_alphabet, {}, "q0", [])
my_fa.define_transitions()

while True:
 start_state = input("Choose a start state from {states}: ")
 if start_state in states:
 break
 else:
 print("Invalid start state. Please choose from the listed states.")

while True:
 accept_states_input = input("Enter accept states from {states} (comma-separated): ")
 accept_states = accept_states_input.split(",")
 if all(state in states for state in accept_states):
 break
 else:
 print("Invalid accept states. Please ensure all states are from the listed states.")

my_fa.start_state = start_state
my_fa.current_state = start_state
my_fa.accept_states = accept_states

input_string = input("Enter an input string: ")
result = my_fa.process_input(input_string)
print(result)

elif yi == '2':
 break
else:
 print("Invalid input, please enter 1 (yes) or 2 (no)")

```

----- Content of GOSm7.py -----

```

import tkinter as tk
import random
from PIL import ImageGrab
import pickle
from tkinter import simpledialog
import os
from tkinter import Menu, Checkbutton, BooleanVar
from tkinter.colorchooser import askcolor
import logging

```

```

import time
import json
import os
import pyautogui
import keyboard

current_col = 0
current_row = 0

def get_random_color():
 r = lambda: random.randint(0,255)
 return "#%02X%02X%02X" % (r(),r(),r())

def get_random_char():
 random_int = random.randint(0x0021, 0x007E)
 return chr(random_int)

def draw_grid():
 for i in range(grid_size):
 for j in range(grid_size):
 color = get_random_color()
 square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size, (i+1)*square_size,
 fill=color)
 char = get_random_char()
 text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2, text=char,
 font=("Arial", 18), anchor="center")

def draw_grid_IDE():
 global col
 try:
 for i in range(grid_size):
 for j in range(grid_size):
 color = "blue"
 square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size,
 (i+1)*square_size, fill=color, outline = "white")
 if typing_mode:
 # In the function where you create the canvas...
 canvas.bind("<Key>", on_key_press)
 canvas.focus_set()
 except:
 print("Submit an app mode ...")

def toggle_color_mode():
 """Toggles between manual color mode and random color mode"""
 global color_mode_var
 color_mode_var.set(not color_mode_var.get())

```

```

font_size = 9

def draw_char1(canvas, char, row, col, font_size, color):
 global cell_size, font_color, square_size, grid_size, last_drawn, note, char_note
 """
 def draw_char1(canvas, char, row, col, font_size):
 Draws a character on the Tkinter canvas at the specified row and column with the specified font size.

 Parameters:
 canvas (tk.Canvas): The Tkinter canvas to draw on.
 char (str): The character to draw.
 row (int): The row to draw the character at.
 col (int): The column to draw the character at.
 font_size (int): The font size to use for the character.

 """

try:
 x = col * square_size + square_size / 2
 y = row * square_size + square_size / 2

 print(f"Drawing char '{char}' at ({x}, {y})") # Debug print to check coordinates
 canvas.create_rectangle(col * square_size, row * square_size, (col+1) * square_size, (row+1) * square_size, fill=color)
 canvas.create_text(x, y, text=char, fill=font_color, font=('Calibri', int(font_size)))
 note = char_note.get()
 # Log the color and character info
 logging.info(f"Position:{x};{y};Xp:{x};Yp:{y};Character:{char};Square color:{color};Font color:{font_color};Note:{note}")
 # Log the color and character info in a text file
 if(note == ""):
 note = "emptyNote"
 with open("color_log.txt", "a", encoding='utf-8') as login:
 print(f"{x},{y},{char},{color},{font_color},{note}", file=login)

except Exception as e:
 print(f"Something Went Wrong... Error: {e}")

def draw_char(i=None, j=None):
 try:
 """Draws a character from input field in a specific square"""
 global char_count, IDE_mode, typing_mode, last_drawn, font_color
 if(mode == 'IDE'):
 IDE_mode = True
 # Check if last_drawn is defined, otherwise define it

```

```

if 'last_drawn' not in globals():
 last_drawn = []

If i, j are not provided, calculate them based on char_count
if i is None or j is None:
 i, j = divmod(char_count, grid_size)
 # Handle out-of-grid situations
 if i >= grid_size or j >= grid_size:
 print('Out of grid!')
 return

Generate a random color if not in typing mode, white otherwise
##color = get_random_color() if not typing_mode else "white"

Check color mode
if color_mode_var.get():
 # Manual color mode
 # Show a color picker and get the chosen color for the square
 color = askcolor(title="Choose square color")[1]

 # Ask for the font color
 color_result = askcolor(title="Choose font color")
 if color_result is not None:
 font_color = color_result[1]
 else:
 # Handle the case when the user cancelled the color selection
 font_color = "#000000" # default to black, for example

else:
 # Random color mode
 color = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
 color1 = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
 font_color = color1 ##"#000000"

square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size, (i+1)*square_size,
fill=color)
char = char_entry.get()[1]
note = char_note.get()
text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2, text=char,
font=("Arial", font_size), fill=font_color, anchor="center")

"""

Log the color and character info
logging.info(f"Position:{i};{j},Xp:{i},Yp:{j},Character:{char},Square color:{color},Font
color:{font_color}")
Log the color and character info in a text file
with open("color_log.txt", "a") as login:
 print(f"{i},{j},{char},{color},{font_color}", file=login)

```

```

"""

Log the color and character info
logging.info(f"Position:{i};{j},Xp:{i},Yp:{j},Character:{char},Square color:{color},Font
color:{font_color},Note:{note}")
Log the color and character info in a text file
if(note == ""):
 note = "emptyNote"
with open("color_log.txt", "a", encoding='utf-8') as login:
 print(f"{i},{j},{char},{color},{font_color},{note}", file=login)

if len(char_entry.get()) > 0: # Check if there's more than one character
 char_entry.delete(0, 1) # Delete the first character

last_drawn.append((square, text))

if not IDE_mode:
 char_count = (char_count + 1) % (grid_size * grid_size)
 if char_count == 0: # If we've filled the canvas, clear it
 canvas.delete('all')

 return square, text
except:
 print("Submit an app mode ...")

def adjust_grid_and_font():
 global grid_size, square_size, font_size, canvas_width, current_row, current_col

 n_g_s = simpledialog.askstring("Change Grid Size (Grid Width)", "Enter New Grid Size (+Integer):")
 if(n_g_s != ''):
 new_grid_size = n_g_s
 n_f_s = simpledialog.askstring("Change Font Size (Default = 9)", "Enter New Font Size (+Integer):")
 if(n_f_s != ''):
 new_font_size = n_f_s
 # Update the global variables
 if(new_grid_size==None and new_font_size==None):
 grid_size = 10
 font_size = 9
 if((not new_grid_size==None) and (not new_font_size==None)):
 try:
 grid_size = int(new_grid_size)
 square_size = canvas_width / grid_size
 font_size = int(new_font_size)
 except:
 print("positive integers please")

"""

Get new values from input fields

```

```

new_grid_size = grid_size_entry.get()
new_font_size = font_size_entry.get()

Update the global variables
grid_size = int(new_grid_size)
square_size = canvas_width // grid_size
font_size = int(new_font_size)
"""

Redraw the grid
refresh_canvas()
qit = input("Are you in typing mode, or want to enter typing mode? 1 for (No), 2 for (yes): ")
if(qit == '2'):
 current_row = 0
 current_col = 0
 show_typing_mode_menu()

def refresh_canvas():
 """Clears the canvas and resets the char_count"""
 global char_count
 canvas.delete('all')
 char_count = 0
 draw_grid_IDE()
 last_drawn.clear()

def undo_last():
 """Undoes the last drawing operation"""
 if last_drawn:
 square, text = last_drawn.pop()
 canvas.delete(square)
 canvas.delete(text)

def update_canvas():
 if not paused.get():
 canvas.delete('all')
 draw_grid()
 root.after(8000, update_canvas)

def toggle_typing_mode():
 global typing_mode, canvas

 typing_mode = not typing_mode
 if typing_mode:
 canvas.focus_set() # Set focus to the canvas for keyboard input

def on_canvas_click(event):
 try:
 global mode, typing_mode

```

```

if mode == 'IDE' and not typing_mode:
 j = event.x // square_size
 i = event.y // square_size
 draw_char(i, j)
else:
 paused.set(not paused.get())
except:
 print("Submit an app mode ...")

def on_key_press(event):
 global char_count, typing_mode
 if typing_mode:
 char_entry.delete(0, 'end') # Clear the entry box
 try:
 utf8_char = event.char.encode('utf-8').decode('utf-8')
 char_entry.insert(0, utf8_char) # Insert the typed character
 except UnicodeDecodeError:
 print("Non UTF-8 character detected")
 return
 draw_char() # Draw the character
 char_count += 1 # Increment the count

 # If char_count exceeds the total number of squares in the grid, reset it
 if char_count >= grid_size ** 2:
 char_count = 0

def submit_mode():

 global char_count, mode, typing_mode

 mode = mode_entry.get().upper()

 ...
 if mode in ['editor']:

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 draw_grid()
 char_label.pack()
 char_entry.pack()
 char_button.pack()

 ...

```

```
Draw the initial grid
if(mode != 'IDE'):
 print("A valid mode ...")

"""
if(mode == 'normal'):

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 draw_grid()
 create_menu_1()

if mode == 'normal':
 # Schedule the first update
 root.after(8000, update_canvas)
"""

if mode == 'IDE':

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 char_count = 0
 draw_grid_IDE()

 control_frame.pack()
 char_label.pack()
 char_entry.pack()
 #grid_size_label.pack()
 #grid_size_entry.pack()
 #font_size_label.pack()
 #font_size_entry.pack()
 #adjust_button.pack()
 char_note_label.pack()
 char_note.pack()
 #refresh_button.pack(side="left")
 #undo_button.pack(side="left")
 #save_button = tk.Button(root, text='Save', command=save_canvas, bg="white", padx=5, pady=0)
 #save_button.pack(side="left")
 #save_Sbutton.pack(side="left")
 #load_button.pack(side="left")
 create_menu()
```

```
return mode
```

```
def save_canvas():
 global IC_value, xSwitch, ing
 xSwitch = 1
 ing = 1
 if(ing == 0):
 ing = 2
 ing = 1
 if(xSwitch == 1 and ing == 1):
 IC_value = 0
 ing = 0
 """Save the current state of the canvas to a .png file"""
 #filename = simpledialog.askstring("Save Canvas", "Enter filename:")
 filename = f"ImageCanvas{IC_value}_rename"

if filename: # Only save the canvas if a filename was entered
 # Get the bounding box of the canvas
 x = root.winfo_rootx() + canvas.winfo_x()
 y = root.winfo_rooty() + canvas.winfo_y()
 x1 = x + canvas.winfo_width()
 y1 = y + canvas.winfo_height()
 time.sleep(3)
 # Grab the image, crop it to the bounding box, and save it
 ImageGrab.grab().crop((x, y, x1, y1)).save(filename + ".png")

def save_session():
 """Save the current session to a file using pickle"""
 global last_drawn
 # Ask the user to enter a filename
 filename = simpledialog.askstring("Save Session", "Enter filename:")
 if filename: # Only save the session if a filename was entered
 session_data = [(canvas.coords(square), canvas.itemcget(square, "fill"),
 canvas.coords(text), canvas.itemcget(text, "text"), canvas.itemcget(text, "fill"))
 for square, text in last_drawn]
 with open(filename + ".pkl", "wb") as f:
 pickle.dump(session_data, f)

def load_session():
 """Load a saved session from a file using pickle"""
 global last_drawn
 # Ask the user to enter a filename
 filename = simpledialog.askstring("Load Session", "Enter filename:")
 if filename: # Only try to load a session if a filename was entered
```

```

try:
 with open(filename + ".pkl", "rb") as f:
 session_data = pickle.load(f)
 # Clear the canvas and redraw all elements from the loaded session
 canvas.delete('all')
 draw_grid_IDE()
 last_drawn = []
 for square_coords, square_fill, text_coords, text_content, text_fill in session_data:
 square = canvas.create_rectangle(*square_coords, fill=square_fill)
 text = canvas.create_text(*text_coords, text=text_content, fill=text_fill, font=("Arial", font_size),
 anchor="center")
 last_drawn.append((square, text))
 except FileNotFoundError:
 print(f"No session named '{filename}' found.")

#def exit_app():
exit()

def show_about():
 about_window = tk.Toplevel(root)
 about_window.title("About")
 about_msg = "This is a program created to learn and experiment with Tkinter. In IDE mode, you can draw characters on a grid (By typing or copying and pasting then clicking the squares you want to draw each character on), adjust the grid and font size, save and load sessions, and more, like saving the grid as an image (For example). C(num) was written with the aid of Chat GPT. Enjoy!\n\nNote: the workflow ... IDE to color_log.txt (Compiled by compile#.exe or compile#.py);\\nindex_1.txt contains fields of study or research and development and\\nindex_2.txt contains fields of study or research and development also ..\n\nBertotools Digital"
 tk.Message(about_window, text=about_msg, width=500).pack()
 tk.Button(about_window, text="OK", command=about_window.destroy).pack()

def compileGOSmX():
 MAX_LINE_LENGTH = 256
 MAX_TOPICS = 256

global index_1, index_2

index_1 = [
 "Axiom",
 "Theorem",
 "Lemma",
 "Proposition",
 "Corollary",
 "Conjecture",
 "Proof",
 "Premise",
 "Conclusion",
 "Hypothesis",
 "Counterexample",
 "Direct Proof",
]

```

"Indirect Proof",  
"Proof by Contradiction (Reductio ad absurdum)",  
"Proof by Induction",  
"Proof by Contrapositive",  
"Deductive Reasoning",  
"Inference",  
"Assumption",  
"Statement",  
"Postulate",  
"Proof by Exhaustion",  
"Syllogism",  
"Constructive Proof",  
"Non-Constructive Proof",  
"Trivial Proof",  
"Vacuous Proof",  
"Biconditional",  
"Condition",  
"Sufficiency",  
"Necessity",  
"Quantifier",  
"Universal Quantifier",  
"Existential Quantifier",  
"Bound Variable",  
"Free Variable",  
"Predicate",  
"Propositional Logic",  
"Modus Ponens",  
"Modus Tollens",  
"Discrete Mathematics",  
"Set Theory",  
"Function",  
"Bijection",  
"Injection",  
"Surjection",  
"Equivalence Relation",  
"Partial Order",  
"Total Order",  
"Well-Order",  
"Reflexivity",  
"Symmetry",  
"Transitivity",  
"Antisymmetry",  
"Completeness",  
"Compactness",  
"Connectedness",  
"Convergence",  
"Divergence",  
"Limit",  
"Sequence",  
"Series",

"Monotonicity",  
"Cauchy Sequence",  
"Infinite Set",  
"Finite Set",  
"Cardinality",  
"Countable Set",  
"Uncountable Set",  
"Subset",  
"Superset",  
"Intersection",  
"Union",  
"Empty Set",  
"Power Set",  
"Cartesian Product",  
"Equivalence Class",  
"Partition",  
"Field",  
"Ring",  
"Group",  
"Abelian Group",  
"Non-abelian Group",  
"Matrix",  
"Vector Space",  
"Linear Transformation",  
"Eigenvalue",  
"Eigenvector",  
"Norm",  
"Inner Product",  
"Orthogonality",  
"Basis",  
"Dimension",  
"Rank",  
"Nullity",  
"Determinant",  
"Graph Theory",  
"Vertex",  
"Edge",  
"Connectivity",  
"Cycle",  
"Path",  
"Degree",  
"Subgraph",  
"Tree",  
"Forest",  
"Planar Graph",  
"Bipartite Graph",  
"Directed Graph (Digraph)",  
"Eulerian Graph",  
"Hamiltonian Graph",  
"Adjacency Matrix",

"Incidence Matrix",  
"Isomorphism",  
"Homeomorphism",  
"Topology",  
"Open Set",  
"Closed Set",  
"Boundary",  
"Compact Space",  
"Hausdorff Space",  
"Continuity",  
"Differential",  
"Derivative",  
"Integral",  
"Partial Derivative",  
"Multivariable Calculus",  
"Laplace Transform",  
"Fourier Transform",  
"Taylor Series",  
"Maclaurin Series",  
"Conic Sections",  
"Hyperbola",  
"Ellipse",  
"Parabola",  
"Asymptote",  
"Limits at Infinity",  
"Complex Number",  
"Imaginary Unit",  
"Real Number",  
"Rational Number",  
"Irrational Number",  
"Prime Number",  
"Composite Number",  
"GCD (Greatest Common Divisor)",  
"LCM (Least Common Multiple)",  
"Permutation",  
"Combination",  
"Probability",  
"Statistics",  
"Expected Value",  
"Variance",  
"Standard Deviation",  
"Normal Distribution",  
"Poisson Distribution",  
"Binomial Distribution",  
"Hypothesis Testing",  
"Regression",  
"Correlation",  
"Matrix Algebra",  
"Linear Algebra",  
"Vector Calculus",

"Optimization",  
"Algorithm",  
"Computational Complexity",  
"Big O Notation",  
"Pigeonhole Principle",  
"Principle of Inclusion-Exclusion",  
"Turing Machine",  
"Computability",  
"Unsolvability",  
"Parity",  
"Diophantine Equations",  
"Cryptography",  
"Fermat's Last Theorem",  
"Pythagorean Theorem",  
"Triangle Inequality",  
"Trigonometric Functions",  
"Trigonometric Identities",  
"Polar Coordinates",  
"Euler's Formula",  
"Riemann Zeta Function",  
"P vs NP Problem",  
"NP-complete Problem",  
"Stochastic Process",  
"Markov Chain",  
"Random Variable",  
"Conditional Probability",  
"Bayes' Theorem",  
"Monte Carlo Method",  
"Fractal",  
"Chaos Theory",  
"Game Theory",  
"Nash Equilibrium",  
"Zero-Sum Game",  
"Non-Zero-Sum Game",  
"Linear Programming",  
"Nonlinear Programming",  
"Quadratic Programming",  
"Dynamic Programming",  
"Integer Programming",  
"Graph Coloring",  
"Network Flow",  
"Spanning Tree",  
"Bellman-Ford Algorithm",  
"Dijkstra's Algorithm",  
"Kruskal's Algorithm",  
"Prim's Algorithm",  
"Floyd-Warshall Algorithm",  
"Euler's Method",  
"Runge-Kutta Method",  
"Numerical Integration",

```
"Numerical Differentiation",
"Bisection Method",
"Newton's Method",
"Secant Method",
"Fixed Point Iteration",
"Linear Interpolation",
"Polynomial Interpolation",
"Lagrange Interpolation",
"Splines",
"Fourier Series",
"Laplace's Equation",
"Heat Equation",
"Wave Equation",
"Schrodinger Equation",
"Ordinary Differential Equation (ODE)",
"Partial Differential Equation (PDE)",
"Boundary Value Problem",
"Initial Value Problem",
"Green's Theorem",
"Stoke's Theorem",
"Divergence Theorem",
"Curl",
"Gradient",
"Divergence",
"Tensor",
"Manifold",
"Topological Space",
"Measure Theory",
"Lebesgue Integral",
"Borel Set",
"Hilbert Space",
"Banach Space",
"Category Theory",
"Functor",
"Natural Transformation",
"Sheaf",
"Homotopy",
"Homology",
"Cohomology",
"Galois Theory",
"Algebraic Geometry",
"Topological K-Theory",
"Knot Theory",
"Lattice Theory"
]

index_2 = [
 "Biochemistry",
 "Biophysics",
 "Molecular biology",
```

"Genetics",  
"Immunology",  
"Cell biology",  
"Microbiology",  
"Neuroscience",  
"Pharmacology",  
"Bioinformatics",  
"Biotechnology",  
"Proteomics",  
"Genomics",  
"Structural biology",  
"Virology",  
"Systems biology",  
"Developmental biology",  
"Evolutionary biology",  
"Synthetic biology",  
"Metabolomics",  
"Epigenetics",  
"Tissue engineering",  
"Nanotechnology",  
"Materials science",  
"Quantum physics",  
"Condensed matter physics",  
"Particle physics",  
"Astrophysics",  
"Cosmology",  
"Optics",  
"Atomic and molecular physics",  
"Fluid mechanics",  
"Thermodynamics",  
"Environmental science",  
"Climate science",  
"Geology",  
"Oceanography",  
"Atmospheric science",  
"Ecology",  
"Conservation biology",  
"Botany",  
"Zoology",  
"Entomology",  
"Marine biology",  
"Paleontology",  
"Anthropology",  
"Archaeology",  
"Psychology",  
"Cognitive science",  
"Social psychology",  
"Linguistics",  
"Artificial intelligence",  
"Machine learning",

"Computer vision",  
"Natural language processing",  
"Human-computer interaction",  
"Robotics",  
"Computer graphics",  
"Data science",  
"Mathematical modeling",  
"Mathematical physics",  
"Number theory",  
"Algebraic geometry",  
"Differential equations",  
"Computational physics",  
"Mathematical biology",  
"Operations research",  
"Biostatistics",  
"Epidemiology",  
"Cancer research",  
"Diabetes research",  
"Heart disease research",  
"Infectious diseases research",  
"Immunotherapy",  
"Stem cell research",  
"Gene therapy",  
"Drug discovery",  
"Precision medicine",  
"Health informatics",  
"Renewable energy",  
"Energy storage",  
"Sustainable materials",  
"Environmental engineering",  
"Water management",  
"Transportation engineering",  
"Civil engineering",  
"Chemical engineering",  
"Aerospace engineering",  
"Biomedical engineering",  
"Electrical engineering",  
"Mechanical engineering",  
"Robotics engineering",  
"Quantum computing",  
"Cryptography",  
"Cybersecurity",  
"Network engineering",  
"Telecommunications",  
"Human genetics",  
"Forensic science",  
"Space exploration and research",  
"Planetary science",  
"Astrobiology",  
"Astrochemistry",

"Astrogeology",  
"Astroinformatics",  
"Exoplanet research",  
"Stellar evolution",  
"Galactic astronomy",  
"Observational astronomy",  
"Computational astrophysics",  
"Quantum chemistry",  
"Computational chemistry",  
"Organic chemistry",  
"Inorganic chemistry",  
"Physical chemistry",  
"Environmental chemistry",  
"Analytical chemistry",  
"Agricultural science",  
"Food science",  
"Nutritional science",  
"Exercise physiology",  
"Sports science",  
"Biomechanics",  
"Plant physiology",  
"Plant genetics",  
"Plant pathology",  
"Soil science",  
"Hydrology",  
"Geochemistry",  
"Geophysics",  
"Geomorphology",  
"Remote sensing",  
"Geotechnical engineering",  
"Petroleum engineering",  
"Aerospace materials",  
"Nanomaterials",  
"Polymer science",  
"Computational materials science",  
"Photonics",  
"Physical optics",  
"Quantum optics",  
"Neuroengineering",  
"Brain imaging",  
"Cognitive neuroscience",  
"Neuroinformatics",  
"Psychophysics",  
"Developmental psychology",  
"Personality psychology",  
"Clinical psychology",  
"Industrial-organizational psychology",  
"Educational psychology",  
"Psycholinguistics",  
"Human genetics",

"Evolutionary genetics",  
"Population genetics",  
"Genetic engineering",  
"Genetic counseling",  
"Epigenomics",  
"Cardiovascular research",  
"Respiratory research",  
"Gastroenterology research",  
"Endocrinology research",  
"Nephrology research",  
"Hematology research",  
"Ophthalmology research",  
"Orthopedic research",  
"Dermatology research",  
"Veterinary medicine",  
"Animal behavior",  
"Conservation ecology",  
"Wildlife biology",  
"Environmental microbiology",  
"Agricultural economics",  
"Development economics",  
"Behavioral economics",  
"Econometrics",  
"Financial mathematics",  
"Operations management",  
"Supply chain management",  
"Industrial engineering",  
"Human-computer interaction",  
"Virtual reality",  
"Augmented reality",  
"Data mining",  
"Text mining",  
"Big data analytics",  
"Computational linguistics",  
"Quantum information science",  
"Quantum cryptography",  
"Biometrics",  
"Information retrieval",  
"Software engineering",  
"Computer networks",  
"Embedded systems",  
"Human-robot interaction",  
"Control systems",  
"Biopharmaceuticals",  
"Drug delivery systems",  
"Clinical trials",  
"Regenerative medicine",  
"Agricultural biotechnology",  
"Plant breeding",  
"Animal breeding",

"Food technology",  
"Sensory science",  
"Poultry science",  
"Aquaculture",  
"Marine ecology",  
"Limnology",  
"Population ecology",  
"Landscape ecology",  
"Evolutionary ecology",  
"Environmental toxicology",  
"Environmental chemistry",  
"Environmental microbiology",  
"Ecotoxicology",  
"Green chemistry",  
"Space physics",  
"Space weather",  
"Astrostatistics",  
"Computational fluid dynamics",  
"Mathematical optimization",  
"Operations research",  
"Human genetics",  
"Functional genomics",  
"Molecular genetics",  
"Cancer genetics",  
"Psychiatric genetics",  
"Population genomics",  
"Bioengineering",  
"Biomaterials",  
"Biomechatronics",  
"Cardiovascular engineering",  
"Neural engineering",  
"Rehabilitation engineering",  
"Genetic engineering",  
"Environmental engineering",  
"Water resources engineering",  
"Structural engineering",  
"Robotics engineering",  
"Quantum information theory",  
"Quantum simulation",  
"Quantum sensing",  
"Geographical information systems (GIS)",  
"Urban planning",  
"Renewable energy systems",  
"Solar cell technology",  
"Wind energy research",  
"Energy policy and economics",  
"Computational neuroscience",  
"Neurobiology",  
"Cognitive neuroscience",  
"Systems neuroscience",

```

"Human-robot interaction",
"Evolutionary psychology",
"Social network analysis"
]

def square_print_topic_indices(r_value, g_value, b_value, file):
 index1 = r_value % len(index_1)
 index2 = g_value % len(index_1)
 index3 = b_value % len(index_1)
 ...
 file.write(f"\t\t\t{topics[index1]}\n")
 file.write(f"\t\t\t{topics[index2]}\n")
 file.write(f"\t\t\t{topics[index3]}\n")
 ...
 file.write(f"\t\t\t{index_1[index1]}\n")
 file.write(f"\t\t\t{index_1[index2]}\n")
 file.write(f"\t\t\t{index_1[index3]}\n")

 squareRatio = str(index1) + ":" + str(index2) + ":" + str(index3)
 return squareRatio

def font_print_topic_indices(r_value, g_value, b_value, file):
 index1_ = r_value % len(index_2)
 index2_ = g_value % len(index_2)
 index3_ = b_value % len(index_2)
 file.write(f"\t\t\t{index_2[index1_]}\n")
 file.write(f"\t\t\t{index_2[index2_]}\n")
 file.write(f"\t\t\t{index_2[index3_]}\n")

 fontRatio = str(index1_) + ":" + str(index2_) + ":" + str(index3_)
 return fontRatio

def get_index(i, j, grid_size):
 return i * grid_size + j

def extract_rgb_values(color):
 if len(color) != 7 or color[0] != '#':
 raise ValueError("Input should be a hex color code in the format '#RRGGBB'")
 try:
 red = int(color[1:3], 16)
 green = int(color[3:5], 16)
 blue = int(color[5:7], 16)
 return red, green, blue
 except ValueError:
 raise ValueError("Invalid color code. RGB values should be hex digits (0-9, A-F)'")

```

```

def GO():
 log_file = "color_log.txt"
 ## topic_file = "index_1.txt"
 ## topic2_file = "index_2.txt" # Corrected this line to read from a different file
 gridWidth = int(input("Enter gridWidth: "))
 # Read topic files to get the topics and topic2s
 ## with open(topic_file, "r", encoding='utf8') as file:
 ## topics = [line.strip() for line in file]
 ##
 ## with open(topic2_file, "r", encoding='utf8') as file:
 ## topic2s = [line.strip() for line in file]

 numTopics = len(index_1)
 numTopic2s = len(index_2)

 # Read color log file
 with open(log_file, "r") as file:
 lines = file.readlines()

 lineNumber = 0
 # Save output to a file
 output_file = "output.txt"
 with open(output_file, "w") as file:
 print()
 for line in lines:
 parts = line.strip().split(",")
 if len(parts) >= 5:
 xcoor = int(parts[1])
 ycoor = int(parts[0])
 index_ = (gridWidth * ycoor) + xcoor
 character = parts[2]
 squareColor = parts[3]
 fontColor = parts[4]
 notes = parts[5]

 # Extract RGB values from squareColor
 red, green, blue = extract_rgb_values(squareColor)

 # Extract RGB values from fontColor
 fontRed, fontGreen, fontBlue = extract_rgb_values(fontColor)

 # Calculate the grid size
 grid_size = int(parts[0])

 # Select topics based on RGB values from squareColor
 topicIndices = [(red + get_index(i, j, grid_size)) % numTopics for i in range(grid_size) for j in
range(grid_size)]

 lineNumber += 1

```

```

with open(output_file, "a") as file:
 file.write(f"Compiling Line/s: {lineNumber}\n\n")
 file.write(f"Line({lineNumber})\n\n")
 file.write(f"\tLineNumber: {lineNumber}\tx-coordinate: {xcoor}\ty-coordinate: {ycoor}\t
Character: {character}\tRGB: {squareColor}\tRGB: {fontColor}\t Note: {notes}\t GridDimensions:
{gridWidth}x{gridWidth}\n\n")
 #print(topicIndices)
 file.write(f"\tLine/s({lineNumber}) Output:\n\n")
 file.write(f"\t#define Line({lineNumber}){\n\n")
 # Inside the main function
 file.write(f"\t\t_i_1_0(squareColor){\n\n")
 file.write(f"\t\t\t{character}\n")
 file.write(f"\t\t\t(R: {red}, G: {green}, B: {blue})\n")
 s_ratio = square_print_topic_indices(red, green, blue, file)
 #square_print_topic_indices(red, green, blue, topics, file) # Modify this line
 file.write("\t\t)\n")
 file.write("\t\t;\n\n")
 file.write(f"\t\t_i_1_1(fontColor){\n\n")
 file.write(f"\t\t\t{character}\n")
 file.write(f"\t\t\t(R: {fontRed}, G: {fontGreen}, B: {fontBlue})\n")
 f_ratio = font_print_topic_indices(fontRed, fontGreen, fontBlue, file)
 #font_print_topic_indices(fontRed, fontGreen, fontBlue, topic2s, file) # And this line
 file.write("\t\t)\n")
 file.write(f"\treturn squareRatio({s_ratio}), fontRatio({f_ratio}), sqaureIndex({index_})\n")
 #file.write(f"\treturn fontRatio({f_ratio})\n")
 file.write("}\n\n")

```

GO()

```

def create_menu():
 global control_frame
 def home():
 canvas.pack_forget()
 menubar.destroy()
 control_frame.pack()
 char_entry.pack_forget()
 char_label.pack_forget()
 #grid_size_label.pack_forget()
 #grid_size_entry.pack_forget()
 #font_size_label.pack_forget()
 #font_size_entry.pack_forget()
 #adjust_button.pack_forget()
 char_note.pack_forget()
 char_note_label.pack_forget()

root.destroy()

```

```

switch = 1
setup_mode(switch)

Create a menubar
menubar = tk.Menu(root)

Create an options menu and add it to the menubar
options_menu = tk.Menu(menubar, tearoff=0)
#menubar.add_cascade(label="Options", menu=options_menu)

Create a dropdown menu and add it to the menubar
dropdown = tk.Menu(menubar, tearoff=0)
menubar.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"

Add commands to dropdown menu

Add checkbutton to options menu
dropdown.add_command(label="Toggle Manual/Random Colour", command=manual_colour)

dropdown.add_command(label="Refresh", command=refresh_canvas)
dropdown.add_command(label="Undo", command=undo_last)
dropdown.add_command(label="Save Session", command=save_session)
dropdown.add_command(label="Load Session", command=load_session)
dropdown.add_command(label="Save as Image", command=save_canvas)
dropdown.add_command(label="Compile Canvas", command=compileGOSmX)
#dropdown.add_command(label="Exit", command=exit_app)
#dropdown.add_command(label="Toggle Type/Click Mode", command=toggle_typing_mode)
dropdown.add_command(label="About Program", command=show_about)
dropdown.add_command(label="Go Home", command=home)
dropdown.add_command(label="Edit Grid Dimensions", command=adjust_grid_and_font)
dropdown.add_command(label="Typing Mode", command=show_typing_mode_menu)

Set the menubar
root.config(menu=menubar)

```

```

def create_menu_1():
 global control_frame
 def home1():
 canvas.pack_forget()
 menubar1.destroy()
 root.destroy()
 switch = 1
 setup_mode(switch)

Create a menubar
menubar1 = tk.Menu(root)

```

```

Create an options menu and add it to the menubar
options_menu1 = tk.Menu(menuBar1, tearoff=0)
menuBar1.add_cascade(label="Options", menu=options_menu)

Create a dropdown menu and add it to the menubar
dropdown = tk.Menu(menuBar1, tearoff=0)
menuBar1.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"

Add commands to dropdown menu

Add checkbutton to options menu
dropdown.add_command(label="About Program", command=show_about)
dropdown.add_command(label="Change Mode", command=home1)

Set the menubar
root.config(menu=menuBar1)

def manual_colour():
 # Toggle the value of color_mode_var
 current_value = color_mode_var.get()
 color_mode_var.set(not current_value)

"""

Create a tkinter window
root = tk.Tk()
root.title("General Operating System")
root.iconbitmap("logo.ico")
"""

def show_typing_mode_menu():
 global color_
 while True:
 print("1. Use default UTF-8 mapping")
 print("4. Exit")
 choice = input("Choose an option: ")

 if choice == "1":
 # Use default UTF-8 mapping
 print("Using default UTF-8 mapping.")
 color_ = input("Use Default Colour Scheme (1): ")
 if(color_ == '1'):
 # Initialize bg_color and font_color with default values at the global scope
 bg_color = "#FFFFFF" # Default white background color
 font_color = "#000000" # Default black font color
 break
 else:
 print("Invalid choice. Please try again.")
```

```
canvas.bind("<Key>", type_character)
canvas.focus_set()

def set_background_color():
 global color
 #bg_color = tk.colorchooser.askcolor()[1]
 color = input("Enter background color (#FFFFFF for example):")

def set_font_color():
 global font_color, font_size
 #font_color = tk.colorchooser.askcolor()[1]
 font_color = input("Enter background color (#000000 for example): ")
 font_size = input("Enter font_size: ")

def type_character(event):
 global current_row, current_col, col, font_size, font_color, color, canvas_width, square_size
 color = "#FFFFFF"
 font_color = "#000000"
 char = event.char
 if char: # Ignore non-character events
 try:
 draw_char1(canvas, char, current_row, current_col, font_size, color)
 current_col += 1
 if (current_col >= canvas_width/square_size):
 current_col = 0
 current_row += 1
 if(current_row >= canvas_width/square_size):
 current_col = 0
 current_row = 0
 except Exception as e:
 print("Something Went Wrong...")
 print(e)

def setup_mode(sw):
 global mode_entry, mode_label, square_size, grid_size, char_count, last_drawn
 global root, color_mode_var, logging, paused, canvas_width, canvas_height, canvas
 global grid_size_label, grid_size_entry, font_size_label, font_size_entry, adjust_button
 global save_Sbutton, load_button, char_label, char_entry, char_note_label, char_note, char_button
```

```
global refresh_button, undo_button, control_frame, submit_button, col, color_
current_row = 0
current_col = 0
color_ = 1

Define the size of the squares and the grid
square_size = 70
grid_size = 10
char_count = 0
last_drawn = []

window_height = square_size * grid_size + (square_size * 2)
window_width = square_size * grid_size

if(sw == 1):
 # Create a tkinter window
 root = tk.Tk()
 root.title("C22")
 #root.iconbitmap("logo.ico")
 sw = 0
 switch = 0

 ###

 # Get screen width and height
 screen_width = root.winfo_screenwidth()
 screen_height = root.winfo_screenheight()

 # Calculate position of the window
 position_top = int(screen_height / 2 - window_height / 2)
 position_right = int(screen_width / 2 - window_width / 2)

 # Set the dimensions of the window and where it is placed
 root.geometry(f'{window_width}x{window_height}+{position_right}+{position_top}')

 ###

 # Create a BooleanVar for the color mode and set it to False initially
 color_mode_var = BooleanVar(value=False)

 # Setup logging
 logging.basicConfig(filename='logging.txt', level=logging.INFO)

 paused = tk.BooleanVar()
 paused.set(False)

 canvas_width = grid_size * square_size
```

```

canvas_height = grid_size * square_size
canvas = tk.Canvas(root, width=canvas_width, height=canvas_height)
col = canvas_width
canvas.pack()
canvas.bind("<Button-1>", on_canvas_click)

In the function where you create the canvas...
canvas.bind("<Key>", on_key_press)
canvas.focus_set()

Create input fields and buttons for grid and font size input
grid_size_label = tk.Label(root, text="Grid size:")
grid_size_entry = tk.Entry(root, width=5)
font_size_label = tk.Label(root, text="Font size:")
font_size_entry = tk.Entry(root, width=5)
adjust_button = tk.Button(root, text='Adjust Grid & Font', command=adjust_grid_and_font)

Create the save and load buttons
save_Sbutton = tk.Button(root, text='Save Session', command=save_session, bg="white", padx=5,
pady=0)
load_button = tk.Button(root, text='Load Session', command=load_session, bg="white", padx=5, pady=0)

Create input fields and buttons for mode and character input
mode_label = tk.Label(root, text="Enter ... 'IDE' for an\nInteractive Development Environment\nfor
Learning anything!")
mode_entry = tk.Entry(root)
submit_button = tk.Button(root, text='LEARN', command=submit_mode)

"""

filename = simpledialog.askstring("Save Canvas", "Enter filename:")
if filename:
"""

char_label = tk.Label(root, text="Enter symbol/s to draw: ")
char_entry = tk.Entry(root, width=12)
char_note_label = tk.Label(root, text="Enter Note: ")
char_note = tk.Entry(root, width=12)
#char_button = tk.Button(root, text='Draw Character', command=draw_char)

refresh_button = tk.Button(root, text='Refresh', command=refresh_canvas, bg="white", padx=5, pady=0)
undo_button = tk.Button(root, text='Undo', command=undo_last, bg="white", padx=5, pady=0)

control_frame = tk.Frame(root)

mode_label.pack()
mode_entry.pack()
submit_button.pack()
#create_menu(root)

```

```

Run the tkinter main loop
root.mainloop()

global switch
switch = 1

setup_mode(switch)

----- Content of GraphPlotter.py-----
import tkinter as tk
from tkinter import ttk, filedialog
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.lines import Line2D # Import this at the top of your file

class PlotterApp:
 def __init__(self, master):
 self.master = master
 self.master.title("2D & 3D Graph Plotter")

 self.plot_mode = tk.StringVar(value="2D")

 self.frame_input = ttk.Frame(self.master, padding="10")
 self.frame_input.grid(row=0, column=0, sticky=(tk.W, tk.E))

 ttk.Label(self.frame_input, text="Function:").grid(row=0, column=0, sticky=tk.W)
 self.entry_function = ttk.Entry(self.frame_input, width=30)
 self.entry_function.grid(row=0, column=1)

 ttk.Label(self.frame_input, text="Label:").grid(row=1, column=0, sticky=tk.W)
 self.entry_label = ttk.Entry(self.frame_input, width=30)
 self.entry_label.grid(row=1, column=1)

 self.button_add = ttk.Button(self.frame_input, text="Add", command=self.add_function)
 self.button_add.grid(row=0, column=2, rowspan=2)

 self.button_plot = ttk.Button(self.frame_input, text="Plot All", command=self.plot_graph)
 self.button_plot.grid(row=0, column=3, rowspan=2)

 self.button_save = ttk.Button(self.frame_input, text="Save Plot", command=self.save_plot)
 self.button_save.grid(row=0, column=4, rowspan=2)

 self.radio_2d = ttk.Radiobutton(self.frame_input, text="2D", variable=self.plot_mode, value="2D")
 self.radio_2d.grid(row=0, column=5)

 self.radio_3d = ttk.Radiobutton(self.frame_input, text="3D", variable=self.plot_mode, value="3D")

```

```

self.radio_3d.grid(row=1, column=5)

self.functions = []
self.labels = []

self.fig = Figure(figsize=(12, 6), dpi=100)
self.ax_2d = self.fig.add_subplot(122)
self.ax_3d = self.fig.add_subplot(121, projection='3d')

self.canvas = FigureCanvasTkAgg(self.fig, self.master)
self.canvas.get_tk_widget().grid(row=1, column=0)

def add_function(self):
 func = self.entry_function.get()
 label = self.entry_label.get()
 self.functions.append((func, self.plot_mode.get()))
 self.labels.append(label)
 self.entry_function.delete(0, tk.END)
 self.entry_label.delete(0, tk.END)

def plot_graph(self):
 self.ax_2d.clear()
 self.ax_3d.clear()
 plotted_2d = False
 proxy_3d = []

for i in range(len(self.functions)):
 func, mode = self.functions[i]
 label = self.labels[i]

 if mode == "2D":
 x = np.linspace(-10, 10, 400)
 y = eval(func.replace("x", "x"))
 line, = self.ax_2d.plot(x, y)
 line.set_label(label) # Set label for 2D plot
 plotted_2d = True

 elif mode == "3D":
 x = np.linspace(-10, 10, 40)
 y = np.linspace(-10, 10, 40)
 x, y = np.meshgrid(x, y)
 z = eval(func.replace("x", "x").replace("y", "y"))
 surface = self.ax_3d.plot_surface(x, y, z)
 proxy = Line2D([0], [0], linestyle='none', c='b', marker='o') # Create a proxy artist for the legend
 proxy_3d.append((proxy, label))

if plotted_2d:
 self.ax_2d.legend()

if proxy_3d:

```

```

 self.ax_3d.legend([x[0] for x in proxy_3d], [x[1] for x in proxy_3d])

 self.canvas.draw()

def save_plot(self):
 file_path = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "*.png")])
 if file_path:
 self.fig.savefig(file_path)

if __name__ == "__main__":
 root = tk.Tk()
 app = PlotterApp(root)
 root.mainloop()

----- Content of html-creator.py-----
import tkinter as tk
from tkinter import simpledialog, messagebox

def generate_html():
 # Fetch the integer values
 try:
 int1 = int(entry_int1.get())
 int2 = int(entry_int2.get())
 except ValueError:
 messagebox.showerror("Error", "Please enter valid integers!")
 return

 # Fetch the corresponding text
 text = entry_text.get()

 # Remove undesired character combinations
 chars_to_remove = entry_remove.get().split(',')
 for char in chars_to_remove:
 text = text.replace(char, "")

 # Format the HTML content
 html_content = f"""
{int1}.{int2}

{text}

"""

 # Append to the existing content
 textarea.insert(tk.END, html_content)

def save_html():
 content = textarea.get("1.0", tk.END)
 with open('output.html', 'a') as f:
 f.write(content)

```

```
messagebox.showinfo("Info", "HTML saved as output.html")

GUI setup
root = tk.Tk()
root.title("HTML Generator")

Label and Entry for Integer1
label_int1 = tk.Label(root, text="Enter first integer:")
label_int1.pack(pady=5)
entry_int1 = tk.Entry(root)
entry_int1.pack(pady=5)

Label and Entry for Integer2
label_int2 = tk.Label(root, text="Enter second integer:")
label_int2.pack(pady=5)
entry_int2 = tk.Entry(root)
entry_int2.pack(pady=5)

Label and Entry for text
label_text = tk.Label(root, text="Enter the text:")
label_text.pack(pady=5)
entry_text = tk.Entry(root, width=50)
entry_text.pack(pady=5)

Label and Entry for characters to remove
label_remove = tk.Label(root, text="Enter characters/combinations to remove (comma-delimited):")
label_remove.pack(pady=5)
entry_remove = tk.Entry(root, width=50)
entry_remove.pack(pady=5)

Button to generate HTML
generate_btn = tk.Button(root, text="Generate HTML", command=generate_html)
generate_btn.pack(pady=10)

Textarea to show the generated HTML
textarea = tk.Text(root, width=60, height=20)
textarea.pack(pady=10)

Button to save the HTML content to a file
save_btn = tk.Button(root, text="Save to HTML File", command=save_html)
save_btn.pack(pady=10)

root.mainloop()
```

----- Content of html-formation.py-----

```
import tkinter as tk
from tkinter import simpledialog, messagebox

def generate_html():
```

```
Fetch the integer values
try:
 int1 = int(entry_int1.get())
 int2 = int(entry_int2.get())
except ValueError:
 messagebox.showerror("Error", "Please enter valid integers!")
 return

Fetch the corresponding text
text = entry_text.get()

Format the HTML content
html_content = f"""
{int1}.{int2}

{text}

"""

Append to the existing content
textarea.insert(tk.END, html_content)

def save_html():
 content = textarea.get("1.0", tk.END)
 with open('output.html', 'w') as f:
 f.write(content)
 messagebox.showinfo("Info", "HTML saved as output.html")

GUI setup
root = tk.Tk()
root.title("HTML Generator")

Label and Entry for Integer1
label_int1 = tk.Label(root, text="Enter first integer:")
label_int1.pack(pady=5)
entry_int1 = tk.Entry(root)
entry_int1.pack(pady=5)

Label and Entry for Integer2
label_int2 = tk.Label(root, text="Enter second integer:")
label_int2.pack(pady=5)
entry_int2 = tk.Entry(root)
entry_int2.pack(pady=5)

Label and Entry for text
label_text = tk.Label(root, text="Enter the text:")
label_text.pack(pady=5)
entry_text = tk.Entry(root, width=50)
entry_text.pack(pady=5)

Button to generate HTML
```

```
generate_btn = tk.Button(root, text="Generate HTML", command=generate_html)
generate_btn.pack(pady=10)

Textarea to show the generated HTML
textarea = tk.Text(root, width=60, height=20)
textarea.pack(pady=10)

Button to save the HTML content to a file
save_btn = tk.Button(root, text="Save to HTML File", command=save_html)
save_btn.pack(pady=10)

root.mainloop()
```

----- Content of html-navigation.py-----

```
GenerateNavigation.py

def generate_html_content(total_pages):
 html_content = """
 <!DOCTYPE html>
 <html lang="en">
 <head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Navigation</title>
 <style>
 body, html {
 width: 100%;
 height: 100%;
 margin: 0;
 padding: 0;
 display: flex;
 flex-wrap: wrap;
 justify-content: center;
 align-content: flex-start;
 overflow-y: auto;
 }
 .btn {
 flex: 1 1 calc(33.33%- 20px);
 height: calc(25%- 20px);
 margin: 10px;
 background-color: #007BFF;
 color: white;
 text-align: center;
 display: flex;
 justify-content: center;
 align-items: center;
 text-decoration: none;
 font-size: 1em;
 transition: background-color 0.3s;
```

```
 }
 .btn:hover {
 background-color: #0056b3;
 }
</style>
</head>
<body>
"""

for i in range(1, total_pages + 1):
 html_content += f'{{i}}\n'

html_content += """
</body>
</html>
"""

return html_content

def save_to_file(filename, content):
 with open(filename, 'w') as f:
 f.write(content)

if __name__ == "__main__":
 total_pages = int(input("Enter the total number of pages: "))
 html_content = generate_html_content(total_pages)
 save_to_file("navigation.html", html_content)
 print("Generated 'navigation.html'")
```

----- Content of imageProcessing.py-----

```
import cv2
import numpy as np
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk

class ShapeClassifierApp:
 def __init__(self, root):
 self.root = root
 self.root.title("Shape Classifier")

 # Create widgets
 self.load_button = tk.Button(root, text="Load Image", command=self.load_image)
 self.detect_button = tk.Button(root, text="Detect Shapes", command=self.detect_shapes)
 self.canvas = tk.Canvas(root, width=400, height=400)
 self.canvas.pack()

 # Initialize variables
 self.image_path = None
```

```
self.image = None

Place widgets in the GUI
self.load_button.pack()
self.detect_button.pack()

def load_image(self):
 self.image_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg *.png *.bmp")])
 if self.image_path:
 self.image = Image.open(self.image_path)
 self.display_image(self.image)

def display_image(self, img):
 img = ImageTk.PhotoImage(img)
 self.canvas.create_image(0, 0, anchor=tk.NW, image=img)
 self.canvas.image = img # Keep a reference to prevent garbage collection

def detect_shapes(self):
 if self.image is None:
 messagebox.showwarning("Warning", "Please load an image first.")
 return

 # Convert the image to grayscale
 gray = cv2.cvtColor(np.array(self.image), cv2.COLOR_RGB2GRAY)

 # Apply edge detection to find contours
 edges = cv2.Canny(gray, threshold1=30, threshold2=100)

 # Find contours in the image
 contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

 # Loop through the contours and classify shapes
 for contour in contours:
 epsilon = 0.04 * cv2.arcLength(contour, True)
 approx = cv2.approxPolyDP(contour, epsilon, True)
 num_vertices = len(approx)

 x, y, _, _ = cv2.boundingRect(contour)
 shape = self.classify_shape(num_vertices)

 # Draw the shape name on the image
 cv2.putText(np.array(self.image), shape, (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

 # Display the image with classified shapes
 self.display_image(self.image)

def classify_shape(self, num_vertices):
 if num_vertices == 3:
 return "Triangle"
 elif num_vertices == 4:
```

```

 return "Rectangle"
 elif num_vertices == 5:
 return "Pentagon"
 else:
 return "Unknown"

if __name__ == "__main__":
 root = tk.Tk()
 app = ShapeClassifierApp(root)
 root.mainloop()

----- Content of json-Mathematics.py-----
import json
import os

def process_math_data(data):
 # Process the mathematical data
 # This function is a placeholder for whatever processing you need to do
 # For example, you could perform calculations or modify the data in some way
 return data

Reading JSON data
userd = input("Path to .json file: ")

Check if the file exists
if not os.path.exists(userd):
 print("File not found. Please check the path and try again.")
else:
 try:
 with open(userd, 'r') as file:
 math_data = json.load(file)

 # Processing the data
 processed_data = process_math_data(math_data)

 # Writing JSON data
 new_filename = "result_" + os.path.basename(userd)
 with open(new_filename, 'w') as file:
 json.dump(processed_data, file, indent=4)

 print(f"Processed data has been saved to {new_filename}")
 except json.JSONDecodeError:
 print("Error reading JSON file. Please ensure the file contains valid JSON.")
 except Exception as e:
 print(f"An unexpected error occurred: {e}")

```

----- Content of json-to-csv.py-----

```
import json
```

```
import pandas as pd
import os

def convert_json_to_csv(json_path):
 # Check if the file exists
 if not os.path.exists(json_path):
 print("File not found. Please check the path and try again.")
 return

 # Attempt to load JSON data
 try:
 with open(json_path, 'r') as file:
 data = json.load(file)

 # Convert to DataFrame
 df = pd.DataFrame(data)

 # Generate CSV file name
 csv_filename = os.path.basename(json_path).replace('.json', '.csv')

 # Save to CSV
 df.to_csv(csv_filename, index=False)
 print(f"CSV file saved as {csv_filename}")
 except json.JSONDecodeError:
 print("Error reading JSON file. Please ensure the file contains valid JSON.")
 except ValueError as e:
 print(f"Error converting to DataFrame: {e}")
 except Exception as e:
 print(f"An unexpected error occurred: {e}")

 # Load JSON file
 userd = input("Enter path to .json file: ")
 convert_json_to_csv(userd)
```

----- Content of json-txt.py -----

```
import json
import os

def process_json_to_txt(json_file, txt_file):
 # Open the JSON file and read data
 with open(json_file, 'r') as file:
 data = json.load(file)

 # Open the text file for writing
 with open(txt_file, 'w') as file:
 def write_data(key, value, indent_level):
 # Write the key if it's not empty
 if key:
 file.write(' ' * indent_level + key + '\n')

 # Write the value if it's not empty
 if value:
 file.write(' ' * (indent_level + 1) + str(value) + '\n')

 # Write each item in the data
 for item in data:
 write_data(item['key'], item['value'])
```

```

Check the type of value
if isinstance(value, dict):
 for sub_key, sub_value in value.items():
 write_data(sub_key, sub_value, indent_level + 1)
elif isinstance(value, list):
 for item in value:
 if isinstance(item, (dict, list)):
 write_data("", item, indent_level + 1)
 else:
 file.write(' ' * (indent_level + 1) + str(item) + '\n')
else:
 file.write(' ' * (indent_level + 1) + str(value) + '\n')

Write a blank line
file.write('\n')

Start writing data recursively
if isinstance(data, dict):
 for k, v in data.items():
 write_data(k, v, 0)
elif isinstance(data, list):
 for item in data:
 write_data("", item, 0)

print(f"Data has been written to {txt_file}")

Ask the user for the JSON file path
user_json_file = input("Enter path to .json file: ")

Check if the file exists
if os.path.exists(user_json_file):
 # Construct the output file name
 output_txt_file = os.path.splitext(user_json_file)[0] + '.txt'
 process_json_to_txt(user_json_file, output_txt_file)
else:
 print("File not found. Please check the path and try again.")

```

----- Content of Knuth-Morris-Pratt-fsm-n-state.py-----

```

def compute_lps_array(sequence):
 lps = [0] * len(sequence)
 length = 0
 i = 1

 while i < len(sequence):
 if sequence[i] == sequence[length]:
 length += 1
 lps[i] = length
 i += 1

```

```

else:
 if length != 0:
 length = lps[length- 1]
 else:
 lps[i] = 0
 i += 1
return lps

def kmp_search(input_stream, sequence):
 lps = compute_lps_array(sequence)
 output = []

 i, j = 0, 0
 while i < len(input_stream):
 if sequence[j] == input_stream[i]:
 i += 1
 j += 1
 if j == len(sequence):
 output.append(1)
 j = lps[j- 1]
 elif i < len(input_stream) and sequence[j] != input_stream[i]:
 if j != 0:
 j = lps[j- 1]
 else:
 output.append(0)
 i += 1
 while len(output) < len(input_stream):
 output.append(0)
 return output

def main():
 sequence = input("Enter the binary sequence to be detected: ")
 while not all(c in '01' for c in sequence):
 sequence = input("Invalid input. Enter a valid binary sequence: ")

 input_stream = input("Enter the input stream: ")

 print(f"Input Stream: {input_stream}")
 print("Output Stream: ", end="")

 output_stream = kmp_search(input_stream, sequence)
 for output in output_stream:
 print(output, end="")

 print()

if __name__ == "__main__":
 main()

```

```

----- Content of Language.py-----
def read_file(filename):
 with open(filename, 'r') as file:
 return [line.strip() for line in file]

def parse_logic_line(line):
 statement, truth_value, binary_id = line.rsplit(' ', 2)
 return statement, truth_value, binary_id

def evaluate_logic(logic_list, words_dict):
 for logic, truth_value, binary_id in logic_list:
 # Evaluate the logic expression
 eval_result = eval(logic)
 # If the evaluation result matches the desired truth value, process the statement
 if eval_result == (truth_value == '1'):
 # Substitute numbers with words
 for num, word in words_dict.items():
 logic = logic.replace(num, word)
 # Output the result
 with open('enumerated_outputs.txt', 'a') as output_file:
 output_file.write(f"{logic} {binary_id}\n")

def evaluate_logic_sentence(logic_list, words_dict):
 for logic, truth_value, binary_id in logic_list:
 # Evaluate the logic expression
 eval_result = eval(logic)
 # If the evaluation result matches the desired truth value, process the statement
 if eval_result == (truth_value == '1'):
 # Substitute numbers with words
 sentence = logic
 for num, word in words_dict.items():
 sentence = sentence.replace(num, word)
 """
 # Output the result to enumerated_outputs.txt
 with open('enumerated_outputs.txt', 'a') as output_file:
 output_file.write(f"{sentence} {binary_id}\n")
 """

 # Output the sentence to sentences.txt
 with open('e_sentences.txt', 'a') as sentences_file:
 sentences_file.write(f"{sentence}\n")

Parse logic.txt to create a list of tuples containing logic expressions, truth values, and binary identifiers
logic_file = read_file('logic.txt')
logic_list = [parse_logic_line(line) for line in logic_file if line]

Parse words.txt to create a dictionary mapping numbers to words
words_file = read_file('words.txt')
words_dict = {line.split()[0]: line.split()[1] for line in words_file if line}

```

```

Evaluate logic for each logic expression and compare against truth value
evaluate_logic(logic_list, words_dict)
evaluate_logic_sen(logic_list, words_dict)

----- Content of languageCreator.py-----
import tkinter as tk
from tkinter import filedialog, simpledialog, Text
from tkinter.colorchooser import askcolor

class CombinedApp:

 def __init__(self, root):
 self.root = root
 self.root.title("Custom Language & Drawing App")

 self.canvas_bg = "#ffffff"
 self.ink_color = "#000000"
 self.character_map = {} # Key: key_combination, Value: filename

 self.canvas = tk.Canvas(root, bg=self.canvas_bg, width=800, height=600)
 self.canvas.pack(pady=20, padx=20, side=tk.LEFT)
 self.canvas.bind("<B1-Motion>", self.paint)

 self.text_editor = Text(self.root, wrap='word')
 self.text_editor.pack(pady=20, padx=20, side=tk.RIGHT, fill=tk.BOTH, expand=True)
 self.root.bind("<Key>", self.insert_custom_character)

 menu = tk.Menu(root)
 root.config(menu=menu)

 file_menu = tk.Menu(menu)
 menu.add_cascade(label="File", menu=file_menu)
 file_menu.add_command(label="Save Character", command=self.save_character)
 file_menu.add_command(label="Load Character", command=self.load_character)

 colorMenu = tk.Menu(menu)
 menu.add_cascade(label="Colors", menu=colorMenu)
 colorMenu.add_command(label="Canvas Background", command=self.change_canvas_bg)
 colorMenu.add_command(label="Ink Color", command=self.change_ink_color)

 editMenu = tk.Menu(menu)
 menu.add_cascade(label="Edit", menu=editMenu)
 editMenu.add_command(label="Clear Canvas", command=self.clear_canvas)

 def paint(self, event):
 x1, y1 = (event.x- 1), (event.y- 1)
 x2, y2 = (event.x + 1), (event.y + 1)
 self.canvas.create_oval(x1, y1, x2, y2, fill=self.ink_color, outline=self.ink_color, width=2)

```

```

def change_canvas_bg(self):
 color = askcolor()[1]
 if color:
 self.canvas_bg = color
 self.canvas.config(bg=self.canvas_bg)

def change_ink_color(self):
 color = askcolor()[1]
 if color:
 self.ink_color = color

def clear_canvas(self):
 self.canvas.delete("all")

def save_character(self):
 filename = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files", "*.png")])
 if filename:
 x = self.root.winfo_rootx() + self.canvas.winfo_x()
 y = self.root.winfo_rooty() + self.canvas.winfo_y()
 x1 = x + self.canvas.winfo_width()
 y1 = y + self.canvas.winfo_height()
 self.root.grab_set() # Prevent interaction with other windows
 self.root.after(100, self.root.grab_release) # Release after 100ms

 self.root.update()
 from PIL import ImageGrab
 img = ImageGrab.grab(bbox=(x, y, x1, y1))
 img.save(filename)

 key_combination = simpledialog.askstring("Input", "Which key should this character map to?")
 if key_combination:
 self.character_map[key_combination] = filename

def load_character(self):
 filename = filedialog.askopenfilename(filetypes=[("PNG files", "*.png")])
 if filename:
 key_combination = simpledialog.askstring("Input", "Which key should this character map to?")
 if key_combination:
 self.character_map[key_combination] = filename

def insert_custom_character(self, event):
 key_combination = event.keysym
 if key_combination in self.character_map:
 filename = self.character_map[key_combination]
 img = tk.PhotoImage(file=filename)
 self.text_editor.image_create(tk.END, image=img)
 # To prevent garbage collection of the image
 self.text_editor.img = img

if __name__ == "__main__":

```

```
root = tk.Tk()
app = CombinedApp(root)
root.mainloop()

----- Content of LE.py-----
import tkinter as tk
import random
from PIL import ImageGrab
import pickle
from tkinter import simpledialog
import os
from tkinter import Menu, Checkbutton, BooleanVar
from tkinter.colorchooser import askcolor
import logging
import time
import json
import os
import pyautogui
import keyboard
from tkinter import filedialog
from PIL import Image, ImageTk

current_col = 0
current_row = 0
resized_images = [] # Global variable to store resized images

Rest of your code...
Global variable for the image selection frame
image_selection_frame = None
selected_image = None # Global variable to hold the currently selected image
Global variable to keep track of the current mode
current_mode = "draw_char" # Possible values: "draw_char", "image_mode"

def toggle_mode():
 global current_mode, canvas
 if current_mode == "draw_char":
 current_mode = "image_mode"
 canvas.unbind("<Button-1>") # Unbind draw_char function
 canvas.bind("<Button-1>", on_canvas_click_for_image) # Bind image drawing function
 else:
 current_mode = "draw_char"
 canvas.unbind("<Button-1>") # Unbind image drawing function
 #canvas.bind("<Button-1>", lambda event: draw_char(event.x // square_size, event.y // square_size))
Bind draw_char function
 canvas.bind("<Button-1>", lambda event: draw_char(event.y // square_size, event.x // square_size))

def select_image(img):
 global selected_image
```

```
selected_image = img
You can add additional logic here, e.g., updating the UI to indicate the selected image

def resize_and_add_image(file_path, size):
 try:
 with Image.open(file_path) as img:
 # Resize and add to the list
 #resized_img = img.resize(size, Image.Resampling.LANCZOS) # For Pillow versions 8.0.0 and later
 #resized_img = img.resize(size, Image.LANCZOS) # For older versions of Pillow
 resized_img = img.resize(size, Image.LANCZOS)
 resized_images.append(resized_img)
 update_image_selection_area()
 except Exception as e:
 print(f"Error loading image: {e}")

 """
 def on_canvas_click(event):
 # Calculate the sub-square coordinates
 col = event.x // square_size
 row = event.y // square_size

 if selected_image:
 draw_image_on_canvas(row, col, selected_image)
 """

def on_canvas_click_for_image(event):
 global selected_image
 if selected_image:
 col = event.x // square_size
 row = event.y // square_size
 draw_image_on_canvas(row, col, selected_image)

def draw_image_on_canvas(row, col, image):
 x = col * square_size
 y = row * square_size

 resized_image = image.resize((square_size, square_size), Image.LANCZOS)
 tk_image = ImageTk.PhotoImage(resized_image)

 canvas.create_image(x, y, image=tk_image, anchor='nw')

 if not hasattr(canvas, 'images'):
 canvas.images = []
 canvas.images.append(tk_image) # Keep a reference

 """
 # Function to update the image selection area
 def update_image_selection_area():

```

```

for img in resized_images:
 # Convert PIL image to Tkinter PhotoImage
 tk_image = ImageTk.PhotoImage(img)
 btn = tk.Button(image_selection_frame, image=tk_image, command=lambda img=img:
select_image(img))
 btn.image = tk_image # Keep a reference to avoid garbage collection
 btn.pack(side='left')
"""

def update_image_selection_area():
 global image_selection_frame

 # Clear existing buttons in the frame
 for widget in image_selection_frame.winfo_children():
 widget.destroy()

 # Create buttons for each image
 for img in resized_images:
 tk_image = ImageTk.PhotoImage(img)
 btn = tk.Button(image_selection_frame, image=tk_image, command=lambda img=img:
select_image(img))
 btn.image = tk_image # Keep a reference to avoid garbage collection
 btn.pack(side='left')

def select_and_add_images():

 # Define a target size for the images
 wid = int(input("Target Width: "))
 hei = int(input("Target Height: "))
 target_size = (wid, hei) # You can change this size as needed

 file_paths = filedialog.askopenfilenames(filetypes=[("PNG files", "*.png")]) # Enable multi-file selection
 if file_paths:
 for file_path in file_paths:
 # Assuming you have a predefined target size for the images
 resize_and_add_image(file_path, target_size)

def resize_and_add_image(file_path, size):
 try:
 with Image.open(file_path) as img:
 # Use Image.Resampling.LANCZOS for Pillow versions 8.0.0 and later
 resized_img = img.resize(size, Image.Resampling.LANCZOS)
 # For older versions of Pillow, use Image.LANCZOS
 # resized_img = img.resize(size, Image.LANCZOS)
 resized_images.append(resized_img)
 update_image_selection_area() # Update the display area with new images
 except Exception as e:
 print(f"Error loading image: {e}")

```

```

def select_and_resize_image():
 # Ask the user to select an image file
 file_path = filedialog.askopenfilename(filetypes=[("PNG files", "*.png")])
 if not file_path:
 return # User cancelled the dialog

 # Prompt the user for the new size
 new_size = simpledialog.askstring("Resize Image", "Enter new size (width,height):")
 if not new_size:
 return # User cancelled the dialog

 try:
 # Parse the size input and resize the image
 width, height = map(int, new_size.split(','))
 resize_image(file_path, (width, height))
 except Exception as e:
 tk.messagebox.showerror("Error", f"An error occurred: {e}")

def resize_image(input_path, size):
 try:
 # Construct a new file name based on the original path
 dir_name, file_name = os.path.split(input_path)
 name, ext = os.path.splitext(file_name)
 output_path = os.path.join(dir_name, f"{name}_resized{ext}")

 # Open, resize, and save the image
 with Image.open(input_path) as img:
 # Use Image.Resampling.LANCZOS for Pillow versions 8.0.0 and later
 img = img.resize(size, Image.Resampling.LANCZOS)
 # For older versions of Pillow, use Image.LANCZOS
 # img = img.resize(size, Image.LANCZOS)
 img.save(output_path)

 tk.messagebox.showinfo("Success", f"Image saved successfully to {output_path}")
 except Exception as e:
 tk.messagebox.showerror("Error", f"An error occurred: {e}")

def get_random_color():
 r = lambda: random.randint(0,255)
 return '#%02X%02X%02X' % (r(),r(),r())

def get_random_char():
 random_int = random.randint(0x0021, 0x007E)
 return chr(random_int)

def draw_grid():
 for i in range(grid_size):

```

```

for j in range(grid_size):
 color = get_random_color()
 square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size, (i+1)*square_size,
fill=color)
 char = get_random_char()
 text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2, text=char,
font=("Arial", 18), anchor="center")

def draw_grid_IDE():
 global col
 try:
 for i in range(grid_size):
 for j in range(grid_size):
 color = "blue"
 square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size,
(i+1)*square_size, fill=color, outline = "white")
 if typing_mode:
 # In the function where you create the canvas...
 canvas.bind("<Key>", on_key_press)
 canvas.focus_set()
 except:
 print("Submit an app mode ...")

def toggle_color_mode():
 """Toggles between manual color mode and random color mode"""
 global color_mode_var
 color_mode_var.set(not color_mode_var.get())

font_size = 9

def draw_char1(canvas, char, row, col, font_size, color):
 global cell_size, font_color, square_size, grid_size, last_drawn, note, char_note

 ...
 def draw_char1(canvas, char, row, col, font_size):

Draws a character on the Tkinter canvas at the specified row and column with the specified font size.
```

**Parameters:**

canvas (tk.Canvas): The Tkinter canvas to draw on.  
 char (str): The character to draw.  
 row (int): The row to draw the character at.  
 col (int): The column to draw the character at.  
 font\_size (int): The font size to use for the character.

...

```

try:
 x = col * square_size + square_size / 2
 y = row * square_size + square_size / 2

 print(f"Drawing char '{char}' at ({x},{y})") # Debug print to check coordinates
 canvas.create_rectangle(col * square_size, row * square_size, (col+1) * square_size, (row+1) * square_size, fill=color)
 canvas.create_text(x, y, text=char, fill=font_color, font=('Calibri', int(font_size)))
 note = char_note.get()
 # Log the color and character info
 logging.info(f"Position:{x};{y};Xp:{x};Yp:{y};Character:{char};Square color:{color};Font color:{font_color};Note:{note}")
 # Log the color and character info in a text file
 if(note == ""):
 note = "emptyNote"
 with open("color_log.txt", "a", encoding='utf-8') as login:
 print(f"{x},{y},{char},{color},{font_color},{note}", file=login)

except Exception as e:
 print(f"Something Went Wrong... Error: {e}")

def draw_char(i=None, j=None):
 try:
 """Draws a character from input field in a specific square"""
 global char_count, IDE_mode, typing_mode, last_drawn, font_color
 if(mode == 'IDE'):
 IDE_mode = True
 # Check if last_drawn is defined, otherwise define it
 if 'last_drawn' not in globals():
 last_drawn = []
 # If i, j are not provided, calculate them based on char_count
 if i is None or j is None:
 i, j = divmod(char_count, grid_size)
 # Handle out-of-grid situations
 if i >= grid_size or j >= grid_size:
 print('Out of grid!')
 return
 # Generate a random color if not in typing mode, white otherwise
 ##color = get_random_color() if not typing_mode else "white"

 # Check color mode
 if color_mode_var.get():
 # Manual color mode
 # Show a color picker and get the chosen color for the square
 color = askcolor(title="Choose square color")[1]

```

```

Ask for the font color
color_result = askcolor(title="Choose font color")
if color_result is not None:
 font_color = color_result[1]
else:
 # Handle the case when the user cancelled the color selection
 font_color = "#000000" # default to black, for example

else:
 # Random color mode
 color = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
 color1 = "#" + "".join([random.choice('0123456789ABCDEF') for i in range(6)])
 font_color = color1 "##000000"

square = canvas.create_rectangle(j*square_size, i*square_size, (j+1)*square_size, (i+1)*square_size,
fill=color)
char = char_entry.get()[:1]
note = char_note.get()
text = canvas.create_text(j*square_size + square_size/2, i*square_size + square_size/2, text=char,
font=("Arial", font_size), fill=font_color, anchor="center")

...
Log the color and character info
logging.info(f"Position:{i};{j},Xp:{i},Yp:{j},Character:{char},Square color:{color},Font
color:{font_color}")
Log the color and character info in a text file
with open("color_log.txt", "a") as login:
 print(f"{i},{j},{char},{color},{font_color}", file=login)
...
Log the color and character info
logging.info(f"Position:{i};{j},Xp:{i},Yp:{j},Character:{char},Square color:{color},Font
color:{font_color},Note:{note}")
Log the color and character info in a text file
if(note == ""):
 note = "emptyNote"
with open("color_log.txt", "a", encoding='utf-8') as login:
 print(f"{i},{j},{char},{color},{font_color},{note}", file=login)

if len(char_entry.get()) > 0: # Check if there's more than one character
 char_entry.delete(0, 1) # Delete the first character

last_drawn.append((square, text))

if not IDE_mode:
 char_count = (char_count + 1) % (grid_size * grid_size)
 if char_count == 0: # If we've filled the canvas, clear it

```

```

 canvas.delete('all')

 return square, text
except:
 print("Submit an app mode ...")

def adjust_grid_and_font():
 global grid_size, square_size, font_size, canvas_width, current_row, current_col

 n_g_s = simpledialog.askstring("Change Grid Size (Grid Width)", "Enter New Grid Size (+Integer):")
 if(n_g_s != ''):
 new_grid_size = n_g_s
 n_f_s = simpledialog.askstring("Change Font Size (Default = 9)", "Enter New Font Size (+Integer):")
 if(n_f_s != ''):
 new_font_size = n_f_s
 # Update the global variables
 if(new_grid_size==None and new_font_size==None):
 grid_size = 10
 font_size = 9
 if((not new_grid_size==None) and (not new_font_size==None)):
 try:
 grid_size = int(new_grid_size)
 square_size = canvas_width / grid_size
 font_size = int(new_font_size)
 except:
 print("positive integers please")

 """
 # Get new values from input fields
 new_grid_size = grid_size_entry.get()
 new_font_size = font_size_entry.get()

 # Update the global variables
 grid_size = int(new_grid_size)
 square_size = canvas_width // grid_size
 font_size = int(new_font_size)
 """

 # Redraw the grid
 refresh_canvas()
 qit = input("Are you in typing mode, or want to enter typing mode? 1 for (No), 2 for (yes): ")
 if(qit == '2'):
 current_row = 0
 current_col = 0
 show_typing_mode_menu()

def refresh_canvas():
 """Clears the canvas and resets the char_count"""
 global char_count

```

```
canvas.delete('all')
char_count = 0
draw_grid_IDE()
last_drawn.clear()

def undo_last():
 """Undoes the last drawing operation"""
 if last_drawn:
 square, text = last_drawn.pop()
 canvas.delete(square)
 canvas.delete(text)

def update_canvas():
 if not paused.get():
 canvas.delete('all')
 draw_grid()
 root.after(8000, update_canvas)

def toggle_typing_mode():
 global typing_mode, canvas

 typing_mode = not typing_mode
 if typing_mode:
 canvas.focus_set() # Set focus to the canvas for keyboard input

def on_canvas_click(event):
 try:
 global mode, typing_mode
 if mode == 'IDE' and not typing_mode:
 j = event.x // square_size
 i = event.y // square_size
 draw_char(i, j)
 else:
 paused.set(not paused.get())
 except:
 print("Submit an app mode ...")

def on_key_press(event):
 global char_count, typing_mode
 if typing_mode:
 char_entry.delete(0, 'end') # Clear the entry box
 try:
 utf8_char = event.char.encode('utf-8').decode('utf-8')
 char_entry.insert(0, utf8_char) # Insert the typed character
 except UnicodeDecodeError:
 print("Non UTF-8 character detected")
 return
 draw_char() # Draw the character
```

```
char_count += 1 # Increment the count

If char_count exceeds the total number of squares in the grid, reset it
if char_count >= grid_size ** 2:
 char_count = 0

def submit_mode():

 global char_count, mode, typing_mode

 mode = mode_entry.get().upper()

 ...

 if mode in ['editor']:

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 draw_grid()
 char_label.pack()
 char_entry.pack()
 char_button.pack()

 ...

 # Draw the initial grid
 if(mode != 'IDE'):
 print("A valid mode ...")

 ...

 if(mode == 'normal'):

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 draw_grid()
 create_menu_1()

 if mode == 'normal':
 # Schedule the first update
 root.after(8000, update_canvas)
```

'''

```
if mode == 'IDE':

 typing_mode = False # Start in clicking mode
 mode_entry.delete(0, 'end') # Clear the input field
 mode_label.pack_forget()
 mode_entry.pack_forget()
 submit_button.pack_forget()

 char_count = 0
 draw_grid_IDE()

 control_frame.pack()
 char_label.pack()
 char_entry.pack()
 #grid_size_label.pack()
 #grid_size_entry.pack()
 #font_size_label.pack()
 #font_size_entry.pack()
 #adjust_button.pack()
 char_note_label.pack()
 char_note.pack()
 #refresh_button.pack(side="left")
 #undo_button.pack(side="left")
 #save_button = tk.Button(root, text='Save', command=save_canvas, bg="white", padx=5, pady=0)
 #save_button.pack(side="left")
 #save_Sbutton.pack(side="left")
 #load_button.pack(side="left")
 create_menu()

return mode
```

```
def save_canvas():
 global IC_value, xSwitch, ing
 xSwitch = 1
 ing = 1
 if(ing == 0):
 ing = 2
 ing = 1
 if(xSwitch == 1 and ing == 1):
 IC_value = 0
 ing = 0
 """Save the current state of the canvas to a .png file"""
 #filename = simpledialog.askstring("Save Canvas", "Enter filename:")
 filename = f"ImageCanvas{IC_value}_rename"
```

```

if filename: # Only save the canvas if a filename was entered
 # Get the bounding box of the canvas
 x = root.winfo_rootx() + canvas.winfo_x()
 y = root.winfo_rooty() + canvas.winfo_y()
 x1 = x + canvas.winfo_width()
 y1 = y + canvas.winfo_height()
 time.sleep(3)
 # Grab the image, crop it to the bounding box, and save it
 ImageGrab.grab().crop((x, y, x1, y1)).save(filename + ".png")

def save_session():
 """Save the current session to a file using pickle"""
 global last_drawn
 # Ask the user to enter a filename
 filename = simpledialog.askstring("Save Session", "Enter filename:")
 if filename: # Only save the session if a filename was entered
 session_data = [(canvas.coords(square), canvas.itemcget(square, "fill"),
 canvas.coords(text), canvas.itemcget(text, "text"), canvas.itemcget(text, "fill"))
 for square, text in last_drawn]
 with open(filename + ".pkl", "wb") as f:
 pickle.dump(session_data, f)

def load_session():
 """Load a saved session from a file using pickle"""
 global last_drawn
 # Ask the user to enter a filename
 filename = simpledialog.askstring("Load Session", "Enter filename:")
 if filename: # Only try to load a session if a filename was entered
 try:
 with open(filename + ".pkl", "rb") as f:
 session_data = pickle.load(f)
 # Clear the canvas and redraw all elements from the loaded session
 canvas.delete('all')
 draw_grid_IDE()
 last_drawn = []
 for square_coords, square_fill, text_coords, text_content, text_fill in session_data:
 square = canvas.create_rectangle(*square_coords, fill=square_fill)
 text = canvas.create_text(*text_coords, text=text_content, fill=text_fill, font=("Arial", font_size),
 anchor="center")
 last_drawn.append((square, text))
 except FileNotFoundError:
 print(f"No session named '{filename}' found.")

#def exit_app():
exit()

def show_about():
 about_window = tk.Toplevel(root)

```

```
about_window.title("About")
about_msg = "This is a program created to learn and experiment with Tkinter. In IDE mode, you can draw
characters on a grid (By typing or copying and pasting then clicking the squares you want to draw each
character on), adjust the grid and font size, save and load sessions, and more, like saving the grid as an
image (For example). C(num) was written with the aid of Chat GPT. Enjoy!\n\nNote: the workflow ... IDE to
color_log.txt (Compiled by compile#.exe or compile#.py);\nindex_1.txt contains fields of study or research
and development and\nindex_2.txt contains fields of study or research and development also
...\n\nBertotools Digital"
tk.Message(about_window, text=about_msg, width=500).pack()
tk.Button(about_window, text="OK", command=about_window.destroy).pack()

def compileGOSmX():
 MAX_LINE_LENGTH = 256
 MAX_TOPICS = 256

 global index_1, index_2

 index_1 = [
 "Axiom",
 "Theorem",
 "Lemma",
 "Proposition",
 "Corollary",
 "Conjecture",
 "Proof",
 "Premise",
 "Conclusion",
 "Hypothesis",
 "Counterexample",
 "Direct Proof",
 "Indirect Proof",
 "Proof by Contradiction (Reductio ad absurdum)",
 "Proof by Induction",
 "Proof by Contrapositive",
 "Deductive Reasoning",
 "Inference",
 "Assumption",
 "Statement",
 "Postulate",
 "Proof by Exhaustion",
 "Syllogism",
 "Constructive Proof",
 "Non-Constructive Proof",
 "Trivial Proof",
 "Vacuous Proof",
 "Biconditional",
 "Condition",
 "Sufficiency",
 "Necessity",
 "Quantifier",
```

"Universal Quantifier",  
"Existential Quantifier",  
"Bound Variable",  
"Free Variable",  
"Predicate",  
"Propositional Logic",  
"Modus Ponens",  
"Modus Tollens",  
"Discrete Mathematics",  
"Set Theory",  
"Function",  
"Bijection",  
"Injection",  
"Surjection",  
"Equivalence Relation",  
"Partial Order",  
"Total Order",  
"Well-Order",  
"Reflexivity",  
"Symmetry",  
"Transitivity",  
"Antisymmetry",  
"Completeness",  
"Compactness",  
"Connectedness",  
"Convergence",  
"Divergence",  
"Limit",  
"Sequence",  
"Series",  
"Monotonicity",  
"Cauchy Sequence",  
"Infinite Set",  
"Finite Set",  
"Cardinality",  
"Countable Set",  
"Uncountable Set",  
"Subset",  
"Superset",  
"Intersection",  
"Union",  
"Empty Set",  
"Power Set",  
"Cartesian Product",  
"Equivalence Class",  
"Partition",  
"Field",  
"Ring",  
"Group",  
"Abelian Group",

"Non-abelian Group",  
"Matrix",  
"Vector Space",  
"Linear Transformation",  
"Eigenvalue",  
"Eigenvector",  
"Norm",  
"Inner Product",  
"Orthogonality",  
"Basis",  
"Dimension",  
"Rank",  
"Nullity",  
"Determinant",  
"Graph Theory",  
"Vertex",  
"Edge",  
"Connectivity",  
"Cycle",  
"Path",  
"Degree",  
"Subgraph",  
"Tree",  
"Forest",  
"Planar Graph",  
"Bipartite Graph",  
"Directed Graph (Digraph)",  
"Eulerian Graph",  
"Hamiltonian Graph",  
"Adjacency Matrix",  
"Incidence Matrix",  
"Isomorphism",  
"Homeomorphism",  
"Topology",  
"Open Set",  
"Closed Set",  
"Boundary",  
"Compact Space",  
"Hausdorff Space",  
"Continuity",  
"Differential",  
"Derivative",  
"Integral",  
"Partial Derivative",  
"Multivariable Calculus",  
"Laplace Transform",  
"Fourier Transform",  
"Taylor Series",  
"Maclaurin Series",  
"Conic Sections",

"Hyperbola",  
"Ellipse",  
"Parabola",  
"Asymptote",  
"Limits at Infinity",  
"Complex Number",  
"Imaginary Unit",  
"Real Number",  
"Rational Number",  
"Irrational Number",  
"Prime Number",  
"Composite Number",  
"GCD (Greatest Common Divisor)",  
"LCM (Least Common Multiple)",  
"Permutation",  
"Combination",  
"Probability",  
"Statistics",  
"Expected Value",  
"Variance",  
"Standard Deviation",  
"Normal Distribution",  
"Poisson Distribution",  
"Binomial Distribution",  
"Hypothesis Testing",  
"Regression",  
"Correlation",  
"Matrix Algebra",  
"Linear Algebra",  
"Vector Calculus",  
"Optimization",  
"Algorithm",  
"Computational Complexity",  
"Big O Notation",  
"Pigeonhole Principle",  
"Principle of Inclusion-Exclusion",  
"Turing Machine",  
"Computability",  
"Unsolvability",  
"Parity",  
"Diophantine Equations",  
"Cryptography",  
"Fermat's Last Theorem",  
"Pythagorean Theorem",  
"Triangle Inequality",  
"Trigonometric Functions",  
"Trigonometric Identities",  
"Polar Coordinates",  
"Euler's Formula",  
"Riemann Zeta Function",

"P vs NP Problem",  
"NP-complete Problem",  
"Stochastic Process",  
"Markov Chain",  
"Random Variable",  
"Conditional Probability",  
"Bayes' Theorem",  
"Monte Carlo Method",  
"Fractal",  
"Chaos Theory",  
"Game Theory",  
"Nash Equilibrium",  
"Zero-Sum Game",  
"Non-Zero-Sum Game",  
"Linear Programming",  
"Nonlinear Programming",  
"Quadratic Programming",  
"Dynamic Programming",  
"Integer Programming",  
"Graph Coloring",  
"Network Flow",  
"Spanning Tree",  
"Bellman-Ford Algorithm",  
"Dijkstra's Algorithm",  
"Kruskal's Algorithm",  
"Prim's Algorithm",  
"Floyd-Warshall Algorithm",  
"Euler's Method",  
"Runge-Kutta Method",  
"Numerical Integration",  
"Numerical Differentiation",  
"Bisection Method",  
"Newton's Method",  
"Secant Method",  
"Fixed Point Iteration",  
"Linear Interpolation",  
"Polynomial Interpolation",  
"Lagrange Interpolation",  
"Splines",  
"Fourier Series",  
"Laplace's Equation",  
"Heat Equation",  
"Wave Equation",  
"Schrodinger Equation",  
"Ordinary Differential Equation (ODE)",  
"Partial Differential Equation (PDE)",  
"Boundary Value Problem",  
"Initial Value Problem",  
"Green's Theorem",  
"Stoke's Theorem",

"Divergence Theorem",  
"Curl",  
"Gradient",  
"Divergence",  
"Tensor",  
"Manifold",  
"Topological Space",  
"Measure Theory",  
"Lebesgue Integral",  
"Borel Set",  
"Hilbert Space",  
"Banach Space",  
"Category Theory",  
"Functor",  
"Natural Transformation",  
"Sheaf",  
"Homotopy",  
"Homology",  
"Cohomology",  
"Galois Theory",  
"Algebraic Geometry",  
"Topological K-Theory",  
"Knot Theory",  
"Lattice Theory"

]

index\_2 = [  
    "Biochemistry",  
    "Biophysics",  
    "Molecular biology",  
    "Genetics",  
    "Immunology",  
    "Cell biology",  
    "Microbiology",  
    "Neuroscience",  
    "Pharmacology",  
    "Bioinformatics",  
    "Biotechnology",  
    "Proteomics",  
    "Genomics",  
    "Structural biology",  
    "Virology",  
    "Systems biology",  
    "Developmental biology",  
    "Evolutionary biology",  
    "Synthetic biology",  
    "Metabolomics",  
    "Epigenetics",  
    "Tissue engineering",  
    "Nanotechnology",

"Materials science",  
"Quantum physics",  
"Condensed matter physics",  
"Particle physics",  
"Astrophysics",  
"Cosmology",  
"Optics",  
"Atomic and molecular physics",  
"Fluid mechanics",  
"Thermodynamics",  
"Environmental science",  
"Climate science",  
"Geology",  
"Oceanography",  
"Atmospheric science",  
"Ecology",  
"Conservation biology",  
"Botany",  
"Zoology",  
"Entomology",  
"Marine biology",  
"Paleontology",  
"Anthropology",  
"Archaeology",  
"Psychology",  
"Cognitive science",  
"Social psychology",  
"Linguistics",  
"Artificial intelligence",  
"Machine learning",  
"Computer vision",  
"Natural language processing",  
"Human-computer interaction",  
"Robotics",  
"Computer graphics",  
"Data science",  
"Mathematical modeling",  
"Mathematical physics",  
"Number theory",  
"Algebraic geometry",  
"Differential equations",  
"Computational physics",  
"Mathematical biology",  
"Operations research",  
"Biostatistics",  
"Epidemiology",  
"Cancer research",  
"Diabetes research",  
"Heart disease research",  
"Infectious diseases research",

"Immunotherapy",  
"Stem cell research",  
"Gene therapy",  
"Drug discovery",  
"Precision medicine",  
"Health informatics",  
"Renewable energy",  
"Energy storage",  
"Sustainable materials",  
"Environmental engineering",  
"Water management",  
"Transportation engineering",  
"Civil engineering",  
"Chemical engineering",  
"Aerospace engineering",  
"Biomedical engineering",  
"Electrical engineering",  
"Mechanical engineering",  
"Robotics engineering",  
"Quantum computing",  
"Cryptography",  
"Cybersecurity",  
"Network engineering",  
"Telecommunications",  
"Human genetics",  
"Forensic science",  
"Space exploration and research",  
"Planetary science",  
"Astrobiology",  
"Astrochemistry",  
"Astrogeology",  
"Astroinformatics",  
"Exoplanet research",  
"Stellar evolution",  
"Galactic astronomy",  
"Observational astronomy",  
"Computational astrophysics",  
"Quantum chemistry",  
"Computational chemistry",  
"Organic chemistry",  
"Inorganic chemistry",  
"Physical chemistry",  
"Environmental chemistry",  
"Analytical chemistry",  
"Agricultural science",  
"Food science",  
"Nutritional science",  
"Exercise physiology",  
"Sports science",  
"Biomechanics",

"Plant physiology",  
"Plant genetics",  
"Plant pathology",  
"Soil science",  
"Hydrology",  
"Geochemistry",  
"Geophysics",  
"Geomorphology",  
"Remote sensing",  
"Geotechnical engineering",  
"Petroleum engineering",  
"Aerospace materials",  
"Nanomaterials",  
"Polymer science",  
"Computational materials science",  
"Photonics",  
"Physical optics",  
"Quantum optics",  
"Neuroengineering",  
"Brain imaging",  
"Cognitive neuroscience",  
"Neuroinformatics",  
"Psychophysics",  
"Developmental psychology",  
"Personality psychology",  
"Clinical psychology",  
"Industrial-organizational psychology",  
"Educational psychology",  
"Psycholinguistics",  
"Human genetics",  
"Evolutionary genetics",  
"Population genetics",  
"Genetic engineering",  
"Genetic counseling",  
"Epigenomics",  
"Cardiovascular research",  
"Respiratory research",  
"Gastroenterology research",  
"Endocrinology research",  
"Nephrology research",  
"Hematology research",  
"Ophthalmology research",  
"Orthopedic research",  
"Dermatology research",  
"Veterinary medicine",  
"Animal behavior",  
"Conservation ecology",  
"Wildlife biology",  
"Environmental microbiology",  
"Agricultural economics",

"Development economics",  
"Behavioral economics",  
"Econometrics",  
"Financial mathematics",  
"Operations management",  
"Supply chain management",  
"Industrial engineering",  
"Human-computer interaction",  
"Virtual reality",  
"Augmented reality",  
"Data mining",  
"Text mining",  
"Big data analytics",  
"Computational linguistics",  
"Quantum information science",  
"Quantum cryptography",  
"Biometrics",  
"Information retrieval",  
"Software engineering",  
"Computer networks",  
"Embedded systems",  
"Human-robot interaction",  
"Control systems",  
"Biopharmaceuticals",  
"Drug delivery systems",  
"Clinical trials",  
"Regenerative medicine",  
"Agricultural biotechnology",  
"Plant breeding",  
"Animal breeding",  
"Food technology",  
"Sensory science",  
"Poultry science",  
"Aquaculture",  
"Marine ecology",  
"Limnology",  
"Population ecology",  
"Landscape ecology",  
"Evolutionary ecology",  
"Environmental toxicology",  
"Environmental chemistry",  
"Environmental microbiology",  
"Ecotoxicology",  
"Green chemistry",  
"Space physics",  
"Space weather",  
"Astrostatistics",  
"Computational fluid dynamics",  
"Mathematical optimization",  
"Operations research",

```
"Human genetics",
"Functional genomics",
"Molecular genetics",
"Cancer genetics",
"Psychiatric genetics",
"Population genomics",
"Bioengineering",
"Biomaterials",
"Biomechatronics",
"Cardiovascular engineering",
"Neural engineering",
"Rehabilitation engineering",
"Genetic engineering",
"Environmental engineering",
"Water resources engineering",
"Structural engineering",
"Robotics engineering",
"Quantum information theory",
"Quantum simulation",
"Quantum sensing",
"Geographical information systems (GIS)",
"Urban planning",
"Renewable energy systems",
"Solar cell technology",
"Wind energy research",
"Energy policy and economics",
"Computational neuroscience",
"Neurobiology",
"Cognitive neuroscience",
"Systems neuroscience",
"Human-robot interaction",
"Evolutionary psychology",
"Social network analysis"
]
```

```
def square_print_topic_indices(r_value, g_value, b_value, file):
 index1 = r_value % len(index_1)
 index2 = g_value % len(index_1)
 index3 = b_value % len(index_1)
 """
 file.write(f"\t\t\t{topics[index1]}\n")
 file.write(f"\t\t\t{topics[index2]}\n")
 file.write(f"\t\t\t{topics[index3]}\n")
 """
 file.write(f"\t\t\t{index_1[index1]}\n")
 file.write(f"\t\t\t{index_1[index2]}\n")
 file.write(f"\t\t\t{index_1[index3]}\n")
```

```

squareRatio = str(index1) + ":" + str(index2) + ":" + str(index3)
return squareRatio

def font_print_topic_indices(r_value, g_value, b_value, file):
 index1_ = r_value % len(index_2)
 index2_ = g_value % len(index_2)
 index3_ = b_value % len(index_2)
 file.write(f"\t\t\t{index_2[index1_]}\n")
 file.write(f"\t\t\t{index_2[index2_]}\n")
 file.write(f"\t\t\t{index_2[index3_]}\n")

fontRatio = str(index1_) + ":" + str(index2_) + ":" + str(index3_)
return fontRatio

def get_index(i, j, grid_size):
 return i * grid_size + j

def extract_rgb_values(color):
 if len(color) != 7 or color[0] != '#':
 raise ValueError('Input should be a hex color code in the format "#RRGGBB"')
 try:
 red = int(color[1:3], 16)
 green = int(color[3:5], 16)
 blue = int(color[5:7], 16)
 return red, green, blue
 except ValueError:
 raise ValueError('Invalid color code. RGB values should be hex digits (0-9, A-F)')

def GO():
 log_file = "color_log.txt"
 ## topic_file = "index_1.txt"
 ## topic2_file = "index_2.txt" # Corrected this line to read from a different file
 gridWidth = int(input("Enter gridWidth: "))
 # Read topic files to get the topics and topic2s
 ## with open(topic_file, "r", encoding='utf8') as file:
 ## topics = [line.strip() for line in file]
 ##
 ## with open(topic2_file, "r", encoding='utf8') as file:
 ## topic2s = [line.strip() for line in file]

 numTopics = len(index_1)
 numTopic2s = len(index_2)

 # Read color log file
 with open(log_file, "r") as file:
 lines = file.readlines()

```

```

lineNumber = 0
Save output to a file
output_file = "output.txt"
with open(output_file, "w") as file:
 print()
for line in lines:
 parts = line.strip().split(",")
 if len(parts) >= 5:
 xcoor = int(parts[1])
 ycoor = int(parts[0])
 index_ = (gridWidth * ycoor) + xcoor
 character = parts[2]
 squareColor = parts[3]
 fontColor = parts[4]
 notes = parts[5]

 # Extract RGB values from squareColor
 red, green, blue = extract_rgb_values(squareColor)

 # Extract RGB values from fontColor
 fontRed, fontGreen, fontBlue = extract_rgb_values(fontColor)

 # Calculate the grid size
 grid_size = int(parts[0])

 # Select topics based on RGB values from squareColor
 topicIndices = [(red + get_index(i, j, grid_size)) % numTopics for i in range(grid_size) for j in
range(grid_size)]

 lineNumber += 1

 with open(output_file, "a") as file:
 file.write(f"Compiling Line/s: {lineNumber}\n\n")
 file.write(f"Line({lineNumber})\n\n")
 file.write(f"O\tLineNumber: {lineNumber}\ttx-coordinate: {xcoor}\ty-coordinate: {ycoor}\t
Character: {character}\tRGB: {squareColor}\tRGB: {fontColor}\t Note: {notes}\t GridDimensions:
{gridWidth}x{gridWidth}\n\n")
 #print(topicIndices)
 file.write(f"Line/{lineNumber} Output:\n\n")
 file.write(f"#define Line({lineNumber}){{\n\n}")
 # Inside the main function
 file.write(f"\ti_0({squareColor}){{\n\n")
 file.write(f"\t\t{character}\n")
 file.write(f"\t\t(R: {red}, G: {green}, B: {blue})\n")
 s_ratio = square_print_topic_indices(red, green, blue, file)
 #square_print_topic_indices(red, green, blue, topics, file) # Modify this line
 file.write("\t)\n")
 file.write("\t;\n\n")
 file.write(f"\ti_1({fontColor}){{\n\n")

```

```
file.write(f"\t\t{character}\n")
file.write(f"\t\t(R: {fontRed}, G: {fontGreen}, B: {fontBlue})\n")
f_ratio = font_print_topic_indices(fontRed, fontGreen, fontBlue, file)
#font_print_topic_indices(fontRed, fontGreen, fontBlue, topic2s, file) # And this line
file.write("\t\t)\n")
file.write(f"\treturn squareRatio({s_ratio}), fontRatio({f_ratio}), sqaureIndex({index_})\n")
#file.write(f"\treturn fontRatio({f_ratio})\n")
file.write("}\n\n")
```

GO()

```
def create_menu():
 global control_frame
 def home():
 canvas.pack_forget()
 menubar.destroy()
 control_frame.pack()
 char_entry.pack_forget()
 char_label.pack_forget()
 #grid_size_label.pack_forget()
 #grid_size_entry.pack_forget()
 #font_size_label.pack_forget()
 #font_size_entry.pack_forget()
 #adjust_button.pack_forget()
 char_note.pack_forget()
 char_note_label.pack_forget()

 root.destroy()
 switch = 1
 setup_mode(switch)

 # Create a menubar
 menubar = tk.Menu(root)

 # Create an options menu and add it to the menubar
 options_menu = tk.Menu(menubar, tearoff=0)
 #menubar.add_cascade(label="Options", menu=options_menu)

 # Create a dropdown menu and add it to the menubar
 dropdown = tk.Menu(menubar, tearoff=0)
 menubar.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"

 # Add commands to dropdown menu

 # Add checkbutton to options menu
 dropdown.add_command(label="Toggle Manual/Random Colour", command=manual_colour)

 dropdown.add_command(label="Refresh", command=refresh_canvas)
```

```
dropdown.add_command(label="Undo", command=undo_last)
dropdown.add_command(label="Save Session", command=save_session)
dropdown.add_command(label="Load Session", command=load_session)
dropdown.add_command(label="Save as Image", command=save_canvas)
dropdown.add_command(label="Compile Canvas", command=compileGOSmX)
Add the new command for resizing images
dropdown.add_command(label="Select and Resize Image", command=select_and_resize_image)
Add the command for selecting images
dropdown.add_command(label="Select Images", command=select_and_add_images)
Add the toggle mode command
dropdown.add_command(label="Toggle Draw Chars/Images", command=toggle_mode)
#dropdown.add_command(label="Exit", command=exit_app)
#dropdown.add_command(label="Toggle Type/Click Mode", command=toggle_typing_mode)
dropdown.add_command(label="About Program", command=show_about)
dropdown.add_command(label="Go Home", command=home)
dropdown.add_command(label="Edit Grid Dimensions", command=adjust_grid_and_font)
dropdown.add_command(label="Typing Mode", command=show_typing_mode_menu)

Set the menubar
root.config(menu=menubar)

def create_menu_1():
 global control_frame
 def home1():
 canvas.pack_forget()
 menubar1.destroy()
 root.destroy()
 switch = 1
 setup_mode(switch)

 # Create a menubar
 menubar1 = tk.Menu(root)

 # Create an options menu and add it to the menubar
 options_menu1 = tk.Menu(menubar1, tearoff=0)
 #menubar.add_cascade(label="Options", menu=options_menu)

 # Create a dropdown menu and add it to the menubar
 dropdown = tk.Menu(menubar1, tearoff=0)
 menubar1.add_cascade(label="Options", menu=dropdown) # Use a different label, e.g., "Dropdown"

 # Add commands to dropdown menu

 # Add checkbutton to options menu
 dropdown.add_command(label="About Program", command=show_about)
 dropdown.add_command(label="Change Mode", command=home1)
```

```
Set the menubar
root.config(menu=menubar1)

def manual_colour():
 # Toggle the value of color_mode_var
 current_value = color_mode_var.get()
 color_mode_var.set(not current_value)

"""

Create a tkinter window
root = tk.Tk()
root.title("General Operating System")
root.iconbitmap("logo.ico")
"""

def show_typing_mode_menu():
 global color_
 while True:
 print("1. Use default UTF-8 mapping")
 print("4. Exit")
 choice = input("Choose an option: ")

 if choice == "1":
 # Use default UTF-8 mapping
 print("Using default UTF-8 mapping.")
 color_ = input("Use Default Colour Scheme (1): ")
 if(color_ == '1'):
 # Initialize bg_color and font_color with default values at the global scope
 bg_color = "#FFFFFF" # Default white background color
 font_color = "#000000" # Default black font color
 break
 else:
 print("Invalid choice. Please try again.")

 canvas.bind("<Key>", type_character)
 canvas.focus_set()

def set_background_color():
 global color_
 #bg_color = tk.colorchooser.askcolor()[1]
 color = input("Enter background color (#FFFFFF for example):")

def set_font_color():
```

```

global font_color, font_size
#font_color = tk.colorchooser.askcolor()[1]
font_color = input("Enter background color (#000000 for example): ")
font_size = input("Enter font_size: ")

def type_character(event):
 global current_row, current_col, col, font_size, font_color, color, canvas_width, square_size
 color = "#FFFFFF"
 font_color = "#000000"
 char = event.char
 if char: # ignore non-character events
 try:
 draw_char1(canvas, char, current_row, current_col, font_size, color)
 current_col += 1
 if (current_col >= canvas_width/square_size):
 current_col = 0
 current_row += 1
 if(current_row >= canvas_width/square_size):
 current_col = 0
 current_row = 0
 except Exception as e:
 print("Something Went Wrong...")
 print(e)

def setup_mode(sw):

 global mode_entry, mode_label, square_size, grid_size, char_count, last_drawn
 global root, color_mode_var, logging, paused, canvas_width, canvas_height, canvas
 global grid_size_label, grid_size_entry, font_size_label, font_size_entry, adjust_button
 global save_Sbutton, load_button, char_label, char_entry, char_note_label, char_note, char_button
 global refresh_button, undo_button, control_frame, submit_button, col, color_
 global image_selection_frame # Use the global variable

 current_row = 0
 current_col = 0
 color_ = 1

 # Define the size of the squares and the grid
 #square_size = 70
 square_size = int(input("Enter sub-square size: "))
 #grid_size = 10
 grid_size = int(input("Enter grid size by the number of sub-squares: "))

```

```
char_count = 0
last_drawn = []

window_height = square_size * grid_size + (square_size * 2)
window_width = square_size * grid_size

if(sw == 1):
 # Create a tkinter window
 root = tk.Tk()
 # Define the frame for displaying the image selection
 image_selection_frame = tk.Frame(root)
 image_selection_frame.pack(side='top') # Adjust the side as per your layout
 root.title("C")
 #root.iconbitmap("logo.ico")
 sw = 0
 switch = 0

###

Get screen width and height
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()

Calculate position of the window
position_top = int(screen_height / 2 - window_height / 2)
position_right = int(screen_width / 2 - window_width / 2)

Set the dimensions of the window and where it is placed
root.geometry(f'{window_width}x{window_height}+{position_right}+{position_top}')

###

Create a BooleanVar for the color mode and set it to False initially
color_mode_var = BooleanVar(value=False)

Setup logging
logging.basicConfig(filename='logging.txt', level=logging.INFO)

paused = tk.BooleanVar()
paused.set(False)

canvas_width = grid_size * square_size
canvas_height = grid_size * square_size
canvas = tk.Canvas(root, width=canvas_width, height=canvas_height)
col = canvas_width
canvas.pack()
canvas.bind("<Button-1>", on_canvas_click)
```

```
In the function where you create the canvas...
canvas.bind("<Key>", on_key_press)
canvas.focus_set()

Create input fields and buttons for grid and font size input
grid_size_label = tk.Label(root, text="Grid size:")
grid_size_entry = tk.Entry(root, width=5)
font_size_label = tk.Label(root, text="Font size:")
font_size_entry = tk.Entry(root, width=5)
adjust_button = tk.Button(root, text='Adjust Grid & Font', command=adjust_grid_and_font)

Create the save and load buttons
save_Sbutton = tk.Button(root, text='Save Session', command=save_session, bg="white", padx=5, pady=0)
load_button = tk.Button(root, text='Load Session', command=load_session, bg="white", padx=5, pady=0)

Create input fields and buttons for mode and character input
mode_label = tk.Label(root, text="Enter ... 'IDE' for an\nInteractive Development Environment\nfor Learning anything!")
mode_entry = tk.Entry(root)
submit_button = tk.Button(root, text='LEARN', command=submit_mode)

"""

filename = simpledialog.askstring("Save Canvas", "Enter filename:")
if filename:
"""

char_label = tk.Label(root, text="Enter symbol/s to draw: ")
char_entry = tk.Entry(root, width=12)
char_note_label = tk.Label(root, text="Enter Note: ")
char_note = tk.Entry(root, width=12)
#char_button = tk.Button(root, text='Draw Character', command=draw_char)

refresh_button = tk.Button(root, text='Refresh', command=refresh_canvas, bg="white", padx=5, pady=0)
undo_button = tk.Button(root, text='Undo', command=undo_last, bg="white", padx=5, pady=0)

control_frame = tk.Frame(root)

mode_label.pack()
mode_entry.pack()
submit_button.pack()
create_menu(root)

Run the tkinter main loop
root.mainloop()

global switch
switch = 1
```

```
setup_mode(switch)

----- Content of Major_Language_Model.py-----
import tkinter as tk
from tkinter import ttk

def create_table(op_type, n):
 for widget in table_frame.winfo_children():
 widget.destroy()

 counter = 0

 # Open logic.txt in write mode to save new logic statements
 with open('logicBuffer.txt', 'w') as logic_file:
 for i in range(1, n+1): # start range from 1 to avoid zero division
 for j in range(1, n+1):
 # Generate a logic statement based on the operation type
 if op_type == "Addition":
 statement = f"{{i}} + {{j}} % {n} == {{i + j % n}}"
 res = i + j % n
 elif op_type == "Multiplication":
 statement = f"{{i}} * {{j}} % {n} == {{i * j % n}}"
 res = i * j % n
 binary_id = format(counter, '012b')
 counter += 1
 # Assume all statements are true and assign a binary ID (adjust as needed)
 logic_file.write(f"{statement} 1 {binary_id}\n")

 # Create labels for the GUI table
 label = tk.Label(table_frame, text=statement, width=20, height=2, borderwidth=1, relief="solid")
 label.grid(row=i, column=j)
 print("Output in logicBuffer.txt")

root = tk.Tk()
root.title("Modular Arithmetic Table")

control_frame = tk.Frame(root)
control_frame.pack(side=tk.TOP, padx=10, pady=10)

table_frame = tk.Frame(root)
table_frame.pack(side=tk.BOTTOM, padx=10, pady=10)

operation_label = tk.Label(control_frame, text="Operation:")
operation_label.pack(side=tk.LEFT)

operation_combobox = ttk.Combobox(control_frame, values=["Addition", "Multiplication"],
state="readonly")
operation_combobox.pack(side=tk.LEFT)
```

```

operation_combobox.current(0)

n_label = tk.Label(control_frame, text="Modulo (n):")
n_label.pack(side=tk.LEFT)

n_entry = tk.Entry(control_frame, width=5)
n_entry.pack(side=tk.LEFT)
n_entry.insert(0, "6")

def update_table():
 op_type = operation_combobox.get()
 n = int(n_entry.get())
 create_table(op_type, n)

update_button = tk.Button(control_frame, text="Update Table", command=update_table)
update_button.pack(side=tk.LEFT)

create_table("Addition", 6)

root.mainloop()

```

----- Content of MealyMachine-n-state\_3.py-----

```

#Written with Chat GPT-4 (OpenAI)
class MealyMachine:
 def __init__(self, sequence):
 self.state = 0
 self.sequence = sequence

 def transition(self, input_symbol):
 if input_symbol == self.sequence[self.state]:
 self.state += 1
 if self.state == len(self.sequence):
 self.state = 1 if input_symbol == self.sequence[0] else 0
 return 1
 else:
 return 0
 else:
 self.state = 1 if input_symbol == self.sequence[0] else 0
 return 0

 def main():
 sequence = input("Enter the binary sequence to be detected: ")
 while not all(c in '01' for c in sequence):
 sequence = input("Invalid input. Enter a valid binary sequence: ")

 mealy_machine = MealyMachine(sequence)
 input_stream = input("Enter the input stream: ")

 print(f"Input Stream: {input_stream}")

```

```
print("Output Stream: ", end="")

for symbol in input_stream:
 output = mealy_machine.transition(symbol)
 print(output, end="")

print()

if __name__ == "__main__":
 main()
```

----- Content of Mechanical Computer.py-----

```
import FreeCAD, Part

Create a new document
doc = FreeCAD.newDocument("Stainless_Steel_Slab")

Create the stainless steel slab
slab = Part.makeBox(135, 63, 6)
slab_obj = doc.addObject("Part::Feature", "Slab")
slab_obj.Shape = slab

Create the first slot (one-third down from the top, starting from the top face)
slot1 = Part.makeBox(5, 63, 3, FreeCAD.Vector((135-5)/3, 0, 6- 3))

Create the second slot (two-thirds down from the top, starting from the top face)
slot2 = Part.makeBox(5, 63, 3, FreeCAD.Vector(2*(135-5)/3, 0, 6- 3))

Cut the slots from the slab
cut1 = slab.cut(slot1)
cut2 = cut1.cut(slot2)
slab_obj.Shape = cut2

Recompute the document
doc.recompute()
```

----- Content of MechanicalComputer.py-----

```
import FreeCAD, Part

Create a new document
doc = FreeCAD.newDocument("Stainless_Steel_Slab")

Create the stainless steel slab
slab = Part.makeBox(135, 63, 6)
slab_obj = doc.addObject("Part::Feature", "Slab")
slab_obj.Shape = slab
```

```

Create the first slot (one-third down from the top, starting from the top face)
slot1 = Part.makeBox(5, 63, 3, FreeCAD.Vector((135-5)/3, 0, 6- 3))

Create the second slot (two-thirds down from the top, starting from the top face)
slot2 = Part.makeBox(5, 63, 3, FreeCAD.Vector(2*(135-5)/3, 0, 6- 3))

Cut the slots from the slab
cut1 = slab.cut(slot1)
cut2 = cut1.cut(slot2)
slab_obj.Shape = cut2

Recompute the document
doc.recompute()

```

----- Content of Music-1.py-----

```

from midiutil import MIDIFile

def generate_ambient_percussive_music(file_path, filter_set={2, 3, 5}, mod_values=[2, 3, 5]):
 with open(file_path, 'r') as file:
 content = file.read().split()

 filtered_content = [float(num) for num in content if any((int(digit) in filter_set) for digit in str(num)) if
digit.isdigit()])

 # Initialize MIDI file with two tracks
 midi = MIDIFile(2)

 # Track for ambient sounds
 ambient_track = 0
 midi.addTrackName(ambient_track, 0, "Ambient Track")
 midi.addTempo(ambient_track, 0, 60) # Slower tempo for ambient part

 # Track for percussive sounds
 percussive_track = 1
 midi.addTrackName(percussive_track, 0, "Percussive Track")
 midi.addTempo(percussive_track, 0, 120) # Faster tempo for rhythm

 # Program change for ambient and percussive instruments
 ambient_instrument = 91 # Pad, for example
 percussive_instrument = 114 # Steel Drums, for example
 midi.addProgramChange(ambient_track, 0, 0, ambient_instrument)
 midi.addProgramChange(percussive_track, 0, 0, percussive_instrument)

 time = 0
 for i, value in enumerate(filtered_content):
 for mod_value in mod_values:
 pitch = (int(value * i) + int(value)) % mod_value + 60

```

```

if mod_value in [2, 5]: # Example condition for ambient sounds
 duration = 4 # Longer duration for ambient notes
 volume = 100
 midi.addNote(ambient_track, 0, pitch, time, duration, volume)
else: # For percussive sounds
 duration = 1 # Shorter, percussive duration
 volume = 100
 midi.addNote(percussive_track, 9, pitch, time, duration, volume) # Channel 9 is typically used for
percussion

time += 1

Output MIDI file
output_file = file_path.replace('.txt', '_ambient_percussive.mid')
with open(output_file, 'wb') as outf:
 midi.writeFile(outf)

print(f"Generated music with ambient and percussive elements saved to {output_file}")

Example usage
file_path = input("Enter the path to the input file: ")
generate_ambient_percussive_music(file_path)

```

----- Content of Music-2.py-----

```

from midiutil import MIDIFile

def generate_music(file_path, filter_set, mod_values, ambient_instrument, percussive_instrument,
ambient_tempo, percussive_tempo):
 # Simplified version of the generation function for brevity
 print(f"Generating music with settings:\nFilter Set: {filter_set}\nModulo Values: {mod_values}\nAmbient
Instrument: {ambient_instrument}\nPercussive Instrument: {percussive_instrument}\nAmbient Tempo:
{ambient_tempo}\nPercussive Tempo: {percussive_tempo}")
 # MIDI generation logic goes here
 # For demonstration, just print the settings
 print("Music generated successfully!")

def main_menu():
 print("\nMIDI Music Generation Workbench")
 print("1. Set Filter Set")
 print("2. Set Modulo Values")
 print("3. Set Ambient Instrument")
 print("4. Set Percussive Instrument")
 print("5. Set Ambient Tempo")
 print("6. Set Percussive Tempo")
 print("7. Generate MIDI File")
 print("0. Exit")

Default settings
settings = {

```

```

'filter_set': {2, 3, 5},
'mod_values': [2, 3, 5],
'ambient_instrument': 91, # Pad
'percussive_instrument': 114, # Steel Drums
'ambient_tempo': 60,
'percussive_tempo': 120,
}

def update_setting(choice):
 if choice == 1:
 new_set = input("Enter new filter set (e.g., 2,3,5): ")
 settings['filter_set'] = set(map(int, new_set.split(',')))
 elif choice == 2:
 new_values = input("Enter new modulo values (e.g., 2,3,5): ")
 settings['mod_values'] = list(map(int, new_values.split(',')))
 # Continue for other settings...

while True:
 main_menu()
 choice = int(input("Enter your choice: "))

 if choice == 0:
 print("Exiting program...")
 break
 elif 1 <= choice <= 6:
 update_setting(choice)
 elif choice == 7:
 file_path = input("Enter the path to the input file: ")
 generate_music(file_path, **settings)
 else:
 print("Invalid choice. Please enter a number between 0 and 7.")

```

----- Content of Music-3.py-----

```

from midiutil import MIDIFile
import math

Initialize track configuration
tracks_config = {}
Default settings
settings = {
 'filter_set': {2, 3, 5},
}
gm1_instrument_names = [
 "Acoustic Grand Piano", "Bright Acoustic Piano", "Electric Grand Piano", "Honky-tonk Piano",
 "Electric Piano 1", "Electric Piano 2", "Harpsichord", "Clavinet",
 "Celesta", "Glockenspiel", "Music Box", "Vibraphone",
 "Marimba", "Xylophone", "Tubular Bells", "Dulcimer",
 "Drawbar Organ", "Percussive Organ", "Rock Organ", "Church Organ",
 "Reed Organ", "Accordion", "Harmonica", "Tango Accordion",
]

```

```
"Acoustic Guitar (nylon)", "Acoustic Guitar (steel)", "Electric Guitar (jazz)", "Electric Guitar (clean)",
"Electric Guitar (muted)", "Overdriven Guitar", "Distortion Guitar", "Guitar harmonics",
"Acoustic Bass", "Electric Bass (finger)", "Electric Bass (pick)", "Fretless Bass",
"Slap Bass 1", "Slap Bass 2", "Synth Bass 1", "Synth Bass 2",
"Violin", "Viola", "Cello", "Contrabass",
"Tremolo Strings", "Pizzicato Strings", "Orchestral Harp", "Timpani",
"String Ensemble 1", "String Ensemble 2", "Synth Strings 1", "Synth Strings 2",
"Choir Aahs", "Voice Ooohs", "Synth Choir", "Orchestra Hit",
"Trumpet", "Trombone", "Tuba", "Muted Trumpet",
"French Horn", "Brass Section", "Synth Brass 1", "Synth Brass 2",
"Soprano Sax", "Alto Sax", "Tenor Sax", "Baritone Sax",
"Oboe", "English Horn", "Bassoon", "Clarinet",
"Piccolo", "Flute", "Recorder", "Pan Flute",
"Blown Bottle", "Shakuhachi", "Whistle", "Ocarina",
"Lead 1 (square)", "Lead 2 (sawtooth)", "Lead 3 (calliope)", "Lead 4 (chiff)",
"Lead 5 (charang)", "Lead 6 (voice)", "Lead 7 (fifths)", "Lead 8 (bass + lead)",
"Pad 1 (new age)", "Pad 2 (warm)", "Pad 3 (polysynth)", "Pad 4 (choir)",
"Pad 5 (bowed)", "Pad 6 (metallic)", "Pad 7 (halo)", "Pad 8 (sweep)",
"FX 1 (rain)", "FX 2 (soundtrack)", "FX 3 (crystal)", "FX 4 (atmosphere)",
"FX 5 (brightness)", "FX 6 (goblins)", "FX 7 (echoes)", "FX 8 (sci-fi)",
"Sitar", "Banjo", "Shamisen", "Koto",
"Kalimba", "Bagpipe", "Fiddle", "Shanai",
"Tinkle Bell", "Agogo", "Steel Drums", "Woodblock",
"Taiko Drum", "Melodic Tom", "Synth Drum", "Reverse Cymbal",
"Guitar Fret Noise", "Breath Noise", "Seashore", "Bird Tweet",
"Telephone Ring", "Helicopter", "Applause", "Gunshot"
```

```
]
```

```
Automatically generate the dictionary of instruments mapped to their corresponding integers
midi_instruments = {name: index for index, name in enumerate(gm1_instrument_names)}
```

```
def get_user_mapping_preferences():
 print("Define how to map numbers to note attributes.")
 pitch_rule = input("Pitch mapping (e.g., 'decimal * 127'): ")
 time_rule = input("Start time mapping (e.g., 'index' for sequential): ")
 duration_rule = input("Duration mapping (e.g., 'fixed', '1'): ")
 volume_rule = input("Volume mapping (e.g., 'fixed', '100'): ")

 return {
 'pitch_rule': pitch_rule,
 'time_rule': time_rule,
 'duration_rule': duration_rule,
 'volume_rule': volume_rule,
 }
```

```
#import math Ensure this import is at the top of your script
```

```
def interpret_and_load_notes(file_path, track_name, preferences):
 if track_name not in tracks_config:
 print(f"Track '{track_name}' not found. Adding it automatically.")
```

```

tracks_config[track_name] = {'instrument': 0, 'tempo': 120, 'pan': 64, 'notes': []}

with open(file_path, 'r') as file:
 numbers = file.read().split()

for index, number in enumerate(numbers):
 # Update the eval environment to include the math module
 eval_env = {
 'index': index,
 'number': float(number),
 'decimal': float(f"0.{str(number).split('.')[1]}"),
 'math': math # Include math module here
 }

 pitch = eval(preferences['pitch_rule'], eval_env)
 time = eval(preferences['time_rule'], eval_env)
 duration = eval(preferences['duration_rule'], eval_env)
 volume = eval(preferences['volume_rule'], eval_env)

 tracks_config[track_name]['notes'].append((int(pitch), time, duration, int(volume)))

Function to display the instruments and their IDs
def list_available_instruments():
 print("Available MIDI Instruments:")
 for name, id in sorted(midi_instruments.items(), key=lambda item: item[1]):
 print(f"{id:3}: {name}")

def add_track():
 track_name = input("Enter new track name: ")
 if track_name in tracks_config:
 print("Track already exists.")
 return
 tracks_config[track_name] = {
 'instrument': 0, # Default instrument
 'tempo': 120, # Default tempo
 'pan': 64, # Center panning
 'notes': [] # List of tuples (pitch, time, duration, volume)
 }
 print(f"Track '{track_name}' added.")

def set_instrument():
 track_name = input("Enter track name: ")
 # Example usage
 list_available_instruments()
 instrument = int(input("Enter instrument number: "))
 if track_name in tracks_config:
 tracks_config[track_name]['instrument'] = instrument
 print(f"Instrument for '{track_name}' set to {instrument}.")
 else:

```

```

print("Track not found.")

def set_track_pan():
 track_name = input("Enter track name: ")
 pan = int(input("Enter pan position (0-127): "))
 if track_name in tracks_config and 0 <= pan <= 127:
 tracks_config[track_name]['pan'] = pan
 print(f"Panning for '{track_name}' set to {pan}.")
 else:
 print("Invalid track name or pan position.")

def add_note_to_track():
 track_name = input("Enter track name: ")
 pitch = int(input("Enter pitch (0-127): "))
 # Apply the language filter to the pitch
 if pitch % 10 not in settings['filter_set']:
 print(f"Pitch {pitch} does not pass the filter criteria.")
 return
 time = float(input("Enter time: "))
 duration = float(input("Enter duration: "))
 volume = int(input("Enter volume (0-127): "))
 if track_name in tracks_config:
 tracks_config[track_name]['notes'].append((pitch, time, duration, volume))
 print(f"Note added to '{track_name}'.")
 else:
 print("Track not found.")

def shift_notes_in_track():
 track_name = input("Enter track name: ")
 shift_amount = float(input("Enter shift amount (negative to shift left, positive to shift right): "))
 if track_name in tracks_config:
 shifted_notes = [(pitch, time + shift_amount, duration, volume) for pitch, time, duration, volume in
tracks_config[track_name]['notes']]
 tracks_config[track_name]['notes'] = shifted_notes
 print(f"Notes in '{track_name}' shifted by {shift_amount}.")
 else:
 print("Track not found.")

def generate_music(file_path):
 midi = MIDIFile(len(tracks_config)) # Number of tracks

 track_index = 0
 for track_name, config in tracks_config.items():
 midi.addTrackName(track_index, 0, track_name)
 midi.addTempo(track_index, 0, config['tempo'])
 midi.addProgramChange(track_index, track_index, 0, config['instrument'])
 midi.addControllerEvent(track_index, track_index, 0, 10, config['pan']) # Apply panning

 for note in config['notes']:

```

```
pitch, time, duration, volume = note
midi.addNote(track_index, track_index, time, duration, pitch, volume)

track_index += 1

output_filename = file_path.replace('.txt', '_generated.mid')
with open(output_filename, 'wb') as outf:
 midi.writeFile(outf)

print(f"Music generated successfully and saved to {output_filename}")

def update_filter_set():
 new_set = input("Enter new filter set (e.g., 2,3,5): ")
 settings['filter_set'] = set(map(int, new_set.split(',')))
 print(f"Filter set updated to {settings['filter_set']}.")

def main_menu():
 print("\nMIDI Music Generation Workbench")
 print("1. Add Track")
 print("2. Set Instrument for Track")
 print("3. Set Track Pan")
 print("4. Add Note to Track")
 print("5. Shift Notes in Track")
 print("6. Update Language Filter Set")
 print("7. Generate MIDI File")
 print("8. Add note sequence to a Track")
 print("0. Exit")

while True:
 main_menu()
 choice = int(input("Enter your choice: "))

 if choice == 0:
 print("Exiting program...")
 break
 elif choice == 1:
 add_track()
 elif choice == 2:
 set_instrument()
 elif choice == 3:
 set_track_pan()
 elif choice == 4:
 add_note_to_track()
 elif choice == 5:
 shift_notes_in_track()
 elif choice == 6:
 update_filter_set()
 elif choice == 7:
 file_path = input("Enter the path to the input file (used for naming the MIDI file): ")
```

```

 generate_music(file_path)
elif choice == 8:
 file_path = input("Enter the path to the note sequence file: ")
 track_name = input("Enter the track name to add notes to: ")
 preferences = get_user_mapping_preferences()
 interpret_and_load_notes(file_path, track_name, preferences)
else:
 print("Invalid choice. Please enter a number between 0 and 7.")

```

----- Content of Music-4.py-----

```

from midiutil import MIDIFile
import math

Initialize track configuration
tracks_config = {}
Default settings
settings = {
 'filter_set': {2, 3, 5},
}
gm1_instrument_names = [
 "Acoustic Grand Piano", "Bright Acoustic Piano", "Electric Grand Piano", "Honky-tonk Piano",
 "Electric Piano 1", "Electric Piano 2", "Harpsichord", "Clavinet",
 "Celesta", "Glockenspiel", "Music Box", "Vibraphone",
 "Marimba", "Xylophone", "Tubular Bells", "Dulcimer",
 "Drawbar Organ", "Percussive Organ", "Rock Organ", "Church Organ",
 "Reed Organ", "Accordion", "Harmonica", "Tango Accordion",
 "Acoustic Guitar (nylon)", "Acoustic Guitar (steel)", "Electric Guitar (jazz)", "Electric Guitar (clean)",
 "Electric Guitar (muted)", "Overdriven Guitar", "Distortion Guitar", "Guitar harmonics",
 "Acoustic Bass", "Electric Bass (finger)", "Electric Bass (pick)", "Fretless Bass",
 "Slap Bass 1", "Slap Bass 2", "Synth Bass 1", "Synth Bass 2",
 "Violin", "Viola", "Cello", "Contrabass",
 "Tremolo Strings", "Pizzicato Strings", "Orchestral Harp", "Timpani",
 "String Ensemble 1", "String Ensemble 2", "Synth Strings 1", "Synth Strings 2",
 "Choir Ahhs", "Voice Ooohs", "Synth Choir", "Orchestra Hit",
 "Trumpet", "Trombone", "Tuba", "Muted Trumpet",
 "French Horn", "Brass Section", "Synth Brass 1", "Synth Brass 2",
 "Soprano Sax", "Alto Sax", "Tenor Sax", "Baritone Sax",
 "Oboe", "English Horn", "Bassoon", "Clarinet",
 "Piccolo", "Flute", "Recorder", "Pan Flute",
 "Blown Bottle", "Shakuhachi", "Whistle", "Ocarina",
 "Lead 1 (square)", "Lead 2 (sawtooth)", "Lead 3 (calliope)", "Lead 4 (chiff)",
 "Lead 5 (charang)", "Lead 6 (voice)", "Lead 7 (fifths)", "Lead 8 (bass + lead)",
 "Pad 1 (new age)", "Pad 2 (warm)", "Pad 3 (polysynth)", "Pad 4 (choir)",
 "Pad 5 (bowed)", "Pad 6 (metallic)", "Pad 7 (halo)", "Pad 8 (sweep)",
 "FX 1 (rain)", "FX 2 (soundtrack)", "FX 3 (crystal)", "FX 4 (atmosphere)",
 "FX 5 (brightness)", "FX 6 (goblins)", "FX 7 (echoes)", "FX 8 (sci-fi)",
 "Sitar", "Banjo", "Shamisen", "Koto",
 "Kalimba", "Bagpipe", "Fiddle", "Shanai",
]

```

```

 "Tinkle Bell", "Agogo", "Steel Drums", "Woodblock",
 "Taiko Drum", "Melodic Tom", "Synth Drum", "Reverse Cymbal",
 "Guitar Fret Noise", "Breath Noise", "Seashore", "Bird Tweet",
 "Telephone Ring", "Helicopter", "Applause", "Gunshot"
]
MIDI instruments mapping remains unchanged
midi_instruments = {name: index for index, name in enumerate(gm1_instrument_names)}

def get_user_mapping_preferences():
 # Enhanced to include start time customization
 print("Define how to map numbers to note attributes.")
 pitch_rule = input("Pitch mapping (e.g., 'decimal * 127'): ")
 time_rule = input("Start time mapping (e.g., 'index * 0.5' for staggered): ")
 duration_rule = input("Duration mapping (e.g., 'fixed', '1'): ")
 volume_rule = input("Volume mapping (e.g., 'fixed', '100'): ")

 return {
 'pitch_rule': pitch_rule,
 'time_rule': time_rule,
 'duration_rule': duration_rule,
 'volume_rule': volume_rule,
 }

#import math Ensure this import is at the top of your script

def interpret_and_load_notes(file_path, track_name, preferences):
 if track_name not in tracks_config:
 print(f"Track '{track_name}' not found. Adding it automatically.")
 tracks_config[track_name] = {'instrument': 0, 'tempo': 120, 'pan': 64, 'notes': []}

 with open(file_path, 'r') as file:
 numbers = file.read().split()

 for index, number in enumerate(numbers):
 # Update the eval environment to include the math module
 eval_env = {
 'index': index,
 'number': float(number),
 'decimal': float(f"0.{str(number).split('.')[1]}"),
 'math': math # Include math module here
 }

 pitch = eval(preferences['pitch_rule'], eval_env)
 time = eval(preferences['time_rule'], eval_env)
 duration = eval(preferences['duration_rule'], eval_env)
 volume = eval(preferences['volume_rule'], eval_env)

 tracks_config[track_name]['notes'].append((int(pitch), time, duration, int(volume)))

```

```

Function to display the instruments and their IDs
def list_available_instruments():
 print("Available MIDI Instruments:")
 for name, id in sorted(midi_instruments.items(), key=lambda item: item[1]):
 print(f"{id:3}: {name}")

def add_track(track_name):
 # Prompt the user for each setting
 print(f"Adding new track: {track_name}")

 # List available instruments for user selection
 list_available_instruments()
 instrument = int(input("Enter instrument number: "))

 tempo = int(input("Enter tempo (BPM): "))
 pan = int(input("Enter pan position (0-127, where 64 is center): "))

 # Initialize the new track with user-defined settings
 tracks_config[track_name] = {
 'instrument': instrument,
 'tempo': tempo,
 'pan': pan,
 'notes': [] # Initialize an empty list for notes
 }
 print(f"Track '{track_name}' added with instrument {instrument}, tempo {tempo}, and pan {pan}.")

def set_instrument():
 track_name = input("Enter track name: ")
 # Example usage
 list_available_instruments()
 instrument = int(input("Enter instrument number: "))
 if track_name in tracks_config:
 tracks_config[track_name]['instrument'] = instrument
 print(f"Instrument for '{track_name}' set to {instrument}.")
 else:
 print("Track not found.")

def set_track_pan():
 track_name = input("Enter track name: ")
 pan = int(input("Enter pan position (0-127): "))
 if track_name in tracks_config and 0 <= pan <= 127:
 tracks_config[track_name]['pan'] = pan
 print(f"Panning for '{track_name}' set to {pan}.")
 else:
 print("Invalid track name or pan position.")

def add_note_to_track():
 track_name = input("Enter track name: ")
 pitch = int(input("Enter pitch (0-127): "))

```

```

Apply the language filter to the pitch
if pitch % 10 not in settings['filter_set']:
 print(f"Pitch {pitch} does not pass the filter criteria.")
 return
time = float(input("Enter time: "))
duration = float(input("Enter duration: "))
volume = int(input("Enter volume (0-127): "))
if track_name in tracks_config:
 tracks_config[track_name]['notes'].append((pitch, time, duration, volume))
 print(f"Note added to '{track_name}'")
else:
 print("Track not found.")

def shift_notes_in_track():
 track_name = input("Enter track name: ")
 shift_amount = float(input("Enter shift amount (negative to shift left, positive to shift right): "))
 if track_name in tracks_config:
 shifted_notes = [(pitch, time + shift_amount, duration, volume) for pitch, time, duration, volume in
tracks_config[track_name]['notes']]
 tracks_config[track_name]['notes'] = shifted_notes
 print(f"Notes in '{track_name}' shifted by {shift_amount}.")
 else:
 print("Track not found.")

def generate_music(file_path):
 midi = MIDIFile(len(tracks_config)) # Number of tracks

 track_index = 0
 for track_name, config in tracks_config.items():
 midi.addTrackName(track_index, 0, track_name)
 midi.addTempo(track_index, 0, config['tempo'])
 midi.addProgramChange(track_index, track_index, 0, config['instrument'])
 midi.addControllerEvent(track_index, track_index, 0, 10, config['pan']) # Apply panning

 for note in config['notes']:
 pitch, start_time, duration, volume = note
 # Ensure that pitch and volume are integers
 pitch = int(pitch)
 volume = int(volume)
 midi.addNote(track_index, track_index, start_time, duration, pitch, volume)

 track_index += 1

 output_filename = file_path.replace('.txt', '_generated.mid')
 with open(output_filename, 'wb') as outf:
 midi.writeFile(outf)

 print(f"Music generated successfully and saved to {output_filename}")

```

```

def interpret_and_distribute_notes(file_path, track_names, preferences):
 with open(file_path, 'r') as file:
 numbers = file.read().split()

 note_count = len(numbers)
 notes_per_track = max(note_count // len(track_names), 1) # Ensure at least one note per track

 for index, number in enumerate(numbers):
 track_index = index // notes_per_track % len(track_names)
 track_name = track_names[track_index]

 # Assuming the environment setup for eval includes 'index', 'number', 'math'
 eval_env = {'index': index, 'number': float(number), 'decimal': float(f"0.{str(number).split('.')[1]}"),
 'math': math}
 pitch = eval(preferences['pitch_rule'], eval_env)
 time = eval(preferences['time_rule'], eval_env)
 duration = eval(preferences['duration_rule'], eval_env)
 volume = eval(preferences['volume_rule'], eval_env)

 # Add note to the appropriate track
 tracks_config[track_name]['notes'].append((int(pitch), time, duration, int(volume)))

def update_filter_set():
 new_set = input("Enter new filter set (e.g., 2,3,5): ")
 settings['filter_set'] = set(map(int, new_set.split(',')))
 print(f"Filter set updated to {settings['filter_set']}.")

def main_menu():
 print("\nMIDI Music Generation Workbench")
 print("1. Add Track")
 print("2. Set Instrument for Track")
 print("3. Set Track Pan")
 print("4. Add Note to Track")
 print("5. Shift Notes in Track")
 print("6. Update Language Filter Set")
 print("7. Generate MIDI File")
 print("8. Add note sequence to a Track")
 print("9. Help and Documentation") # New help command
 print("0. Exit")

def display_help():
 print("Help and Documentation:")
 print("- 1 Add Track: Adds a new track. Specify instrument, tempo, and pan.")
 print("- 2 Set Instrument: Change the instrument for a track.")
 print("- 3 Set Track Pan: Adjusts stereo pan for a track.")
 print("- 4 Add Note: Add a note with pitch, time, duration, and volume.")
 print("- 5 Shift Notes: Moves all notes in a track in time.")

```

```

print("- 6 Update Filter Set: Change pitch filter criteria.")
print("- 7 Generate MIDI: Creates a MIDI file from your configurations.")
print("- 8 Add Sequence: Add a sequence of notes to tracks, with timing and distribution options.")
print("- 9 Help: Displays this help information.")
print("Each function will guide you with prompts for necessary inputs.")

while True:
 main_menu()
 choice = int(input("Enter your choice: "))

 if choice == 0:
 print("Exiting program...")
 break
 elif choice == 1:
 tn = input("Enter track name: ")
 add_track(tn)
 elif choice == 2:
 set_instrument()
 elif choice == 3:
 set_track_pan()
 elif choice == 4:
 add_note_to_track()
 elif choice == 5:
 shift_notes_in_track()
 elif choice == 6:
 update_filter_set()
 elif choice == 7:
 file_path = input("Enter the path to the input file (used for naming the MIDI file): ")
 generate_music(file_path)
 elif choice == 8:
 file_path = input("Enter the path to the note sequence file: ")
 distribute_notes = int(input("Distribute notes across multiple tracks automatically? Enter 1 for Yes, 0 for No: "))
 if distribute_notes:
 num_tracks = int(input("Enter the number of tracks to distribute notes across: "))
 track_names = [f"Track_{i+1}" for i in range(num_tracks)]
 for name in track_names:
 if name not in tracks_config:
 add_track(name)
 preferences = get_user_mapping_preferences()
 interpret_and_distribute_notes(file_path, track_names, preferences)
 else:
 track_name = input("Enter the track name to add notes to: ")
 preferences = get_user_mapping_preferences()
 interpret_and_load_notes(file_path, track_name, preferences)
 elif choice == 9:
 display_help()
 else:
 print("Invalid choice. Please enter a number between 0 and 9.")

```

----- Content of Music-5.py-----

```
from midiutil import MIDIFile
import math
import random

Initialize track configuration
tracks_config = {}

Default settings
settings = {
 'filter_set': {2, 3, 5},
}

gm1_instrument_names = [
 "Acoustic Grand Piano", "Bright Acoustic Piano", "Electric Grand Piano", "Honky-tonk Piano",
 "Electric Piano 1", "Electric Piano 2", "Harpsichord", "Clavinet",
 "Celesta", "Glockenspiel", "Music Box", "Vibraphone",
 "Marimba", "Xylophone", "Tubular Bells", "Dulcimer",
 "Drawbar Organ", "Percussive Organ", "Rock Organ", "Church Organ",
 "Reed Organ", "Accordion", "Harmonica", "Tango Accordion",
 "Acoustic Guitar (nylon)", "Acoustic Guitar (steel)", "Electric Guitar (jazz)", "Electric Guitar (clean)",
 "Electric Guitar (muted)", "Overdriven Guitar", "Distortion Guitar", "Guitar harmonics",
 "Acoustic Bass", "Electric Bass (finger)", "Electric Bass (pick)", "Fretless Bass",
 "Slap Bass 1", "Slap Bass 2", "Synth Bass 1", "Synth Bass 2",
 "Violin", "Viola", "Cello", "Contrabass",
 "Tremolo Strings", "Pizzicato Strings", "Orchestral Harp", "Timpani",
 "String Ensemble 1", "String Ensemble 2", "Synth Strings 1", "Synth Strings 2",
 "Choir Ahhs", "Voice Oohs", "Synth Choir", "Orchestra Hit",
 "Trumpet", "Trombone", "Tuba", "Muted Trumpet",
 "French Horn", "Brass Section", "Synth Brass 1", "Synth Brass 2",
 "Soprano Sax", "Alto Sax", "Tenor Sax", "Baritone Sax",
 "Oboe", "English Horn", "Bassoon", "Clarinet",
 "Piccolo", "Flute", "Recorder", "Pan Flute",
 "Blown Bottle", "Shakuhachi", "Whistle", "Ocarina",
 "Lead 1 (square)", "Lead 2 (sawtooth)", "Lead 3 (calliope)", "Lead 4 (chiff)",
 "Lead 5 (charang)", "Lead 6 (voice)", "Lead 7 (fifths)", "Lead 8 (bass + lead)",
 "Pad 1 (new age)", "Pad 2 (warm)", "Pad 3 (polysynth)", "Pad 4 (choir)",
 "Pad 5 (bowed)", "Pad 6 (metallic)", "Pad 7 (halo)", "Pad 8 (sweep)",
 "FX 1 (rain)", "FX 2 (soundtrack)", "FX 3 (crystal)", "FX 4 (atmosphere)",
 "FX 5 (brightness)", "FX 6 (goblins)", "FX 7 (echoes)", "FX 8 (sci-fi)",
 "Sitar", "Banjo", "Shamisen", "Koto",
 "Kalimba", "Bagpipe", "Fiddle", "Shanai",
 "Tinkle Bell", "Agogo", "Steel Drums", "Woodblock",
 "Taiko Drum", "Melodic Tom", "Synth Drum", "Reverse Cymbal",
 "Guitar Fret Noise", "Breath Noise", "Seashore", "Bird Tweet",
 "Telephone Ring", "Helicopter", "Applause", "Gunshot"
]
MIDI instruments mapping remains unchanged
midi_instruments = {name: index for index, name in enumerate(gm1_instrument_names)}
```

```

Define a helper function to randomly select from sets based on Caley Tables
def random_selection_from_caley(prime_elements):
 selection = random.choice(prime_elements)
 prime_elements.remove(selection) # Ensure uniqueness by removing the selected element
 return selection

def add_notes_with_caley_logic_to_tracks(track_names, preferences, number_of_notes=100):
 # Caley Table elements for pitch, duration, and volume
 pitch_elements = list(range(60, 65)) # Example pitches from C5 to C#5/Eb5
 duration_elements = [1, 2, 4] # Quarter, half, whole notes as example durations
 volume_elements = [64, 127] # Medium and high volumes

 random.shuffle(pitch_elements)
 random.shuffle(duration_elements)
 random.shuffle(volume_elements)

 for track_name in track_names:
 # Distribute notes evenly across tracks
 notes_for_track = number_of_notes // len(track_names)
 for _ in range(notes_for_track):
 pitch = pitch_elements.pop() if pitch_elements else random.choice(range(60, 72))
 duration = duration_elements.pop() if duration_elements else random.choice([1, 2, 4])
 volume = volume_elements.pop() if volume_elements else random.choice([64, 127])

 # Reset lists if empty to ensure continuous note generation
 if not pitch_elements: pitch_elements = list(range(60, 65))
 if not duration_elements: duration_elements = [1, 2, 4]
 if not volume_elements: volume_elements = [64, 127]

 # Calculate start time based on preferences or a simple increment
 start_time = _ * 1 # Simplified start time calculation

 # Add note to the track
 if track_name in tracks_config:
 tracks_config[track_name]['notes'].append((pitch, start_time, duration, volume))
 else:
 print(f"Track {track_name} not found.")

 # After distributing notes, proceed to generate the MIDI file or further processing

Modify or integrate a similar function to utilize Caley logic for note attributes
def add_notes_with_caley_logic(track_name, number_of_notes):
 # Define Caley Table elements (as an example, real implementation may vary)
 pitch_elements = [0, 1, 2, 3, 4] # Modulo 5 for pitch
 duration_elements = [0, 1, 2] # Modulo 3 for duration
 volume_elements = [0, 1] # Modulo 2 for volume

```

```

for _ in range(number_of_notes):
 # Randomly select pitch, duration, and volume
 pitch = random_selection_from_caley(pitch_elements) + 60 # Example pitch offset
 duration = random_selection_from_caley(duration_elements) + 1 # Ensure non-zero duration
 volume = random_selection_from_caley(volume_elements) * 127 # Scale volume

 # Check and reset the elements if all are used
 if not pitch_elements: pitch_elements = [0, 1, 2, 3, 4]
 if not duration_elements: duration_elements = [0, 1, 2]
 if not volume_elements: volume_elements = [0, 1]

 # Add the note with selected attributes
 if track_name in tracks_config:
 tracks_config[track_name]['notes'].append((pitch, _, duration, volume)) # Placeholder for start time

def get_user_mapping_preferences():
 # Enhanced to include start time customization
 print("Define how to map numbers to note attributes.")
 pitch_rule = input("Pitch mapping (e.g., 'decimal * 127'): ")
 time_rule = input("Start time mapping (e.g., 'index * 0.5' for staggered): ")
 duration_rule = input("Duration mapping (e.g., 'fixed', '1'): ")
 volume_rule = input("Volume mapping (e.g., 'fixed', '100'): ")

 return {
 'pitch_rule': pitch_rule,
 'time_rule': time_rule,
 'duration_rule': duration_rule,
 'volume_rule': volume_rule,
 }

Import math. Ensure this import is at the top of your script

def interpret_and_load_notes(file_path, track_name, preferences):
 if track_name not in tracks_config:
 print(f"Track '{track_name}' not found. Adding it automatically.")
 tracks_config[track_name] = {'instrument': 0, 'tempo': 120, 'pan': 64, 'notes': []}

 with open(file_path, 'r') as file:
 numbers = file.read().split()

 for index, number in enumerate(numbers):
 # Update the eval environment to include the math module
 eval_env = {
 'index': index,
 'number': float(number),
 'decimal': float(f"0.{str(number).split('.')[1]}"),
 'math': math # Include math module here
 }

```

```

pitch = eval(preferences['pitch_rule'], eval_env)
time = eval(preferences['time_rule'], eval_env)
duration = eval(preferences['duration_rule'], eval_env)
volume = eval(preferences['volume_rule'], eval_env)

tracks_config[track_name]['notes'].append((int(pitch), time, duration, int(volume)))

Function to display the instruments and their IDs
def list_available_instruments():
 print("Available MIDI Instruments:")
 for name, id in sorted(midi_instruments.items(), key=lambda item: item[1]):
 print(f"{id:3}: {name}")

def add_track(track_name):
 # Prompt the user for each setting
 print(f"Adding new track: {track_name}")

 # List available instruments for user selection
 list_available_instruments()
 instrument = int(input("Enter instrument number: "))

 tempo = int(input("Enter tempo (BPM): "))
 pan = int(input("Enter pan position (0-127, where 64 is center): "))

 # Initialize the new track with user-defined settings
 tracks_config[track_name] = {
 'instrument': instrument,
 'tempo': tempo,
 'pan': pan,
 'notes': [] # Initialize an empty list for notes
 }
 print(f"Track '{track_name}' added with instrument {instrument}, tempo {tempo}, and pan {pan}.")

def set_instrument():
 track_name = input("Enter track name: ")
 # Example usage
 list_available_instruments()
 instrument = int(input("Enter instrument number: "))
 if track_name in tracks_config:
 tracks_config[track_name]['instrument'] = instrument
 print(f"Instrument for '{track_name}' set to {instrument}.")
 else:
 print("Track not found.")

def set_track_pan():
 track_name = input("Enter track name: ")
 pan = int(input("Enter pan position (0-127): "))
 if track_name in tracks_config and 0 <= pan <= 127:

```

```

tracks_config[track_name]['pan'] = pan
print(f"Panning for '{track_name}' set to {pan}.")
else:
 print("Invalid track name or pan position.")

def add_note_to_track():
 track_name = input("Enter track name: ")
 pitch = int(input("Enter pitch (0-127): "))
 # Apply the language filter to the pitch
 if pitch % 10 not in settings['filter_set']:
 print(f"Pitch {pitch} does not pass the filter criteria.")
 return
 time = float(input("Enter time: "))
 duration = float(input("Enter duration: "))
 volume = int(input("Enter volume (0-127): "))
 if track_name in tracks_config:
 tracks_config[track_name]['notes'].append((pitch, time, duration, volume))
 print(f"Note added to '{track_name}'.")
 else:
 print("Track not found.")

def shift_notes_in_track():
 track_name = input("Enter track name: ")
 shift_amount = float(input("Enter shift amount (negative to shift left, positive to shift right): "))
 if track_name in tracks_config:
 shifted_notes = [(pitch, time + shift_amount, duration, volume) for pitch, time, duration, volume in
tracks_config[track_name]['notes']]
 tracks_config[track_name]['notes'] = shifted_notes
 print(f"Notes in '{track_name}' shifted by {shift_amount}.")
 else:
 print("Track not found.")

def generate_music(file_path):
 midi = MIDIFile(len(tracks_config)) # Number of tracks

 track_index = 0
 for track_name, config in tracks_config.items():
 midi.addTrackName(track_index, 0, track_name)
 midi.addTempo(track_index, 0, config['tempo'])
 midi.addProgramChange(track_index, track_index, 0, config['instrument'])
 midi.addControllerEvent(track_index, track_index, 0, 10, config['pan']) # Apply panning

 for note in config['notes']:
 pitch, start_time, duration, volume = note
 # Ensure that pitch and volume are integers
 pitch = int(pitch)
 volume = int(volume)
 midi.addNote(track_index, track_index, start_time, duration, pitch, volume)

```

```

track_index += 1

output_filename = file_path.replace('.txt', '_generated.mid')
with open(output_filename, 'wb') as outf:
 midi.writeFile(outf)

print(f"Music generated successfully and saved to {output_filename}")

def interpret_and_distribute_notes(file_path, track_names, preferences):
 with open(file_path, 'r') as file:
 numbers = file.read().split()

 note_count = len(numbers)
 notes_per_track = max(note_count // len(track_names), 1) # Ensure at least one note per track

 for index, number in enumerate(numbers):
 track_index = index // notes_per_track % len(track_names)
 track_name = track_names[track_index]

 # Assuming the environment setup for eval includes 'index', 'number', 'math'
 eval_env = {'index': index, 'number': float(number), 'decimal': float(f"0.{str(number).split('.')[1]}"),
 'math': math}
 pitch = eval(preferences['pitch_rule'], eval_env)
 time = eval(preferences['time_rule'], eval_env)
 duration = eval(preferences['duration_rule'], eval_env)
 volume = eval(preferences['volume_rule'], eval_env)

 # Add note to the appropriate track
 tracks_config[track_name]['notes'].append((int(pitch), time, duration, int(volume)))

def update_filter_set():
 new_set = input("Enter new filter set (e.g., 2,3,5): ")
 settings['filter_set'] = set(map(int, new_set.split(',')))
 print(f"Filter set updated to {settings['filter_set']}.")

def main_menu():
 print("\nMIDI Music Generation Workbench")
 print("1. Add Track")
 print("2. Set Instrument for Track")
 print("3. Set Track Pan")
 print("4. Add Note to Track")
 print("5. Shift Notes in Track")
 print("6. Update Language Filter Set")
 print("7. Generate MIDI File")
 print("8. Add note sequence to a Track")
 print("9. Help and Documentation") # New help command
 print("0. Exit")

```

```

def display_help():
 print("Help and Documentation:")
 print("- 1 Add Track: Adds a new track. Specify instrument, tempo, and pan.")
 print("- 2 Set Instrument: Change the instrument for a track.")
 print("- 3 Set Track Pan: Adjusts stereo pan for a track.")
 print("- 4 Add Note: Add a note with pitch, time, duration, and volume.")
 print("- 5 Shift Notes: Moves all notes in a track in time.")
 print("- 6 Update Filter Set: Change pitch filter criteria.")
 print("- 7 Generate MIDI: Creates a MIDI file from your configurations.")
 print("- 8 Add Sequence: Add a sequence of notes to tracks, with timing and distribution options.")
 print("- 9 Help: Displays this help information.")
 print("Each function will guide you with prompts for necessary inputs.")

def get_number_of_notes_from_file(file_path):
 with open(file_path, 'r') as file:
 # Example: Assuming each line in the file represents a different note
 # or each value is a note, adjust this logic based on the actual file format
 lines = file.readlines()
 number_of_notes = len(lines) # Each line represents one note
 return number_of_notes

while True:
 main_menu()
 choice = int(input("Enter your choice: "))

 if choice == 0:
 print("Exiting program...")
 break
 elif choice == 1:
 tn = input("Enter track name: ")
 add_track(tn)
 elif choice == 2:
 set_instrument()
 elif choice == 3:
 set_track_pan()
 elif choice == 4:
 add_note_to_track()
 elif choice == 5:
 shift_notes_in_track()
 elif choice == 6:
 update_filter_set()
 elif choice == 7:
 file_path = input("Enter the path to the input file (used for naming the MIDI file): ")
 generate_music(file_path)
 elif choice == 8:
 file_path = input("Enter the path to the note sequence file: ")
 distribute_notes = int(input("Distribute notes across multiple tracks automatically? Enter 1 for Yes, 0 for No: "))

```

```

if distribute_notes:
 num_tracks = int(input("Enter the number of tracks to distribute notes across: "))
 track_names = [f"Track_{i+1}" for i in range(num_tracks)]
 for name in track_names:
 if name not in tracks_config:
 # Directly configure new tracks to avoid user interaction for each track's settings
 tracks_config[name] = {'instrument': 0, 'tempo': 120, 'pan': 64, 'notes': []}
 print(f"Track '{name}' added with default settings.")

Determine the number of notes from the file's content
number_of_notes = get_number_of_notes_from_file(file_path)

Call the function to distribute notes across tracks using Caley logic
add_notes_with_caley_logic_to_tracks(track_names, number_of_notes)

elif choice == 9:
 display_help()
else:
 print("Invalid choice. Please enter a number between 0 and 9.")

```

----- Content of Music.py -----  
from midiutil import MIDIFile

```

def generate_music_with_modulo(file_path, filter_set={2, 3, 5}, mod_values=[2, 3, 5]):
 with open(file_path, 'r') as file:
 content = file.read().split()

 # Filter and apply arithmetic operations
 filtered_content = [float(num) for num in content if any((int(digit) in filter_set) for digit in str(num)) if digit.isdigit())]

 # Initialize MIDI file
 midi = MIDIFile(1)
 track = 0
 time = 0
 midi.addTrackName(track, time, "Modulo Music")
 midi.addTempo(track, time, 120)

 channel = 0
 volume = 100

 for i, value in enumerate(filtered_content):
 for mod_value in mod_values:
 # Apply arithmetic operations with modulo
 pitch = (int(value * i) + int(value)) % mod_value + 60 # Modulo operation for pitch variation
 duration = 1

```

```
midi.addNote(track, channel, pitch, time, duration, volume)
time += 1 # Increment time for sequential placement of notes

Output MIDI file
output_file = file_path.replace('.txt', '_modulo.mid')
with open(output_file, 'wb') as outf:
 midi.writeFile(outf)

print(f"Generated music with modulo operations saved to {output_file}")
```

```
Example usage
file_path = input("Enter the path to the input file: ")
generate_music_with_modulo(file_path)
```

----- Content of NaturalLanguageVerification.py -----

```

#save image to directory file

time.sleep(3)

test = input("Run Test? Yes[y] or No[n]? ")
if(test == 'y'):
 gi1 = "build_1.txt"
 gi2 = "sequence_1.txt"
 gi3 = "1"
 gi4 = "lib_1.txt"

if(test == 'n'):
 gi1 = input("input build filename 'build_id.txt': ")
 gi2 = input("input sequence filename 'sequence_id.txt': ")
 gi3 = input("input Image filename 'id': ")
 gi4 = input("input lib filename 'lib_id.txt': ")

s_f = open(gi2, "w")
s_f.close()
s_f = open(gi2, "a")
lib_file_1 = open(gi4, "r")

print("Note: build file is named (Within the directory of this program)... 'build_id.txt'")
xgv = input("Have you entered your statement (1 = yes, 0 = no)? ... ")
if(xgv == '1'):
 print("Then Let Us Go ...")
elif(xgv == '0'):
 sys.exit(1)
elif(xgv != '1' and xgv != '0'):
 print("error")
 time.sleep(3)
 sys.exit(1)

st_f = open(gi1, "r")
st_buff = ['0']
bf_buff = ['0']
files_ = ['0']

nword = 0
pointer = 1
pointer2 = 1
jot_ = lib_file_1.read()

got_ = st_f.read()

wcount = 0

```

```
tcount = 0
hit1 = len(jot_)
hit2 = len(got_)
xtant = "0"
xtan = int(xtant)

while(jot_ != ""):

 bf_buff.extend(jot_)
 jot_ = lib_file_1.read()

while(got_ != ""):

 print(got_)
 st_buff.extend(got_)
 got_ = st_f.read()

del st_buff[0]
del bf_buff[0]

print(st_buff)
print("Size of Lib by Characters ...")
print(len(bf_buff))

#...

conda = len(st_buff)
yields = 0
yeldo = input("Number of Characters in your Language:")
yeldo_ = int(yeldo)
iter_99 = 0
iter_98 = -1
benine = ['0']
pnaf = len(a)
#s13 = 1
while(iter_99 < conda):

 if(iter_98 < yeldo_):
 iter_98 = iter_98 + 1
 if(iter_98 == yeldo_):
 iter_98 = 0

 if(st_buff[iter_99] == a[iter_98]):
 solumn_ = str(iter_98 + 1)
 s_f.write(solumn_)
 s_f.write(' ')
 benine.append(iter_98 + 1)
 iter_99 = iter_99 + 1
```

```
yields = yields + 1

del benine[0]

print(benine)
print("Length of array is ...")
print(conda)

print("random() : ", random.random())
master = Tk()
master.attributes('-fullscreen', True)
#a = 250
#b = 200
print("Welcome\n")
print("Tip: side length should be a factor of the image width and of the image height")
change1 = input("Enter side length of image block: ")
change = int(change1)
#a = 1880
#b = 1050
a1 = input("Enter width of image: ")
a = int(a1)
b1 = input ("Enter height of image: ")
b = int(b1)
pin_p = a/change
w = Canvas(master, width= a, height= b)
#psv = getscreen().getcanvas()

#files = 3
#files_buffer = input("Enter number of files to be generated (8^((width/side_length)*(height/side_length)):")

#files = int(files_buffer)

#c = ["red","blue","yellow","brown","purple","pink","green","orange"]
c = ["gold"]
xc = 0
zerox = 0
zeroy = 0
p = 1
range_for = int((a/change)*(b/change))
name = 1
cells = ((a//change)*(b//change))
upper = cells- 1
nbr_comb = math.pow(len(c),cells)
files = int(nbr_comb)
#print(len(c))
rown = 0
```

```
#rdiv = math.pow(len(c),cells- 1)
#print(rdiv)
#cell = (row/rdiv) % (len(c))
#print(cell)

#rdiv = math.pow(len(c),cells- 2)
#print(rdiv)
#cell = (row/rdiv) % (len(c))
#print(cell)
#def rone(h,j,l):
cell = (h/j) % l
celled = int(cell)
w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = c[celled])
file_count = 0
st_f.close()
s_f.close()
#giip = 1
aiip = 0
#for t in range(0,files):
x_coun = 0
create_a = ['0']
z_coun = 0
dcoun = 0
lk = 0

if(conda > 1000):
 print("Max Characters allowed: 1000")
 time.sleep(5)
 sys.exit(1)

while(z_coun < cells):
 create_a.append(lk)
 z_coun = z_coun + 1
zcoun = 0

while(zcoun < conda):
 create_a[benine[zcoun]] = benine[dcoun]
 zcoun = zcoun + 1
 dcoun = dcoun + 1

del create_a[0]

print(create_a)
zcoun = 0
tr_ = len(create_a)
print(tr_)
```

```
#file validation check (Natural Language Processing)
lib_file_1.close()
#####
lib_file_1 = open(gi4, "r")
st_f = open (gi1, "r")
gime = len(bf_buff)
grime = 0
liner = 0
chime = len(st_buff)
pount = 0
lount = 0
chount = 0

for tiy in range(0,chime):
 if(st_buff[tiy] == '\n'):
 chount = chount + 1
 chount = chount
 print("There are ... ")
 print(chount)
 print("files in build.")
 gime_ = gime*(chount)
 #print(gime)
 ijn = 0
 ij = 0
 g = 0

 ij1 = 0
 pip = len(benine)
 kilo = lib_file_1.readline()
 kiloe = st_f.readline()
 f = 0

 while(f < gime_ or kiloe != ""):

 if(kilo == ""):
 lib_file_1.seek(0,0)
 if(kilo == kiloe):
 xtant = 1
 kiloe = st_f.readline()
 print("SUCCESS")
 #print(f)
 f = f + 1
 if(kiloe == ""):
 break
 elif(kilo != kiloe and f < gime_):
 kilo = lib_file_1.readline()
 xtant = 0
 #print(f)
 f = f + 1
```

```
if(kiloe == ""):
 break
else:
 print("Error")
 time.sleep(5)
 break

g = 0

if(xtant == 1):
 print("Success, Build file is valid ...")
 time.sleep(5)
elif(xtant == 0):
 print("Build file is not valid")
 time.sleep(5)
 exit()
else:
 print("Fatal Error")
 time.sleep(5)
 exit()
```

```
st_f.close()
lib_file 1.close()
```

```

for t in range(0,files):
 file_count = file_count + 1
 #if(file_count == 121):
 # break
 switch = 1
 sw = 0
 #for x in range(range_for):
 #for x in range (0,cells):
 x = 0
 col = cells-1
 #print(rown)
 pixlp = 0
 while(x < cells):
 #f = open('%s.ps' % name, 'wb')
 #f.close
 if(switch == 1):

```

```

row = 1
nxleft = 0
nxright = change
nyleft = 0
nyright = change
zerox = 0
zeroy = 0
c_length = len(c)
switch = 0
#ran = random.randint(0,c_length- 1)

rdiv = math.pow(len(c),col)
cell = (rown/rdiv) % (len(c))
#rnone(row,rdiv,len(c))
#print(cell)
#print(rdiv)
celled = int(cell)
#print(celled)
#print(cell)
#w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[ran], outline = c[ran])

kij = st_buff[aiilp]

#for inh in range(0,100*conda):

if(create_a[zcoun] > 0):
 w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = "green", width = 0)
 #giilp = 0
 #zcoun += 1

if(create_a[zcoun] == 0):
 w.create_rectangle(zerox, nyleft, nxright,nyright, fill = "purple", outline = "black", width = 0)

zcoun = zcoun + 1

#z_coun = z_coun + 1
#pixlp = 0
#if(giilp > 100):
giilp = 0

if(x <= cells):
 col = col- 1
#w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[ran])
#zxbuffer = zerox
#w.place(x = zerox, y = zeroy)
#w.place(bordemode = OUTSIDE, x = zerox + change, y = zeroy)
w.grid(row = zeroy, column = zerox + change)

```

```

if(p >= pin_p and p%pin_p == 0):
 zeroy = zeroy + change
 zerox = -change
 #zerox = 0
 nxleft = change
 nxright = 0
 nyleft = nyleft + change
 nyright = nyright + change
 zerox = zerox + change
 p = p + 1
 #xc + 1
 nxright = nxright + change
 if(xc == 3):
 xc = 0
 x = x + 1
 rown = rown + 1
 ce = str(name)
 w.update()

def savefirst():
 w.postscript(file ='Map_' + gi3 + '.ps', colormode='color')

savefirst()

name = name + 1
#f_p = 'D:\Kaliber\Portfolio\Content\My PhotoBook\Content\'
#os.rename(f_p + '1.ps', f_p + '2.ps')

#process = subprocess.Popen(["ps2pdf", ce + ".ps", ce + ".pdf"], shell=True)

#Contender 1 For File Saving
"""
def savefirst():
 cnv = getscreen().getcanvas()
 global hen
 ps = cnv.postscript(colormode = 'color')
 hen = filedialog.asksaveasfilename(defaultextension = '.jpg')
 im = Image.open(io.BytesIO(ps.encode('utf-8')))
 im.save(hen + '.jpg')
#savefirst()
"""

Second Contender For File Saving

```

```
def save(w):
 ps = w.canvas.postscript(colormode='color')
 img = Image.open(io.BytesIO(ps.encode('utf-8')))
 img.save('testing.jpg')
 """
 #w.update()
 #script = ce + ".ps"
 #w.save(script)
 #os.startfile(script)
 #print(script)

 print("Done")

#master.mainloop()

----- Content of notepad_4.py-----
import tkinter as tk
from tkinter import filedialog

class Notepad:
 def __init__(self, root):
 self.file_path = None

 self.notepad = tk.Toplevel(root)
 self.notepad.title("Notepad")

 #self.text_area = tk.Text(self.notepad, wrap=tk.WORD)
 #self.text_area.pack(expand=1, fill=tk.BOTH, side=tk.TOP)

 self.scroll_frame = tk.Frame(self.notepad)
 self.scroll_frame.pack(side=tk.TOP, fill=tk.BOTH, expand=1)

 self.text_area = tk.Text(self.scroll_frame, wrap=tk.WORD)
 self.text_area.pack(side=tk.LEFT, fill=tk.BOTH, expand=1)

 # Add scrollbar to the text area
 self.scrollbar = tk.Scrollbar(self.scroll_frame, command=self.text_area.yview)
 self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
 self.text_area.config(yscrollcommand=self.scrollbar.set)

 # Bottom Frame for Counts
 self.bottom_frame = tk.Frame(self.notepad)
 self.bottom_frame.pack(side=tk.BOTTOM, fill=tk.X)

 self.line_count_label = tk.Label(self.bottom_frame, text="Lines: 0")
 self.line_count_label.pack(side=tk.LEFT, padx=10)
```

```
self.char_count_label = tk.Label(self.bottom_frame, text="Chars: 0")
self.char_count_label.pack(side=tk.LEFT, padx=10)

self.text_area.bind('<KeyRelease>', self.on_text_changed)

Check for duplicates after typing
self.text_area.bind('<KeyRelease>', lambda event: self.highlight_duplicates())

Binding the shortcut key
self.text_area.bind('<Control-s>', self.quick_save)

menu = tk.Menu(self.notepad)
file_menu = tk.Menu(menu, tearoff=0)
file_menu.add_command(label="New", command=self.new_file)
file_menu.add_command(label="Open", command=self.open_file)
file_menu.add_command(label="Save", command=self.save_file)
file_menu.add_command(label="Save As", command=self.save_file_as)

menu.add_cascade(label="File", menu=file_menu)
self.notepad.config(menu=menu)

def new_file(self):
 self.text_area.delete(1.0, tk.END)
 self.file_path = None
 self.update_counts()

def open_file(self):
 self.file_path = filedialog.askopenfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"),
 ("All Files", "*.*")])
 if self.file_path:
 self.text_area.delete(1.0, tk.END)
 with open(self.file_path, "r") as file:
 self.text_area.insert(tk.INSERT, file.read())
 self.highlight_duplicates()
 self.update_counts()

def save_file(self):
 if not self.file_path:
 self.save_file_as()
 else:
 with open(self.file_path, "w") as file:
 file.write(self.text_area.get(1.0, tk.END))

def save_file_as(self):
 self.file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"),
 ("All Files", "*.*")])
 if self.file_path:
 with open(self.file_path, "w") as file:
 file.write(self.text_area.get(1.0, tk.END))
```

```

def quick_save(self, event=None):
 if self.file_path:
 with open(self.file_path, "w") as file:
 file.write(self.text_area.get(1.0, tk.END))
 else:
 self.save_file_as()

def highlight_duplicates(self):
 lines = self.text_area.get(1.0, tk.END).strip().split("\n")
 duplicates = [line for line in lines if lines.count(line) > 1]

 # Clear existing tags
 self.text_area.tag_remove("duplicate", 1.0, tk.END)

 # Highlight lines with duplicates
 for duplicate in duplicates:
 start_index = "1.0"
 while True:
 start_index = self.text_area.search(duplicate, start_index, nocase=1, stopindex=tk.END)
 if not start_index:
 break
 end_index = f"{start_index} lineend+1c"
 self.text_area.tag_add("duplicate", start_index, end_index)
 self.text_area.tag_config("duplicate", background="red")
 start_index = end_index

def on_text_changed(self, event=None):
 self.update_counts()

def update_counts(self):
 """
 chars = len(self.text_area.get(1.0, tk.END)) - 1 # -1 to exclude the final newline char added by Text
 widget
 lines = self.text_area.get(1.0, tk.END).count("\n")

 self.line_count_label.config(text=f"Lines: {lines}")
 self.char_count_label.config(text=f"Chars: {chars}")
 """

 text_content = self.text_area.get(1.0, tk.END)
 chars = len(text_content) # Count every character including spaces, tabs, newlines

 # If you want to exclude any character from the count, simply subtract its count from 'chars'.
 # For example, to exclude spaces: chars -= text_content.count(' ')

 lines = text_content.count("\n")

 self.line_count_label.config(text=f"Lines: {lines}")
 self.char_count_label.config(text=f"Chars: {chars}")

```

```

if __name__ == "__main__":
 root = tk.Tk()
 root.title("Main Window")
 tk.Button(root, text="Open Notepad", command=lambda: Notepad(root)).pack()
 root.mainloop()

----- Content of notepad_app-2.py-----
import tkinter as tk
from tkinter import filedialog

def open_notepad(root):
 notepad = tk.Toplevel(root)
 notepad.title("Notepad")

 text_area = tk.Text(notepad, wrap=tk.WORD)
 text_area.pack(expand=1, fill=tk.BOTH)

 # Check for duplicates after typing
 text_area.bind('<KeyRelease>', lambda event: highlight_duplicates(text_area))

 menu = tk.Menu(notepad)
 file_menu = tk.Menu(menu, tearoff=0)
 file_menu.add_command(label="New", command=lambda: new_file(text_area))
 file_menu.add_command(label="Open", command=lambda: open_file(text_area))
 file_menu.add_command(label="Save", command=lambda: save_file(text_area))
 file_menu.add_command(label="Save As", command=lambda: save_file_as(text_area))

 menu.add_cascade(label="File", menu=file_menu)
 notepad.config(menu=menu)

def new_file(text_area):
 text_area.delete(1.0, tk.END)

def open_file(text_area):
 file_path = filedialog.askopenfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")])
 if file_path:
 text_area.delete(1.0, tk.END)
 with open(file_path, "r") as file:
 text_area.insert(tk.INSERT, file.read())
 highlight_duplicates(text_area)

def save_file(text_area):
 file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")])
 if file_path:
 with open(file_path, "w") as file:
 file.write(text_area.get(1.0, tk.END))

```

```

def save_file_as(text_area):
 file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")])
 if file_path:
 with open(file_path, "w") as file:
 file.write(text_area.get(1.0, tk.END))

def highlight_duplicates(text_area):
 lines = text_area.get(1.0, tk.END).strip().split('\n')
 duplicates = [line for line in lines if lines.count(line) > 1]

 # Clear existing tags
 text_area.tag_remove("duplicate", 1.0, tk.END)

 # Highlight lines with duplicates
 for duplicate in duplicates:
 start_index = "1.0"
 while True:
 start_index = text_area.search(duplicate, start_index, nocase=1, stopindex=tk.END)
 if not start_index:
 break
 end_index = f'{start_index} lineend+1c'
 text_area.tag_add("duplicate", start_index, end_index)
 text_area.tag_config("duplicate", background="red")
 start_index = end_index

if __name__ == "__main__":
 root = tk.Tk()
 root.title("Main Window")
 tk.Button(root, text="Open Notepad", command=lambda: open_notepad(root)).pack()
 root.mainloop()

```

----- Content of notepad\_app-3.py-----

```

import tkinter as tk
from tkinter import filedialog

class Notepad:
 def __init__(self, root):
 self.file_path = None

 self.notepad = tk.Toplevel(root)
 self.notepad.title("Notepad")

 self.text_area = tk.Text(self.notepad, wrap=tk.WORD)
 self.text_area.pack(expand=1, fill=tk.BOTH, side=tk.TOP)

 # Bottom Frame for Counts
 self.bottom_frame = tk.Frame(self.notepad)

```

```
self.bottom_frame.pack(side=tk.BOTTOM, fill=tk.X)

self.line_count_label = tk.Label(self.bottom_frame, text="Lines: 0")
self.line_count_label.pack(side=tk.LEFT, padx=10)

self.char_count_label = tk.Label(self.bottom_frame, text="Chars: 0")
self.char_count_label.pack(side=tk.LEFT, padx=10)

self.text_area.bind('<KeyRelease>', self.on_text_changed)

Check for duplicates after typing
self.text_area.bind('<KeyRelease>', lambda event: self.highlight_duplicates())

Binding the shortcut key
self.text_area.bind('<Control-s>', self.quick_save)

menu = tk.Menu(self.notepad)
file_menu = tk.Menu(menu, tearoff=0)
file_menu.add_command(label="New", command=self.new_file)
file_menu.add_command(label="Open", command=self.open_file)
file_menu.add_command(label="Save", command=self.save_file)
file_menu.add_command(label="Save As", command=self.save_file_as)

menu.add_cascade(label="File", menu=file_menu)
self.notepad.config(menu=menu)

def new_file(self):
 self.text_area.delete(1.0, tk.END)
 self.file_path = None
 self.update_counts()

def open_file(self):
 self.file_path = filedialog.askopenfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"),
 ("All Files", "*.*")])
 if self.file_path:
 self.text_area.delete(1.0, tk.END)
 with open(self.file_path, "r") as file:
 self.text_area.insert(tk.INSERT, file.read())
 self.highlight_duplicates()
 self.update_counts()

def save_file(self):
 if not self.file_path:
 self.save_file_as()
 else:
 with open(self.file_path, "w") as file:
 file.write(self.text_area.get(1.0, tk.END))

def save_file_as(self):
 self.file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text Files", "*.*"),
```

```

("All Files", "*.*"))
if self.file_path:
 with open(self.file_path, "w") as file:
 file.write(self.text_area.get(1.0, tk.END))

def quick_save(self, event=None):
 if self.file_path:
 with open(self.file_path, "w") as file:
 file.write(self.text_area.get(1.0, tk.END))
 else:
 self.save_file_as()

def highlight_duplicates(self):
 lines = self.text_area.get(1.0, tk.END).strip().split("\n")
 duplicates = [line for line in lines if lines.count(line) > 1]

 # Clear existing tags
 self.text_area.tag_remove("duplicate", 1.0, tk.END)

 # Highlight lines with duplicates
 for duplicate in duplicates:
 start_index = "1.0"
 while True:
 start_index = self.text_area.search(duplicate, start_index, nocase=1, stopindex=tk.END)
 if not start_index:
 break
 end_index = f"{start_index} lineend+1c"
 self.text_area.tag_add("duplicate", start_index, end_index)
 self.text_area.tag_config("duplicate", background="red")
 start_index = end_index

def on_text_changed(self, event=None):
 self.update_counts()

def update_counts(self):
 """
 chars = len(self.text_area.get(1.0, tk.END)) - 1 # -1 to exclude the final newline char added by Text
 widget
 lines = self.text_area.get(1.0, tk.END).count("\n")

 self.line_count_label.config(text=f"Lines: {lines}")
 self.char_count_label.config(text=f"Chars: {chars}")
 """

 text_content = self.text_area.get(1.0, tk.END)
 chars = len(text_content) # Count every character including spaces, tabs, newlines

 # If you want to exclude any character from the count, simply subtract its count from 'chars'.
 # For example, to exclude spaces: chars -= text_content.count(' ')

 lines = text_content.count("\n")

```

```
self.line_count_label.config(text=f"Lines: {lines}")
self.char_count_label.config(text=f"Chars: {chars}")

if __name__ == "__main__":
 root = tk.Tk()
 root.title("Main Window")
 tk.Button(root, text="Open Notepad", command=lambda: Notepad(root)).pack()
 root.mainloop()

----- Content of notepad_app-4.py-----
import tkinter as tk
from tkinter import filedialog

class Notepad:
 def __init__(self, root):
 self.file_path = None

 self.notepad = tk.Toplevel(root)
 self.notepad.title("Notepad")

 #self.text_area = tk.Text(self.notepad, wrap=tk.WORD)
 #self.text_area.pack(expand=1, fill=tk.BOTH, side=tk.TOP)

 self.scroll_frame = tk.Frame(self.notepad)
 self.scroll_frame.pack(side=tk.TOP, fill=tk.BOTH, expand=1)

 self.text_area = tk.Text(self.scroll_frame, wrap=tk.WORD)
 self.text_area.pack(side=tk.LEFT, fill=tk.BOTH, expand=1)

 # Add scrollbar to the text area
 self.scrollbar = tk.Scrollbar(self.scroll_frame, command=self.text_area.yview)
 self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
 self.text_area.config(yscrollcommand=self.scrollbar.set)

 # Bottom Frame for Counts
 self.bottom_frame = tk.Frame(self.notepad)
 self.bottom_frame.pack(side=tk.BOTTOM, fill=tk.X)

 self.line_count_label = tk.Label(self.bottom_frame, text="Lines: 0")
 self.line_count_label.pack(side=tk.LEFT, padx=10)

 self.char_count_label = tk.Label(self.bottom_frame, text="Chars: 0")
 self.char_count_label.pack(side=tk.LEFT, padx=10)

 self.text_area.bind('<KeyRelease>', self.on_text_changed)
```

```
Check for duplicates after typing
self.text_area.bind('<KeyRelease>', lambda event: self.highlight_duplicates())

Binding the shortcut key
self.text_area.bind('<Control-s>', self.quick_save)

menu = tk.Menu(self.notepad)
file_menu = tk.Menu(menu, tearoff=0)
file_menu.add_command(label="New", command=self.new_file)
file_menu.add_command(label="Open", command=self.open_file)
file_menu.add_command(label="Save", command=self.save_file)
file_menu.add_command(label="Save As", command=self.save_file_as)

menu.add_cascade(label="File", menu=file_menu)
self.notepad.config(menu=menu)

def new_file(self):
 self.text_area.delete(1.0, tk.END)
 self.file_path = None
 self.update_counts()

def open_file(self):
 self.file_path = filedialog.askopenfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"),
("All Files", "*.*")])
 if self.file_path:
 self.text_area.delete(1.0, tk.END)
 with open(self.file_path, "r") as file:
 self.text_area.insert(tk.INSERT, file.read())
 self.highlight_duplicates()
 self.update_counts()

def save_file(self):
 if not self.file_path:
 self.save_file_as()
 else:
 with open(self.file_path, "w") as file:
 file.write(self.text_area.get(1.0, tk.END))

def save_file_as(self):
 self.file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"),
("All Files", "*.*")])
 if self.file_path:
 with open(self.file_path, "w") as file:
 file.write(self.text_area.get(1.0, tk.END))

def quick_save(self, event=None):
 if self.file_path:
 with open(self.file_path, "w") as file:
 file.write(self.text_area.get(1.0, tk.END))
 else:
```

```
 self.save_file_as()

def highlight_duplicates(self):
 lines = self.text_area.get(1.0, tk.END).strip().split("\n")
 duplicates = [line for line in lines if lines.count(line) > 1]

 # Clear existing tags
 self.text_area.tag_remove("duplicate", 1.0, tk.END)

 # Highlight lines with duplicates
 for duplicate in duplicates:
 start_index = "1.0"
 while True:
 start_index = self.text_area.search(duplicate, start_index, nocase=1, stopindex=tk.END)
 if not start_index:
 break
 end_index = f"{start_index} lineend+1c"
 self.text_area.tag_add("duplicate", start_index, end_index)
 self.text_area.tag_config("duplicate", background="red")
 start_index = end_index

def on_text_changed(self, event=None):
 self.update_counts()

def update_counts(self):
 """
 chars = len(self.text_area.get(1.0, tk.END))- 1 # -1 to exclude the final newline char added by Text
 widget
 lines = self.text_area.get(1.0, tk.END).count("\n")

 self.line_count_label.config(text=f"Lines: {lines}")
 self.char_count_label.config(text=f"Chars: {chars}")
 """

 text_content = self.text_area.get(1.0, tk.END)
 chars = len(text_content) # Count every character including spaces, tabs, newlines

 # If you want to exclude any character from the count, simply subtract its count from 'chars'.
 # For example, to exclude spaces: chars-= text_content.count(' ')

 lines = text_content.count("\n")

 self.line_count_label.config(text=f"Lines: {lines}")
 self.char_count_label.config(text=f"Chars: {chars}")

if __name__ == "__main__":
 root = tk.Tk()
 root.title("Main Window")
 tk.Button(root, text="Open Notepad", command=lambda: Notepad(root)).pack()
 root.mainloop()
```

```
----- Content of notepad_app.py-----
import tkinter as tk
from tkinter import filedialog

def open_notepad(root):
 notepad = tk.Toplevel(root)
 notepad.title("Notepad")

 text_area = tk.Text(notepad, wrap=tk.WORD)
 text_area.pack(expand=1, fill=tk.BOTH)

 menu = tk.Menu(notepad)
 file_menu = tk.Menu(menu, tearoff=0)
 file_menu.add_command(label="New", command=lambda: new_file(text_area))
 file_menu.add_command(label="Open", command=lambda: open_file(text_area))
 file_menu.add_command(label="Save", command=lambda: save_file(text_area))
 file_menu.add_command(label="Save As", command=lambda: save_file_as(text_area))

 menu.add_cascade(label="File", menu=file_menu)
 notepad.config(menu=menu)

def new_file(text_area):
 text_area.delete(1.0, tk.END)

def open_file(text_area):
 file_path = filedialog.askopenfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")])
 if file_path:
 text_area.delete(1.0, tk.END)
 with open(file_path, "r") as file:
 text_area.insert(tk.INSERT, file.read())

def save_file(text_area):
 file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")])
 if file_path:
 with open(file_path, "w") as file:
 file.write(text_area.get(1.0, tk.END))

def save_file_as(text_area):
 file_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text Files", "*.txt"), ("All Files", "*.*")])
 if file_path:
 with open(file_path, "w") as file:
 file.write(text_area.get(1.0, tk.END))

if __name__ == "__main__":
 root = tk.Tk()
```

```
root.title("Main Window")
tk.Button(root, text="Open Notepad", command=lambda: open_notepad(root)).pack()
root.mainloop()
```

----- Content of PANS ptia-hdl.cpp -----

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
using namespace std;
//put together by Dominic Alexander Cooper
int main(){
 puts("(pt) == Universal Set = Computer");
 puts("(pt)i - sub-set of (pt)");
 puts("(pt)ia = element of sub-set of (pt)");
 puts("ti = ia ");
 puts("t = i of (pt)ia ");
 puts("i = a of ((pt)ia)");
 puts("\nProgressive Abstract Numerical Solution");
 //Code Adapted by DAC from lyst on https://stackoverflow.com
 //k+1 = no. of elements
 //n = exponent = number of cells
 //the k and n values must perfectly fit the size of the set of elements in question
 FILE *p; p = fopen("PROGRESSIVE_ANS_px_RENAME.txt","w");
 //char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
 '\n','\t','\\','\"','<','>','?','!','@','#','~','[','{','}','^','|','!','"','$', '%','^','&','*','(',')','-','_',
 '+','=','`','A','B','C','D','E','F','G','H',
 'I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9'};
 int a[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,3,
 8,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,
 73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,10,
 05,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,
 130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151
 4,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,
 79,180,181,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,
 204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,
 8,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,
 53,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,
 278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,
 2,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,
 27,328,329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,346,347,348,
 352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,373,
 6,377,378,379,380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,397,398,
 01,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,
 426,427,428,429,430,431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,
 0,451,452,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,469,470,471,472,
 75,476,477,478,479,480,481,482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,
 500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521
}

```

```
4,525,526,527,528,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,5
49,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,
574,575,576,577,578,579,580,581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597,59
8,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,618,619,620,621,622,6
23,624,625,626,627,628,629,630,631,632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,
648,649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665,666,667,668,669,670,671,67
2,673,674,675,676,677,678,679,680,681,682,683,684,685,686,687,688,689,690,691,692,693,694,695,696,6
97,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721,
722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,74
6,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,7
71,772,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,
796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,82
0,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,8
45,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,
870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,89
4,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,9
19,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,
944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,96
8,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,9
93,994,995,996,997,998,999};
```

```
int pin;
```

```
int pr, pc;
printf("\nLet us begin\n\n");
pc = 1000;
for(pr = 0; pr < pc; pr++){
 printf("%d %d\n", pr, a[pr]);
}
char choice;
cout << "Enter a to uniquely use the 1 to 1000 identifiers for the Default Library or c to create a
custom Library: ";
cin >> choice;

cout << "Enter the size of your array: ";
cin >> pin;

int array[pin], inn, position;
int generate;
cout << "\nDo you want manually create your array (0) or, do you want to generate your array for
a range of numbers (1): ";
cin >> generate;
int lbrange;
int ubrange;
//int i_lbrange, i_ubrange;
int r_check;
r_check = 0;

if(choice == 'c'){

 //cout << "Enter the size of your array: ";
 //cin >> pin;
```

```

//int array[pin], inn, position;

if(generate == 0){
 cout << "\n\n Enter the " << pin << " elements of your array: \n\n";
 for(inn = 0; inn < pin; inn++){
 cin >> array[inn];
 }
}
if(generate == 1){
 cout << "\nEnter the lower inclusive bound of the array range of numbers:
";
 cin >> lrange;
//i_lrange = int(lrange);
 cout << "\nEnter the upper inclusive bound of the array range of numbers:
";
 cin >> ubrange;
//i_ubrange = ubrange;
 r_check = lrange;
//generate custom array from user input;
 for(inn = 0; inn < pin; inn++){
 // r_check = r_check + 1;
 array[inn] = r_check;
 printf("\n%d", r_check);
 r_check = r_check + 1;
 if (r_check > ubrange){
 break;
 }
 }
}

// for(inn = 0; inn < pin; inn++){
// cin >> array[inn];
// }
}

//char a[] = {'1','2','3','4','p1','p2','p3','p4','p5','p6'};
//char b[] = {'a','b'};
//char c[] = {'0','1','2','3'};
//int n = 4; //int k = 3; //int n = x;
//int k = 100;

//int k = strlen(a)- 1;
int k;
if(choice == 'c'){
 k = pin- 1;
}
if(choice == 'a'){
 k = 1000- 1;
}

```

```

}

printf("\n\tnk = %d", k);
int noc; printf("\n\ttn = ");
scanf("%d", &noc);
printf("\n\tNumber Of FILE Cells = %d", noc);
int n = noc;
int row, col;
int cell;
int rdiv;
int id;
id = 0;
int rin;
int nbr_comb = pow(k+1, n);
int ai, ci, switch_c, switch_a;

if(choice == 'c'){

 switch_c = 0;
 cout << "\n\nEnter nth File System (Number of Cells Per File of the nth File System). Let
the value be equal to n for a progressionless session: ";
 cin >> ci;

 for(ci; n < ci + 1; n++){
 if(switch_c == 1){
 nbr_comb = pow(k + 1, n);
 row = 0;
 }

 for (row=0; row < nbr_comb; row++){
 id++; //fprintf(p,"n\n%dCF%6d\nn", n,id);
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);

 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 if(col == 0){
 fprintf(p,"%d\n", a[rin]);
 }
 if(col != 0){
 fprintf(p,"%d ", a[rin]);
 }

 //fprintf(p, "%d", a[cell]);
 }
 //printf("\n");
 }
 }
}

```

```

 switch_c = 1;
 }
}

int check;
check = pin - 1000;

if(choice == 'a' && check == 0){

 switch_a = 0;
 cout << "\n\nEnter nth File System (Number of Cells Per File of the nth File System). Let
the value be equal to n for a progressionless session:";
 cin >> ai;

 for(n; n < ai + 1; n++){
 if(switch_a == 1){
 nbr_comb = pow(k + 1, n);
 row = 0;
 }
 for (row=0; row < nbr_comb; row++){
 id++;
 fprintf(p, "\n\n%dCF%d\n\n", n, id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p, "%c", a[cell]);

 cell = (row/rdiv) % (k+1);
 //rin = array[cell];
 if(col == 0){
 fprintf(p,"%d", a[cell]);
 }
 if(col != 0){
 fprintf(p,"%d ", a[cell]);
 }
 //fprintf(p, "%d", a[cell]);
 }
 //printf("\n");
 }
 switch_a = 1;
 }

 //fprintf(p, "\n\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
 fclose(p);
 //end of adaptation
 return 0;
}

```

----- Content of PANS\_ptia.cpp-----

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
using namespace std;
//put together by Dominic Alexander Cooper
int main(){
 puts("(pt) == Universal Set = Computer");
 puts("(pt)i = sub-set of (pt)");
 puts("(pt)ia = element of sub-set of (pt)");
 puts("ti = ia");
 puts("t = i of (pt)ia ");
 puts("i = a of ((pt)ia)");
 puts("\nProgressive Abstract Numerical Solution");
 //Code Adapted by DAC from lyst on https://stackoverflow.com
 //k+1 = no. of elements
 //n = exponent = number of cells
 //the k and n values must perfectly fit the size of the set of elements in question
 FILE *p; p = fopen("PROGRESSIVE_ANS_px_RENAME.txt","w");
 //char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
 '\n','\t','\\','\"','<','>','?','!','@','#','~','[','{','}',',','!','"','$','%','^','*','(',')','_','_',
 '+','=','','A','B','C','D','E','F','G','H',
 'I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9';
 int a[] =
{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,3
8,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,
73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,1
05,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,129,
130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,15
4,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,1
79,180,181,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,
204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,22
8,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,2
53,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,
278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,30
2,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,3
27,328,329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,346,347,348,349,350,351,
352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,373,374,375,37
6,377,378,379,380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,397,398,399,400,4
01,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,423,424,425,
426,427,428,429,430,431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,448,449,45
0,451,452,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,469,470,471,472,473,474,4
75,476,477,478,479,480,481,482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,499,
500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521,522,523,52
4,525,526,527,528,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,5
49,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,
574,575,576,577,578,579,580,581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597,59
8,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,618,619,620,621,622,6
```

```

23,624,625,626,627,628,629,630,631,632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,
648,649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665,666,667,668,669,670,671,67
2,673,674,675,676,677,678,679,680,681,682,683,684,685,686,687,688,689,690,691,692,693,694,695,696,6
97,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721,
722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,74
6,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,7
71,772,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,
796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,82
0,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,8
45,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,
870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,89
4,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,9
19,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,
944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,96
8,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,9
93,994,995,996,997,998,999};

int pin;

int pr, pc;
printf("\nLet us begin\n\n");
pc = 1000;
for(pr = 0; pr < pc; pr++){
 printf("%d %d\n", pr, a[pr]);
}
char choice;
cout << "Enter a to uniquely use the 1 to 1000 identifiers for the Default Library or c to create a
custom Library: ";
cin >> choice;

cout << "Enter the size of your array: ";
cin >> pin;

int array[pin], inn, position;
int generate;
cout << "\nDo you want manually create your array (0) or, do you want to generate your array for
a range of numbers (1): ";
cin >> generate;
int lbrange;
int ubrange;
//int i_lbrange, i_ubrange;
int r_check;
r_check = 0;

if(choice == 'c'){
 //cout << "Enter the size of your array: ";
 //cin >> pin;
 //int array[pin], inn, position;
}

```

```

if(generate == 0){
 cout << "\n\n Enter the " << pin << " elements of your array: \n\n";
 for(inn = 0; inn < pin; inn++){
 cin >> array[inn];
 }
}
if(generate == 1){
 cout << "\nEnter the lower inclusive bound of the array range of numbers:
";
 cin >> lrange;
 //i_lrange = int(lrange);
 cout << "\nEnter the upper inclusive bound of the array range of numbers:
";
 cin >> ubrange;
 //i_ubrange = ubrange;
 r_check = lrange;
 //generate custom array from user input;
 for(inn = 0; inn < pin; inn++){
 // r_check = r_check + 1;
 array[inn] = r_check;
 printf("\n%d", r_check);
 r_check = r_check + 1;
 if (r_check > ubrange){
 break;
 }
 }
}

for(inn = 0; inn < pin; inn++){
 cin >> array[inn];
}
}

//char a[] = {'1','2','3','4','p1','p2','p3','p4','p5','p6'};
//char b[] = {'a','b'};
//char c[] = {'0','1','2','3'};
//int n = 4; //int k = 3; //int n = x;
//int k = 100;

//int k = strlen(a)- 1;
int k;
if(choice == 'c'){
 k = pin- 1;
}
if(choice == 'a'){
 k = 1000- 1;
}
printf("\n\tnk = %d", k);
int noc; printf("\n\ttn = ");
scanf("%d", &noc);

```

```

printf("\n\nNumber Of FILE Cells = %d", noc);
int n = noc;
int row, col;
int cell;
int rdiv;
int id;
id = 0;
int rin;
int nbr_comb = pow(k+1, n);
int ai, ci, switch_c, switch_a;

if(choice == 'c'){

 switch_c = 0;
 cout << "\n\nEnter nth File System (Number of Cells Per File of the nth File System). Let
the value be equal to n for a progressionless session: ";
 cin >> ci;

 for(ci; n < ci + 1; n++){
 if(switch_c == 1){
 nbr_comb = pow(k + 1, n);
 row = 0;
 }

 for (row=0; row < nbr_comb; row++){
 id++;
 fprintf(p, "\n\n%dCF%d\n\n", n,id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);

 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 if(col == 0){
 fprintf(p,"%d", a[rin]);
 }
 if(col != 0){
 fprintf(p,"%d ", a[rin]);
 }

 //fprintf(p, "%d", a[cell]);
 }
 //printf("\n");
 }
 switch_c = 1;
 }
}
int check;

```

```

check = pin- 1000;

if(choice == 'a' && check == 0){

 switch_a = 0;
 cout << "\n\nEnter nth File System (Number of Cells Per File of the nth File System). Let
the value be equal to n for a progressionless session: ";
 cin >> ai;

 for(n; n < ai + 1; n++){
 if(switch_a == 1){
 nbr_comb = pow(k + 1, n);
 row = 0;
 }
 for (row=0; row < nbr_comb; row++){
 id++;
 fprintf(p,"n\n%dCF%d\nn", n, id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);

 cell = (row/rdiv) % (k+1);
 //rin = array[cell];
 if(col == 0){
 fprintf(p,"%d", a[cell]);
 }
 if(col != 0){
 fprintf(p,"%d ", a[cell]);
 }
 //fprintf(p, "%d", a[cell]);
 }
 //printf("\n");
 }
 switch_a = 1;
 }
}

//fprintf(p,"n\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
fclose(p);
//end of adaptation
return 0;
}

```

----- Content of Particle\_Simulator.py-----

```

import numpy as np
import matplotlib.pyplot as plt

```

```

from mpl_toolkits.mplot3d import Axes3D
from PIL import Image
import os

Function to calculate the gravitational force between two particles
def calculate_gravitational_force(mass1, mass2, pos1, pos2, G=6.67430e-11):
 r = np.linalg.norm(pos1 - pos2)
 force_magnitude = (G * mass1 * mass2) / (r ** 2)
 force_direction = (pos2 - pos1) / r
 force = force_magnitude * force_direction
 return force

User-defined parameters
num_particles = int(input("Enter the number of particles: "))
time_step = float(input("Enter the time step (Δt): "))
num_time_steps = int(input("Enter the number of time steps: "))

Initialize particle properties
masses = np.random.uniform(1.0, 10.0, num_particles) # Random masses
positions = np.random.uniform(-100.0, 100.0, (num_particles, 3)) # Random initial positions
velocities = np.random.uniform(-1.0, 1.0, (num_particles, 3)) # Random initial velocities

Create a 3D scatter plot for visualization
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

Create a directory to store frame images
if not os.path.exists('frames'):
 os.makedirs('frames')

Simulation loop
for step in range(num_time_steps):
 # Calculate forces on each particle
 forces = np.zeros_like(positions)
 for i in range(num_particles):
 for j in range(num_particles):
 if i != j:
 force = calculate_gravitational_force(
 masses[i], masses[j], positions[i], positions[j]
)
 forces[i] += force

 # Update positions and velocities using Euler's method
 positions += velocities * time_step
 velocities += forces / masses[:, np.newaxis] * time_step

 # Update the scatter plot for visualization
 ax.cla()
 ax.set_xlabel('X')
 ax.set_ylabel('Y')

```

```

ax.set_zlabel('Z')
ax.scatter(positions[:, 0], positions[:, 1], positions[:, 2], c='b', marker='o')

Save the current frame as an image
frame = plt.gcf()
frame.savefig(f'frames/frame_{step:04d}.png') # Save the frame as an image

Create a GIF animation from the saved frames
images = [Image.open(f'frames/frame_{i:04d}.png') for i in range(num_time_steps)]
images[0].save('particle_simulation.gif', save_all=True, append_images=images[1:], duration=100, loop=0)

Display the final plot
plt.show()

```

----- Content of permute.cpp -----

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
#include <vector>
#include <algorithm> // For std::sort and std::unique
using namespace std;

bool hasDuplicates(const vector<int>& vec) {
 vector<int> sortedVec = vec;
 sort(sortedVec.begin(), sortedVec.end());
 auto it = unique(sortedVec.begin(), sortedVec.end());
 // If iterator 'it' is not at the end, duplicates were found.
 return it != sortedVec.end();
}

int main() {
 // Existing setup code here...
 FILE *p; p = fopen("SOLUTION_RENAME_MAX.txt", "w");
 char a[] = {/* existing characters */};
 int pin, cheque;

 cout << "Enter the size of your array: ";
 cin >> pin;
 cout << "Do you want to create a custom array? ";
 cin >> cheque;

 int array[pin]; // Assume this is filled with unique elements for permutation
 int k = pin - 1;
 int noc; // User inputs this
 scanf("%d", &noc);
 int n = noc;
 unsigned long long id = 0;
 int nbr_comb = pow(k+1, n);
}

```

```

if (cheque == 1) {
 for (int row = 0; row < nbr_comb; row++) {
 vector<int> indices;
 for (int col = n-1; col >= 0; col--) {
 int rdiv = pow(k+1, col);
 int cell = (row / rdiv) % (k+1);
 indices.push_back(array[cell]);
 }
 if (!hasDuplicates(indices)) {
 id++; fprintf(p, "%d ", (id- 1));
 for (int index : indices) {
 fprintf(p, "%c", a[index]);
 }
 fprintf(p, "\n");
 }
 }
} else {
 for (int row = 0; row < nbr_comb; row++) {
 vector<int> indices;
 for (int col = n-1; col >= 0; col--) {
 int rdiv = pow(k+1, col);
 int cell = (row / rdiv) % (k+1);
 indices.push_back(cell);
 }
 if (!hasDuplicates(indices)) {
 id++; fprintf(p, "%d ", (id- 1));
 for (int cell : indices) {
 fprintf(p, "%c", a[cell]);
 }
 fprintf(p, "\n");
 }
 }
}
fclose(p);
return 0;
}

```

----- Content of Program.py-----

```

import tkinter as tk
from tkinter import simpledialog
from tkinter import messagebox
import os
import pickle
from graphviz import Digraph

root = tk.Tk()
root.title("FSM Builder")

```

```

class State:
 def __init__(self, name):
 self.name = name

class FSM:
 def __init__(self):
 self.states = {}
 self.transitions = {}
 self.current_state = None

 def add_state(self, state_name):
 state = State(state_name)
 self.states[state_name] = state

 def add_transition(self, from_state, to_state, input_symbol):
 if from_state in self.states and to_state in self.states:
 self.transitions[(from_state, input_symbol)] = to_state
 else:
 print("State not found")

 def set_start(self, state_name):
 if state_name in self.states:
 self.current_state = state_name
 else:
 print("Start state not found")

 def transition(self, input_symbol):
 if (self.current_state, input_symbol) in self.transitions:
 self.current_state = self.transitions[(self.current_state, input_symbol)]
 else:
 print("Transition not found")

 def current_state_name(self):
 return self.current_state

 def visualize(self):
 dot = Digraph()
 for state in self.states:
 dot.node(state)
 for (from_state, input_symbol), to_state in self.transitions.items():
 label = f'{input_symbol}'
 dot.edge(from_state, to_state, label=label)
 return dot

 def load_session(fsm):
 file_name = simpledialog.askstring("Load Session", "Enter filename to load:")
 if file_name:
 try:
 with open(file_name + '.pkl', 'rb') as f:
 loaded_fsm = pickle.load(f)

```

```

fsm.states = loaded_fsm.states
fsm.transitions = loaded_fsm.transitions
fsm.current_state = loaded_fsm.current_state
except FileNotFoundError:
 messagebox.showerror("Error", "File not found")

def delete_session():
 file_name = simpledialog.askstring("Delete Session", "Enter filename to delete:")
 if file_name:
 try:
 os.remove(file_name + '.pkl')
 except FileNotFoundError:
 messagebox.showerror("Error", "File not found")

def save_session(fsm):
 file_name = simpledialog.askstring("Save Session", "Enter filename to save:")
 if file_name:
 with open(file_name + '.pkl', 'wb') as f:
 pickle.dump(fsm, f)

def display_fsm_diagram(fsm):
 dot = fsm.visualize()
 dot.format = 'png'
 dot.render('fsm_diagram', view=False)
 img = tk.PhotoImage(file='fsm_diagram.png')
 img_label = tk.Label(root, image=img)
 img_label.image = img
 img_label.pack()

Create an FSM instance
fsm = FSM()

Frame for Buttons
frame = tk.Frame(root)
root.title("Finite State Machine Builder")
frame.pack(padx=10, pady=10)

menu_bar = tk.Menu(root)
root.config(menu=menu_bar)

file_menu = tk.Menu(menu_bar, tearoff=0)
menu_bar.add_cascade(label="File", menu=file_menu)

Adding 'Save', 'Load', and 'Delete' options
file_menu.add_command(label="Save Session", command=lambda: save_session(fsm))
file_menu.add_command(label="Load Session", command=lambda: load_session(fsm))
file_menu.add_command(label="Display FSM Diagram", command=lambda: display_fsm_diagram(fsm))
file_menu.add_command(label="Delete Session", command=delete_session)

```

```

Add State Button
add_state_button = tk.Button(frame, text="Add State", command=lambda: add_state(fsm))
add_state_button.pack(side=tk.LEFT, padx=5)

Add Transition Button
add_transition_button = tk.Button(frame, text="Add Transition", command=lambda: add_transition(fsm))
add_transition_button.pack(side=tk.LEFT, padx=5)

Set Start State Button
set_start_button = tk.Button(frame, text="Set Start State", command=lambda: set_start(fsm))
set_start_button.pack(side=tk.LEFT, padx=5)

Transition Button
transition_button = tk.Button(frame, text="Transition", command=lambda: make_transition(fsm))
transition_button.pack(side=tk.LEFT, padx=5)

Current State Display
current_state_label = tk.Label(root, text="Current State: None")
current_state_label.pack(pady=10)

def add_state(fsm):
 state_name = simpledialog.askstring("Add State", "Enter state name:")
 if state_name:
 fsm.add_state(state_name)

def add_transition(fsm):
 from_state = simpledialog.askstring("Add Transition", "Enter from state:")
 to_state = simpledialog.askstring("Add Transition", "Enter to state:")
 input_symbol = simpledialog.askstring("Add Transition", "Enter input symbol:")
 if from_state and to_state and input_symbol:
 if from_state in fsm.states and to_state in fsm.states:
 fsm.add_transition(from_state, to_state, input_symbol)
 else:
 messagebox.showerror("Error", "One or more states not found")

def set_start(fsm):
 start_state = simpledialog.askstring("Set Start State", "Enter start state:")
 if start_state:
 fsm.set_start(start_state)

def make_transition(fsm):
 input_symbol = simpledialog.askstring("Transition", "Enter input symbol for transition:")
 if input_symbol:
 if (fsm.current_state, input_symbol) in fsm.transitions:
 fsm.transition(input_symbol)
 current_state_label.config(text=f"Current State: {fsm.current_state_name()}")
 else:
 messagebox.showerror("Transition Error", "Transition cannot be made with the given input symbol")

```

```
root.mainloop()
```

----- Content of pycontrol.py-----

```
import pygame
import importlib
import requests
import os
import subprocess
import webbrowser
import csv
from bs4 import BeautifulSoup
import datetime
import sys
import math
import random
import json
import re
import threading

MY_CUSTOM_EVENT = pygame.USEREVENT + 1

class PygameApp:
 def __init__(self):
 pygame.init()
 self.joystick_init()
 self.screen_size_init()
 self.commands_init()
 self.coordinates_init()
 self.images_init()
 self.input_init()
 # Additional initializations as required...

 def joystick_init(self):
 pygame.joystick.init()
 if pygame.joystick.get_count() > 0:
 self.joystick = pygame.joystick.Joystick(0)
 self.joystick.init()

 def maze_layout_init(self):
 self.maze_layout = [
 [0, 1, 2, 3],
 [4, 5, 6, 7],
 [8, 9, 10, 11],
 [12, 13, 14, 15]
]
 self.maze_width = len(self.maze_layout[0]) # Assuming all rows have the same length
 self.maze_height = len(self.maze_layout)
```

```
def screen_size_init(self):
 self.enter = 422
 self.screen_width = self.enter
 self.screen_height = self.enter
 #self.bls = 52
 self.bls = min(self.screen_width // 16, self.screen_height // 16)
 self.screen = pygame.display.set_mode((self.screen_width, self.screen_height))

def commands_init(self):
 self.bus_ = 255
 self.amount_ = (self.bus_ + 1) * (self.bus_ + 1)
 with open("Command_Template.txt", "w") as create_f:
 for line_number_ in range(self.amount_):
 create_f.write("print(\"[Empty Command Slot] (Change using a text-editor to Update this slot in\nCommand_Template.txt\")\n")

 self.star = []
 with open("Command_Template.txt", 'r') as file:
 for command in file:
 self.star.append(command.strip())

def coordinates_init(self):
 self.starsx = []
 self.starsy = []

 with open('Coordinates_Python.txt', 'r') as file:
 for line in file:
 parts = line.strip().split(",")
 if len(parts) >= 3: # Expecting at least 3 parts in each line
 try:
 # Assuming the last two parts are x and y coordinates
 x, y = map(int, parts[-2:]) # Extract the last two values
 self.starsx.append(x)
 self.starsy.append(y)
 except ValueError:
 print(f"Invalid line in coordinates file: {line}")
 else:
 print(f"Unexpected format in coordinates file: {line}")

def images_init(self):

 self.player_image = pygame.image.load("player.png") # This sets the instance attribute
 self.block_images = []
 # Calculate block size based on screen size and maze dimensions (assuming 16x16 maze)
 block_width = self.screen_width // 16
```

```
block_height = self.screen_height // 16
for i in range(256):
 # Load and scale each block image
 img = pygame.image.load(f"block{i}.png")
 scaled_img = pygame.transform.scale(img, (block_width, block_height))
 self.block_images.append(scaled_img)

def input_init(self):
 self.player_x = 0
 self.player_y = 0
 self.typei = "k"

def run(self):
 running = True
 self.maze_layout_init()
 while running:
 for event in pygame.event.get():
 if event.type == pygame.QUIT:
 running = False
 else:
 self.handle_input(event)

 self.update_game_state()
 self.render_game()

 pygame.quit()

def handle_input(self, event):
 # Delegate input handling based on typei
 if self.typei == 'k':
 self.handle_keyboard_input(event)
 elif self.typei == 'd':
 self.handle_xbox_controller_input()
 elif self.typei == 's':
 self.handle_switch_command_input()
 elif self.typei == 'm':
 self.handle_mouse_input()

 if event.type == MY_CUSTOM_EVENT:
 self.handle_custom_event(event)

def handle_keyboard_input(self, event):
 # Get the current state of all keys
 keys = pygame.key.get_pressed()

 # Update move flag based on keyboard input
 move = keys[pygame.K_LEFT] or keys[pygame.K_RIGHT] or keys[pygame.K_UP] or
keys[pygame.K_DOWN]
```

```
Update player position based on keyboard input
if move:
 if keys[pygame.K_LEFT]:
 self.player_x -= 1
 if keys[pygame.K_RIGHT]:
 self.player_x += 1
 if keys[pygame.K_UP]:
 self.player_y -= 1
 if keys[pygame.K_DOWN]:
 self.player_y += 1
 self.clamp_player_position()

def clamp_player_position(self):
 # Ensure the player position stays within bounds
 self.player_x = max(0, min(self.player_x, self.screen_width // self.bls - 1))
 self.player_y = max(0, min(self.player_y, self.screen_height // self.bls - 1))

def handle_xbox_controller_input(self):
 # Xbox controller input handling code
 # ... (your Xbox controller handling code here)

 if(self.typei == 'd'):
 # Check for Xbox controller input
 if pygame.joystick.get_count() > 0:
 joystick = pygame.joystick.Joystick(0)
 joystick.init()
 # Check for D-Pad input
 dpad = joystick.get_hat(0)
 if dpad == (1, 0):
 self.player_x += 1
 elif dpad == (-1, 0):
 self.player_x -= 1
 elif dpad == (0, 1):
 self.player_y -= 1
 elif dpad == (0, -1):
 self.player_y += 1
 if self.player_x < 0:
 self.player_x = 0
 if self.player_x >= int(self.screen_width / self.bls):
 self.player_x = int(self.screen_width / self.bls) - 1
 if self.player_y < 0:
 self.player_y = 0
 if self.player_y >= int(self.screen_height / self.bls):
 self.player_y = int(self.screen_height / self.bls) - 1
```

```

def handle_switch_command_input(self):
 # Switch command matrix input handling code
 # ... (your switch command handling code here)

 #LoadCommands
 self.xin = 0
 filnam = input("Enter name of Command Matrix file, 'filename.txt': ")
 quet = int(input("Enter 1 if file already exists [Read mode], Else Enter 2 [Write mode], choose carefully:
"))
 self.bus = 255
 self.amount = (self.bus + 1) * (self.bus + 1)
 self.line_number = 0
 #ai = log10(amount)
 if(quet == 2):
 create_f = open(filnam, "w")
 while(line_number < amount):
 #i = log10(line_number)
 create_f.write("print(\"[Empty Command Slot] Change using a text-editor to Update this slot in
Command_Template.txt after renaming it accordingly.\")\n")
 self.line_number = self.line_number + 1

 create_f.close()

 with open(filnam, 'r') as file:
 self.commands = file.readlines()
 self.xin = self.xin + 1

 self.star = [0]

 for i, command in enumerate(self.commands):
 self.commands[i] = command.strip()
 #cmd = command.split(",")
 self.star.append(command)
 del self.star[0]

 #reset state of matrix
 self.typepi = "k"
 self.player_x = 0
 self.player_y = 0

def handle_mouse_input(self):
 # Mouse input handling code
 # ... (your mouse handling code here)

 if(self.typepi == 'm'):
 # Get mouse position

```

```

 self.mouse_x, self.mouse_y = pygame.mouse.get_pos()

 # Update player position based on mouse position
 self.player_x = self.mouse_x // self.bls
 self.player_y = self.mouse_y // self.bls

def update_game_state(self):
 # Update game state (e.g., player position)
 # ...
 #reset state of matrix
 self.typepi = "k"
 self.player_x = 0
 self.player_y = 0

 """

def render_game(self):
 self.screen.fill((255, 255, 255))

 # Ensure the loop covers the entire range
 for y in range(int((self.maze_height*52)/self.bls)):
 for x in range(int((self.maze_width*52)/self.bls)):
 relative_x = x - self.player_x
 relative_y = y - self.player_y
 block_index = relative_x + relative_y * self.maze_width
 if 0 <= block_index < len(self.block_images):
 self.screen.blit(self.block_images[block_index], (x * self.bls, y * self.bls))

 self.screen.blit(self.player_image, (self.player_x * self.bls, self.player_y * self.bls))
 pygame.display.update()
 """

def render_game(self):
 # Clear screen
 self.screen.fill((255, 255, 255))

 # Draw blocks
 for y in range(16): # Assuming a 16x16 grid
 for x in range(16):
 block_index = y * 16 + x # Calculate the block index assuming a sequential layout
 block_image = self.block_images[block_index] # Get the correct image
 # Ensure the image is drawn at the correct position; no overlapping
 self.screen.blit(block_image, (x * self.bls, y * self.bls))

 # Draw player image at the player's position
 self.screen.blit(self.player_image, (self.player_x * self.bls, self.player_y * self.bls))

 # Update the display
 pygame.display.update()

def handle_spacebar_event(self):

```

```

self.typei = "n"
enter_cma = int(input("Enter command? 1[yes], 0 [no]: "))
if enter_cma == 0:
 self.typei = "k"
 self.player_x, self.player_y = 0, 0
elif enter_cma == 1:
 input_variable = int(input("Enter command number ID (0 to 65535): "))
 pygame.event.post(pygame.event.Event(MY_CUSTOM_EVENT, {"input_variable": input_variable}))

def handle_custom_event(self, event):
 cma = event.input_variable
 if self.typei == 'n':
 if 0 <= cma <= 65535:
 try:
 # Execute the command from the star list
 command_to_execute = self.star[cma]
 exec(command_to_execute, globals(), locals())
 except IndexError:
 print(f"Command ID {cma} is out of range.")
 except Exception as e:
 print(f"An error occurred executing command {cma}: {e}")
 else:
 print(f"Invalid command number ID: {cma}")

 def start(self):
 self.thread = threading.Thread(target=self.run)
 self.thread.start()

 def stop(self):
 # Send a QUIT event to stop the game loop
 pygame.event.post(pygame.event.Event(pygame.QUIT))

if __name__ == "__main__":
 app = PygameApp()
 app.start()

```

----- Content of pyout.py-----

```

def read_characters_from_file(file_path):
 with open(file_path, 'r', encoding='utf-8') as file:
 # Read the contents of the file and split by lines
 characters = file.read().splitlines()
 return characters

def generate_strings(char_set, initial_cells, num_file_systems, output_file_path):
 with open(output_file_path, "w", encoding='utf-8') as output_file:
 k = len(char_set)
 for cell_count in range(initial_cells, initial_cells + num_file_systems):
 nof = pow(k, cell_count)

```

```

for i in range(nof):
 s = ""
 for col in range(cell_count - 1, -1, -1):
 s += char_set[(i // pow(k, col)) % k]
 output_file.write(s + '\n')

def main():
 char_file_path = 'charSF3.txt' # Replace with your actual file path
 output_file_path = 'pyout.txt'

 char_set = read_characters_from_file(char_file_path)

 try:
 initial_cells = int(input('Initial number of cells per file: '))
 num_file_systems = int(input('Number of File Systems to be generated (Starting from c Celled Files) + 1: '))
 except ValueError:
 print("Invalid input. Please enter integers only.")
 return

 generate_strings(char_set, initial_cells, num_file_systems, output_file_path)
 print("Files generated successfully and saved to pyout.txt.")

if __name__ == "__main__":
 main()

```

----- Content of pytoexe\_manual.py-----

```

import tkinter as tk
from tkinter import ttk
from tkinter import filedialog
import subprocess
import threading

def select_files():
 file_paths = filedialog.askopenfilenames(filetypes=[("Python files", "*.py")])
 selected_files.set(", ".join(file_paths))

def compile_to_exe():
 files = selected_files.get().split(", ")
 one_file_option = single_exe.get()

 for file in files:
 with open(file, 'r') as f:
 content = f.read()

 if 'import tkinter' in content or 'from tkinter import' in content:
 hidden_import_option = '--hidden-import=_tkinter'
 else:
 hidden_import_option = ""

 # Rest of the compilation logic...

```

```
if one_file_option:
 command = f"pyinstaller --onefile {hidden_import_option} {file}"
else:
 command = f"pyinstaller {hidden_import_option} {file}"

subprocess.run(command, shell=True)

output_message.set("Compilation completed.")

root = tk.Tk()
root.title("Python to EXE Converter")

frame = tk.Frame(root, padx=10, pady=10)
frame.pack(padx=20, pady=20)

selected_files = tk.StringVar()
single_exe = tk.BooleanVar()
output_message = tk.StringVar()

select_button = tk.Button(frame, text="Select Python Files", command=select_files)
select_button.grid(row=0, column=0, padx=10, pady=10)

single_exe_checkbox = tk.Checkbutton(frame, text="Compile to single EXE", variable=single_exe)
single_exe_checkbox.grid(row=1, column=0, padx=10, pady=10)

compile_button = tk.Button(frame, text="Compile", command=lambda:
threading.Thread(target=compile_to_exe).start())
compile_button.grid(row=2, column=0, padx=10, pady=10)

output_label = tk.Label(frame, textvariable=output_message)
output_label.grid(row=3, column=0, padx=10, pady=10)

root.mainloop()

----- Content of pytoexe_manual_.py-----
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog
import subprocess
import threading

def select_files():
 file_paths = filedialog.askopenfilenames(filetypes=[("Python files", "*.py")])
 selected_files.set(", ".join(file_paths))

def compile_to_exe():
 files = selected_files.get().split(", ")
```

```

one_file_option = single_exe.get()

for file in files:
 with open(file, 'r') as f:
 content = f.read()

 if 'import tkinter' in content or 'from tkinter import' in content:
 hidden_import_option = '--hidden-import=_tkinter'
 else:
 hidden_import_option = ""

 if one_file_option:
 command = f"pyinstaller --onefile {hidden_import_option} {file}"
 else:
 command = f"pyinstaller {hidden_import_option} {file}"

 subprocess.run(command, shell=True)

output_message.set("Compilation completed.")

root = tk.Tk()
root.title("Python to EXE Converter")

frame = tk.Frame(root, padx=10, pady=10)
frame.pack(padx=20, pady=20)

selected_files = tk.StringVar()
single_exe = tk.BooleanVar()
output_message = tk.StringVar()

select_button = tk.Button(frame, text="Select Python Files", command=select_files)
select_button.grid(row=0, column=0, padx=10, pady=10)

single_exe_checkbox = tk.Checkbutton(frame, text="Compile to single EXE", variable=single_exe)
single_exe_checkbox.grid(row=1, column=0, padx=10, pady=10)

compile_button = tk.Button(frame, text="Compile", command=lambda:
 threading.Thread(target=compile_to_exe).start())
compile_button.grid(row=2, column=0, padx=10, pady=10)

output_label = tk.Label(frame, textvariable=output_message)
output_label.grid(row=3, column=0, padx=10, pady=10)

root.mainloop()

```

----- Content of pywindows.py-----

```

import tkinter as tk
from tkinter import scrolledtext, ttk

```

```
import subprocess
import os
import platform
import shlex

class CustomTerminal(tk.Tk):
 def __init__(self):
 super().__init__()

 self.title("Python Terminal")
 self.geometry("1000x700")

 self.command_history = []
 self.history_pointer = 0
 self.current_dir = os.getcwd()
 self.os_type = platform.system()

 self.menu_bar = tk.Menu(self)
 self.file_menu = tk.Menu(self.menu_bar, tearoff=0)
 self.file_menu.add_command(label="New Tab", command=self.add_tab, accelerator="Ctrl+T")
 self.file_menu.add_command(label="Exit", command=self.on_closing)
 self.menu_bar.add_cascade(label="File", menu=self.file_menu)
 self.config(menu=self.menu_bar)

 self.notebook = ttk.Notebook(self)
 self.notebook.pack(expand=True, fill='both')

 self.status_bar = tk.Label(self, text=f"Current directory: {self.current_dir}", bd=1, relief=tk.SUNKEN,
 anchor=tk.W)
 self.status_bar.pack(side=tk.BOTTOM, fill=tk.X)

 self.add_tab()
 self.bind("<Control-t>", lambda event: self.add_tab())

 self.protocol("WM_DELETE_WINDOW", self.on_closing)

 def add_tab(self):
 tab = ttk.Frame(self.notebook)
 text_widget = scrolledtext.ScrolledText(tab, wrap='word', state=tk.DISABLED)
 text_widget.pack(expand=True, fill='both')
 command_entry = tk.Entry(tab, bd=2, relief=tk.SUNKEN)
 command_entry.pack(fill='x', ipady=4)
 command_entry.bind('<Return>', self.execute_command)
 command_entry.bind('<Up>', self.prev_command)
 command_entry.bind('<Down>', self.next_command)

 tab.text_widget = text_widget

 self.notebook.add(tab, text=f"Terminal {self.notebook.index('end')+1}")
 self.notebook.select(tab)
```

```
command_entry.focus()

def execute_command(self, event):
 command_entry = event.widget
 command = command_entry.get()
 if not command.strip():
 return

 if self.os_type == "Windows":
 cmd = ['cmd.exe', '/c', command]
 else:
 cmd = ['/bin/bash', '-c', command]

 try:
 process = subprocess.Popen(cmd, cwd=self.current_dir, stdout=subprocess.PIPE,
 stderr=subprocess.PIPE, text=True, shell=True)
 stdout, stderr = process.communicate()
 output = stdout + stderr

 if command.startswith('cd'):
 parts = shlex.split(command)
 if len(parts) > 1:
 path = parts[1]
 new_dir = os.path.abspath(os.path.join(self.current_dir, path))
 if os.path.isdir(new_dir):
 self.current_dir = new_dir
 self.status_bar.config(text=f"Current directory: {self.current_dir}")
 else:
 output = f"No such directory: {path}\n"

 except Exception as e:
 output = f"Error: {e}\n"

 self.command_history.append(command)
 self.history_pointer = len(self.command_history)
 command_entry.delete(0, tk.END)
 tab = self.notebook.nametowidget(self.notebook.select())
 text_widget = tab.text_widget
 text_widget.config(state=tk.NORMAL)
 text_widget.insert(tk.END, f'{self.current_dir}> {command}\n{output}\n')
 text_widget.see(tk.END)
 text_widget.config(state=tk.DISABLED)

def prev_command(self, event):
 command_entry = event.widget
 if self.command_history and self.history_pointer > 0:
 self.history_pointer -= 1
 command_entry.delete(0, tk.END)
 command_entry.insert(tk.END, self.command_history[self.history_pointer])
```

```
def next_command(self, event):
 command_entry = event.widget
 if self.command_history and self.history_pointer < len(self.command_history):
 command_entry.delete(0, tk.END)
 command_entry.insert(tk.END, self.command_history[self.history_pointer])
 self.history_pointer += 1
```

```
def on_closing(self):
 self.destroy()
```

```
if __name__ == "__main__":
 terminal = CustomTerminal()
 terminal.mainloop()
```

----- Content of random\_pictography.py-----

```
from tkinter import *
import tkinter as tk
import random
import os
import time

import subprocess

import io #experiment
import pyautogui
from PIL import Image, experiment

print("random() : ", random.random())
master = Tk()
master.attributes('-fullscreen', True)
#a = 250
#b = 200
print("Welcome\n")
print("Tip: side length should be a factor of the image width and of the image height")
change1 = input("Enter side length of image block: ")
change = int(change1)
#a = 1880
#b = 1050
a1 = input("Enter width of image: ")
a = int(a1)
b1 = input ("Enter height of image: ")
b = int(b1)
pin_p = a/change
w = Canvas(master, width= a, height= b)

#files = 3
files_buffer = input("Enter number of files to be generated: ")

```

```

files = int(files_buffer)

c = ["white","gold","black"]
xc = 0
zerox = 0
zeroy = 0
p = 1
range_for = int((a/change)*(b/change))
name = 1

for t in range(0,files):
 switch = 1
 for x in range(range_for):
 #f = open('%s.ps' % name, 'wb')
 #f.close
 if(switch == 1):
 row = 1
 nxleft = 0
 nxright = change
 nyleft = 0
 nyright = change
 zerox = 0
 zeroy = 0
 c_length = len(c)
 switch = 0
 ran = random.randint(0,c_length- 1)
 w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[ran], outline = "black")
 #w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[ran])
 #zxbuffer = zerox
 #w.place(x = zerox, y = zeroy)
 #w.place(bordemode = OUTSIDE, x = zerox + change, y = zeroy)
 w.grid(row = zeroy, column = zerox + change)
 if(p >= pin_p and p%pin_p == 0):
 zeroy = zeroy + change
 zerox =-change
 #zerox = 0
 nxleft = change
 nxright = 0
 nyleft = nyleft + change
 nyright = nyright + change
 zerox = zerox + change
 p = p + 1
 #xc + 1
 nxright = nxright + change
 if(xc == 3):
 xc = 0

ce = str(name)

```

```

w.update()
w.postscript(file = ce + ".ps", colormode='color')
name = name + 1
#f_p = 'D:\\Kaliber\\Portfolio\\Content\\My PhotoBook\\Content\\'
#os.rename(f_p + '1.ps', f_p + '2.ps')

#process = subprocess.Popen(["ps2pdf", ce + ".ps", ce + ".pdf"], shell=True)

#Contender 1 For File Saving
"""
def savefirst():
 cnv = getscreen().getcanvas()
 global hen
 ps = cnv.postscript(colormode = 'color')
 hen = filedialog.asksaveasfilename(defaultextension = '.jpg')
 im = Image.open(io.BytesIO(ps.encode('utf-8')))
 im.save(hen + '.jpg')
#savefirst()
"""

Second Contender For File Saving
def save(w):
 ps = w.canvas.postscript(colormode='color')
 img = Image.open(io.BytesIO(ps.encode('utf-8')))
 img.save('testing.jpg')
"""

#w.update()
#script = ce + ".ps"
#w.save(script)
#os.startfile(script)
#print(script)
print("Done")

#master.mainloop()
time.sleep(7)
master.destroy()

----- Content of rbtoexe.cpp-----
#include <iostream>
#include <cstdlib>
#include <filesystem>

int main() {

```

```

// Get the current working directory
std::filesystem::path current_path = std::filesystem::current_path();

// Iterate through each file in the current directory
for (const auto& entry : std::filesystem::directory_iterator(current_path)) {
 std::filesystem::path file_path = entry.path();

 // Check if the file has a .rb extension
 if (file_path.extension() == ".rb") {
 std::string run_command = "ruby " + file_path.string();

 // Run the .rb file
 std::cout << "Running: " << file_path.string() << std::endl;
 std::system(run_command.c_str());
 }
}

return 0;
}

```

----- Content of reconvert.cpp-----

```

#include <iostream>
#include <fstream>
#include <vector>
#include <unordered_map>
#include <sstream>

using namespace std;

vector<unordered_map<string, string>> readMappings() {
 vector<unordered_map<string, string>> mappings(4);
 for (int i = 1; i <= 4; ++i) {
 ifstream file(to_string(i) + ".txt");
 if (!file) {
 cerr << "Error opening file " << i << ".txt" << endl;
 exit(EXIT_FAILURE);
 }
 string line;
 while (getline(file, line)) {
 int spacePos = line.find(' ');
 if (spacePos != string::npos) {
 string numberStr = line.substr(0, spacePos);
 string str = line.substr(spacePos + 1);
 mappings[i - 1][numberStr] = str;
 }
 }
 }
 return mappings;
}

```

```

int main() {
 auto mappings = readMappings();

 string inputFilePath;
 cout << "Enter the path of the file to re-convert: ";
 cin >> inputFilePath;

 ifstream inputFile(inputFilePath);
 if (!inputFile) {
 cerr << "Could not open the file " << inputFilePath << endl;
 exit(EXIT_FAILURE);
 }

 string line, content;
 while (getline(inputFile, line)) {
 content += line + '\n';
 }
 inputFile.close();

 stringstream ss(content);
 string token;
 string outputContent;
 while (getline(ss, token, ',')) {
 int dotPos = token.find('.');
 if (dotPos != string::npos) {
 int len = stoi(token.substr(0, dotPos));
 string numberStr = token.substr(dotPos + 1);
 if (numberStr == "27" && len == 1) {
 outputContent += '\n';
 } else if (numberStr != "0") {
 numberStr = to_string(stoi(numberStr) - 1);
 if (mappings[len - 1].find(numberStr) != mappings[len - 1].end()) {
 outputContent += mappings[len - 1][numberStr];
 } else {
 outputContent += "?"; // Placeholder for unmapped
values
 }
 }
 }
 }

 ofstream outputFile("0.txt");
 if (outputFile) {
 outputFile << outputContent;
 outputFile.close();
 } else {
 cerr << "Could not create the file 0.txt" << endl;
 }
}

```

```
 return 0;
}
```

----- Content of Renderer2.py-----

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
import math

def plot_equation(equation_type, equation, domain=(-10, 10)):
 if equation_type == "2D":
 plot_2d_curve(equation, domain)
 elif equation_type == "3D":
 plot_3d_surface(equation, domain)
 elif equation_type == "implicit":
 plot_implicit(equation, domain)
 elif equation_type == "parametric":
 plot_parametric(equation, domain)
 elif equation_type == "polar":
 plot_polar(equation, domain)

def plot_2d_curve(equation, domain):
 x = np.linspace(domain[0], domain[1], 400)
 y = eval(equation)
 plt.plot(x, y)
 plt.xlabel('X-axis')
 plt.ylabel('Y-axis')
 plt.title('2D Representation of Equation')
 plt.grid(True)
 plt.show()

def plot_3d_surface(equation, domain):
 x, y = sp.symbols('x y')
 X = np.linspace(domain[0], domain[1], 100)
 Y = np.linspace(domain[0], domain[1], 100)
 X, Y = np.meshgrid(X, Y)

 Z_positive = np.zeros_like(X)
 Z_negative = np.zeros_like(X)
 for i in range(X.shape[0]):
 for j in range(X.shape[1]):
 try:
 Z_positive[i, j] = eval(equation, {"x": X[i, j], "y": Y[i, j], "math": math})
 Z_negative[i, j] = -Z_positive[i, j]
 except ValueError:
 Z_positive[i, j] = np.nan
 Z_negative[i, j] = np.nan
```

```

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z_positive, cmap='viridis')
ax.plot_surface(X, Y, Z_negative, cmap='viridis')
plt.show()

def plot_implicit(equation, domain):
 x, y = sp.symbols('x y')
 implicit_eq = sp.Eq(*map(sp.sympify, equation.split('=')))
 p1 = sp.plot_implicit(implicit_eq, (x, domain[0], domain[1]), (y, domain[0], domain[1]), show=False)
 p1.show()

def plot_parametric(equations, domain):
 t = np.linspace(domain[0], domain[1], 400)
 x_eq, y_eq = equations.split(';')
 x = eval(x_eq.split('=')[1])
 y = eval(y_eq.split('=')[1])
 plt.plot(x, y)
 plt.xlabel('X-axis')
 plt.ylabel('Y-axis')
 plt.title('Parametric Representation')
 plt.grid(True)
 plt.show()

def plot_polar(equation, domain):
 theta = np.linspace(domain[0], domain[1], 400)
 r_eq = equation.split('=')[1]
 r = eval(r_eq)
 plt.polar(theta, r)
 plt.title('Polar Representation')
 plt.grid(True)
 plt.show()

equation_type = input("Enter equation type (2D, 3D, implicit, parametric, polar): ")

if equation_type == "parametric":
 print("For parametric equations, please enter in the format x(t)=...;y(t)=...")
 equation_x = input("Enter your x(t) equation: ")
 equation_y = input("Enter your y(t) equation: ")
 equation = equation_x + ';' + equation_y
else:
 equation = input("Enter your equation: ")

domain_start = float(input("Enter domain start: "))
domain_end = float(input("Enter domain end: "))
domain = (domain_start, domain_end)

plot_equation(equation_type, equation, domain)

```

----- Content of Sandbox Database Kaliber.c -----

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <limits.h>

//nof = number of files
//fn = file number
//noftbc = number of files to be created
//ftd = file to delete
//ftbv = file to be viewed
```

```
int main(){
```

```

printf("\n\n\tWelcome To The Sandbox Database Kaliber\n\n");
printf("\n\t\tt1 = yes\n\n");
printf("\n\t\tt2 = no\n\n");

int prompt1;
printf("\nDo you want to create files?: ");
scanf("%d", &prompt1);
printf("\n\nEnter file number type to be created after entering the number of files to be
created.");
//printf("\n\n");
if(prompt1 == 1){
 int noftbc;
 int nof;
 int i;
 int ftid;
 int set_ftid;
 printf("\nEnter the number of files to be created, then the file number type: ");
 scanf("%d", &noftbc);
 printf("_");
 nof = noftbc;
 scanf("%d", &ftid);
 set_ftid = ftid;
 int fid = 0;
 for(i = 0; i < nof; i = i + 1){
 char fname[500];
 sprintf(fname, "%d_%d.txt", set_ftid, fid);
 FILE *p; p = fopen(fname, "w");
 fclose(p);
 //ftid = ftid + 1;
 fid = fid + 1;
 }
} else{
 return 0;
}

/*
int prompt_n;
printf("\nQuestion?: ");
scanf("d%", &prompt_n);
printf("\n\n");
if(prompt_n == 1 && prompt_n != 2){
 instruction statement/s;
}
*/
return 0;
}

```

----- Content of se.cpp-----



```

k = pin- 1;
printf("\n\tk = %d", k);
int noc; printf("\n\tm = ");
scanf("%d", &noc);
printf("\n\tNumber Of FILE Cells = %d", noc);
int n = noc;
int row, col;
int cell;
int rdiv;
unsigned long long id;
id = 0;
int rin;
int nbr_comb = pow(k+1, n);
if(cheque == 1){
 for (row=0; row < nbr_comb; row++){
 id++; fprintf(p,"%d ", (id- 1));
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);
 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 fprintf(p,"%c", a[rin]);
 }
 fprintf(p,"\n");
 //printf("\n");
 }
}
if(cheque == 0){
 for (row=0; row < nbr_comb; row++){
 id++; fprintf(p,"%d ", (id- 1));
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 cell = (row/rdiv) % (k+1);
 fprintf(p,"%c", a[cell]);
 }
 fprintf(p,"\n");
 //printf("\n");
 }
}
//fprintf(p,"\n\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
fclose(p);
//end of adaptation
return 0;
}

```

----- Content of SequentialWriterVersion3.c-----

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

```



```

 printf("\n");
 }
}

//Get One Specific File From Database and save to active Directory
if(assert == 1){

 printf("\n\tPlease Enter a Value For n:\t");
 scanf("%d", &noc);
 int n = noc;
 int nbr_comb = pow(k+1, n);
 printf("\n\n\tPlease Enter file number wanted between 0 and %d Inclusively:\t",
nbr_comb-1);
 scanf("%d", &fnum);
 printf("\n\tNumber Of FILE Cells To Be Generated = %d", noc);
 int row, col; int cell; int rdiv; int id; id = fnum;

 int sw = 1;
 for (row=0; row < nbr_comb; row++){
 if(row == fnum){
 char filename[500];
 sprintf(filename, "DQUERY%dC%d.txt", n,id);
 //FILE *p; p = fopen("SOLUTION.txt","w");
 FILE *p; p = fopen(filename,"w");
 id++;
 //fprintf(p, "\n\n\nFILE%d\n\n\n", id);
 for (col=n-1; col>=0; col--){
 rdiv = pow(k+1, col); cell = (row/rdiv) % (k+1);
 fprintf(p,"%c", a[cell]);
 sw = 0;
 }
 if(sw == 0)
 break;
 }
 }
}

if(assert == 0 || assert == 3){
 puts("exiting");
}
return 0;
}

```

----- Content of sequential\_pictographyV10.py-----  
from tkinter import \*

```
import tkinter as tk
import random
import os
import math

import subprocess

import io #experiment
import pyautogui
#from PIL import Image, experiment

print("random() : ", random.random())
master = Tk()
master.attributes('-fullscreen', True)
#a = 250
#b = 200
print("Welcome\n")
print("Tip: side length should be a factor of the image width and of the image height")
change1 = input("Enter side length of image block: ")
change = int(change1)
#a = 1880
#b = 1050
a1 = input("Enter width of image: ")
a = int(a1)
b1 = input ("Enter height of image: ")
b = int(b1)
pin_p = a/change
w = Canvas(master, width= a, height= b)

#files = 3
#files_buffer = input("Enter number of files to be generated (8^((width/side_length)*(height/side_length))): ")
#)
#files = int(files_buffer)

#c = ["red","blue","yellow","brown","purple","pink","green","orange"]

#c = ["red","blue","yellow","brown","purple","pink","green","orange","grey","white"]

c = ["white", "black", "green", "red", "blue", "yellow"]
xc = 0
zerox = 0
zeroy = 0
p = 1
range_for = int((a/change)*(b/change))
name = 1
cells = ((a//change)*(b//change))
upper = cells- 1
nbr_comb = math.pow(len(c),cells)
```

```

files = int(nbr_comb)
#print(len(c))
rown = 0
#rdiv = math.pow(len(c),cells- 1)
#print(rdiv)
#cell = (row/rdiv) % (len(c))
#print(cell)

#rdiv = math.pow(len(c),cells- 2)
#print(rdiv)
#cell = (row/rdiv) % (len(c))
#print(cell)

#define rone(h,j,l):
cell = (h/j) % l
celled = int(cell)
w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = c[celled])
file_count = 0
for t in range(0,files):
 file_count = file_count + 1
 #if(file_count == 121):
 # break
 switch = 1
 sw = 0
 #for x in range(range_for):
 #for x in range (0,cells):
 x = 0
 col = cells- 1
 #print(rown)
 while(x < cells):

 #f = open('%s.ps' % name, 'wb')
 #f.close
 if(switch == 1):
 row = 1
 nxleft = 0
 nxright = change
 nyleft = 0
 nyright = change
 zerox = 0
 zeroy = 0
 c_length = len(c)
 switch = 0
 #ran = random.randint(0,c_length- 1)

 rdiv = math.pow(len(c),col)
 cell = (rown/rdiv) % (len(c))
 #rone(row,rdiv,len(c))
 #print(cell)
 #print(rdiv)
 celled = int(cell)

```

```

print(celled)
#print(cell)
#w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[ran], outline = c[ran])
#w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = c[celled], width = 0)
w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = "black", width = 0)
if(x <= cells):
 col = col- 1
#w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[ran])
#zxbuffer = zerox
#w.place(x = zerox, y = zeroy)
#w.place(bordemode = OUTSIDE, x = zerox + change, y = zeroy)
w.grid(row = zeroy, column = zerox + change)
if(p >= pin_p and p%pin_p == 0):
 zeroy = zeroy + change
 zerox =-change
 #zerox = 0
 nxleft = change
 nxright = 0
 nyleft = nyleft + change
 nyright = nyright + change
 zerox = zerox + change
 p = p + 1
 #xc + 1
 nxright = nxright + change
 if(xc == 3):
 xc = 0
 x = x + 1
 rown = rown + 1
 ce = str(name)
 w.update()
 w.postscript(file = ce + ".ps", colormode='color')
 name = name + 1
 #f_p = 'D:\\Kaliber\\Portfolio\\Content\\My PhotoBook\\Content\\'
 #os.rename(f_p + '1.ps', f_p + '2.ps')

#process = subprocess.Popen(["ps2pdf", ce + ".ps", ce + ".pdf"], shell=True)

```

```

#Contender 1 For File Saving
#####
def savefirst():
 cnv = getscreen().getcanvas()
 global hen
 ps = cnv.postscript(colormode = 'color')
 hen = filedialog.asksaveasfilename(defaultextension = '.jpg')
 im = Image.open(io.BytesIO(ps.encode('utf-8')))
 im.save(hen + '.jpg')

```

```
#savefirst()
"""
"""

Second Contender For File Saving
def save(w):
 ps = w.canvas.postscript(colormode='color')
 img = Image.open(io.BytesIO(ps.encode('utf-8')))
 img.save('testing.jpg')
"""

#w.update()
#script = ce + ".ps"
#w.save(script)
#os.startfile(script)
#print(script)
print("Done")

#master.mainloop()
```

```
----- Content of sequential_pictographyV11.py-----
from tkinter import *
import tkinter as tk
import random
import os
import math

import subprocess

import io #experiment
#import pyautogui
#from PIL import Image, experiment

print("random() : ", random.random())
master = Tk()
master.attributes('-fullscreen', True)
#a = 250
#b = 200
print("Welcome\n")
print("Tip: side length should be a factor of the image width and of the image height")
change1 = input("Enter side length of image block: ")
change = int(change1)
#a = 1880
#b = 1050
a1 = input("Enter width of image: ")
a = int(a1)
b1 = input ("Enter height of image: ")
b = int(b1)
pin_p = a/change
```

```
w = Canvas(master, width= a, height= b)

c = ["red", "green", "blue"]
xc = 0
zerox = 0
zeroy = 0
p = 1
range_for = int((a/change)*(b/change))
name = 1
cells = ((a//change)*(b//change))
upper = cells- 1
nbr_comb = math.pow(len(c),cells)
files = int(nbr_comb)
#print(len(c))
rown = 0

file_count = 0
for t in range(0,files):
 file_count = file_count + 1
 #if(file_count == 121):
 # break
 switch = 1
 sw = 0
 #for x in range(range_for):
 #for x in range (0,cells):
 x = 0
 col = cells- 1
 #print(rown)
 while(x < cells):

 #f = open('%s.ps' % name, 'wb')
 #f.close
 if(switch == 1):
 row = 1
 nxleft = 0
 nxright = change
 nyleft = 0
 nyright = change
 zerox = 0
 zeroy = 0
 c_length = len(c)
 switch = 0
 #ran = random.randint(0,c_length- 1)

 rdiv = math.pow(len(c),col)
 cell = (rown/rdiv) % (len(c))

 celled = int(cell)
 print(celled)
```

```

w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = "black", width = 0)
if(x <= cells):
 col = col- 1

w.grid(row = zeroy, column = zerox + change)
if(p >= pin_p and p%pin_p == 0):
 zeroy = zeroy + change
 zerox =-change
 #zerox = 0
 nxleft = change
 nxright = 0
 nyleft = nyleft + change
 nyright = nyright + change
 zerox = zerox + change
p = p + 1
#xc + 1
nxright = nxright + change
if(xc == 3):
 xc = 0
 x = x + 1
rown = rown + 1
ce = str(name)
w.update()
#w.postscript(file = ce + ".ps", colormode='color')
w.postscript(file=ce + ".ps", colormode='color', x=0, y=0, width=a, height=b)
name = name + 1

print("Done")

#master.mainloop()

```

----- Content of Signal Simulator.py-----

```

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def connect_particles(particle_a, particle_b):
 # Connects the two particles with a line segment
 fig = plt.figure()
 ax = fig.add_subplot(111, projection='3d')
 ax.plot([particle_a[0], particle_b[0]], [particle_a[1], particle_b[1]], [particle_a[2], particle_b[2]])
 plt.show()

def disconnect_particles(particle_a, particle_b):
 # Disconnects the two particles by removing the line segment connecting them
 pass

```

```
def move_forward(particle_a, particle_b, a, b):
 # Moves the particles forward by a specified amount
 particle_a += a
 particle_b += b
 return particle_a, particle_b

def move_backward(particle_a, particle_b, c, d):
 # Moves the particles backward by a specified amount
 particle_a -= c
 particle_b -= d
 return particle_a, particle_b

def move_leftward(particle_a, particle_b, e, f):
 # Moves the particles leftward by a specified amount
 particle_a[0] -= e
 particle_b[0] -= f
 return particle_a, particle_b

def move_rightward(particle_a, particle_b, g, h):
 # Moves the particles rightward by a specified amount
 particle_a[0] += g
 particle_b[0] += h
 return particle_a, particle_b

def move_upward(particle_a, particle_b, i, j):
 # Moves the particles upward by a specified amount
 particle_a[1] += i
 particle_b[1] += j
 return particle_a, particle_b

def move_downward(particle_a, particle_b, k, l):
 # Moves the particles downward by a specified amount
 particle_a[1] -= k
 particle_b[1] -= l
 return particle_a, particle_b

def save_gif(coordinates):
 # Saves the sequence of coordinates as a GIF file
 pass

def main():
```

```

particle_a = np.array([0, 0, 0]) # Initial position of particle A
particle_b = np.array([1, 1, 1]) # Initial position of particle B
coordinates = [particle_a.copy(), particle_b.copy()] # Stores the sequence of coordinates

while True:
 print("Choose an option:")
 print("0. Connect the two particles")
 print("1. Disconnect the two particles")
 print("2. Move forward")
 print("3. Move backward")
 print("4. Move leftward")
 print("5. Move rightward")
 print("6. Move upward")
 print("7. Move downward")
 print("8. End program and save output as a GIF")

 option = int(input("Enter your option: "))

 if option == 0:
 connect_particles(particle_a, particle_b)
 elif option == 1:
 disconnect_particles(particle_a, particle_b)
 elif option == 2:
 a = int(input("Enter the amount to move particle A forward: "))
 b = int(input("Enter the amount to move particle B forward: "))
 particle_a, particle_b = move_forward(particle_a, particle_b, a, b)
 elif option == 3:
 c = int(input("Enter the amount to move particle A backward: "))
 d = int(input("Enter the amount to move particle B backward: "))
 particle_a, particle_b = move_backward(particle_a, particle_b, c, d)
 elif option == 4:
 e = int(input("Enter the amount to move particle A leftward: "))
 f = int(input("Enter the amount to move particle B leftward: "))
 particle_a, particle_b = move_leftward(particle_a, particle_b, e, f)
 elif option == 5:
 g = int(input("Enter the amount to move particle A rightward: "))
 h = int(input("Enter the amount to move particle B rightward: "))
 particle_a, particle_b = move_rightward(particle_a, particle_b, g, h)
 elif option == 6:
 i = int(input("Enter the amount to move particle A upward: "))
 j = int(input("Enter the amount to move particle B upward: "))
 particle_a, particle_b = move_upward(particle_a, particle_b, i, j)
 elif option == 7:
 k = int(input("Enter the amount to move particle A downward: "))
 l = int(input("Enter the amount to move particle B downward: "))
 particle_a, particle_b = move_downward(particle_a, particle_b, k, l)
 elif option == 8:
 save_gif(coordinates)
 break
 else:

```

```
print("Invalid option!")

coordinates.append(particle_a.copy())
coordinates.append(particle_b.copy())

if __name__ == "__main__":
 main()
```

----- Content of SignalProcessing.py-----

```
import numpy as np
from scipy.signal import lfilter
from scipy.fftpack import fft

transitions.txt:
Purpose: Defines the transition rules for the DFSM.
Format: Each line represents a transition rule, formatted as <current state> <input condition> <next state>.
Example:
Start condition1 Middle
Middle condition2 End

filter_coeffs.txt:
Purpose: Defines the coefficients for the digital filter.
Format: Two numbers representing the coefficients b and a for the digital filter, separated by whitespace.
Example:
0.1 0.9

fft_settings.txt:
Purpose: Defines the settings for the Fourier Transform operation.
Format: A single number representing the number of points to use in the Fourier Transform.
Example:
256

accepting_conditions.txt:
Purpose: Defines the conditions under which a state is considered to be an accepting state.
Format: Each line represents an accepting condition for a state, formatted as <state> <condition>.
Example:
End condition3

input.txt:
Purpose: Contains the input signals for the DFSM to process.
Format: Each line represents an input signal, with the signal values separated by whitespace.
Example:
0.1 0.2 0.3 0.4
0.5 0.6 0.7 0.8
```

class DeterministicFSM:

```

def __init__(self):
 self.state = 'Start'
 self.transitions = {}
 self.accepting_conditions = {}

def load_transitions(self, filename):
 with open(filename, 'r') as file:
 for line in file:
 state, input_condition, next_state = line.strip().split()
 self.transitions[(state, input_condition)] = next_state

def load_accepting_conditions(self, filename):
 with open(filename, 'r') as file:
 for line in file:
 state, condition = line.strip().split()
 self.accepting_conditions[state] = condition

def transition(self, input_signal):
 # Apply digital filtering
 with open('filter_coeffs.txt', 'r') as file:
 b, a = map(float, file.read().strip().split())
 filtered_signal = lfilter([b], [a, -1], input_signal)

 # Perform Fourier Transform
 with open('fft_settings.txt', 'r') as file:
 n = int(file.read().strip())
 fft_result = fft(filtered_signal, n)

 # Determine transition based on processing results
 input_condition = 'condition1' # Define conditions based on processing results
 self.state = self.transitions.get((self.state, input_condition), self.state)

def is_in_accepting_state(self):
 condition = self.accepting_conditions.get(self.state)
 # Evaluate condition based on current state and/or processing results
 # ...

Usage
dfsm = DeterministicFSM()
dfsm.load_transitions('transitions.txt')
dfsm.load_accepting_conditions('accepting_conditions.txt')

Assume input.txt contains input signals, one per line
with open('input.txt', 'r') as file:
 input_signals = [list(map(float, line.strip().split())) for line in file]

output_lines = []
for signal in input_signals:
 dfsm.transition(signal)
 output_lines.append(f'Current State: {dfsm.state}')

```

```

Check acceptance
output_lines.append(f'Is in accepting state: {dfsm.is_in_accepting_state()}')

Write output to file
with open('dfsm_o.txt', 'w') as file:
 file.write('\n'.join(output_lines))

#Note... Note yet Tested. In Beta

----- Content of simple_os.py-----
import tkinter as tk
from tkinter import ttk, filedialog
import os
import subprocess

class SimpleOS:
 def __init__(self, root):
 self.root = root
 self.root.title("SimpleOS")
 self.root.geometry("800x600")
 self.current_directory = os.getcwd() # Set initial directory to the working directory
 self.create_menu()

 def create_menu(self):
 menu = tk.Menu(self.root)

 app_menu = tk.Menu(menu, tearoff=0)
 self.update_app_menu(app_menu)

 menu.add_cascade(label="Applications", menu=app_menu)

 settings_menu = tk.Menu(menu, tearoff=0)
 settings_menu.add_command(label="Change Directory", command=self.change_directory)

 menu.add_cascade(label="Settings", menu=settings_menu)

 self.root.config(menu=menu)

 def update_app_menu(self, app_menu):
 app_menu.delete(0, tk.END) # Clear existing menu items
 for filename in os.listdir(self.current_directory):
 if filename.endswith('.exe'):
 app_menu.add_command(label=filename, command=lambda f=filename: self.run_file(f))

 def run_file(self, filename):
 filepath = os.path.join(self.current_directory, filename)
 if filename.endswith('.exe'):
 subprocess.run([filepath])

```

```
def change_directory(self):
 new_directory = filedialog.askdirectory()
 if new_directory:
 self.current_directory = new_directory
 self.create_menu() # Re-create the menu to reflect the new directory

if __name__ == "__main__":
 root = tk.Tk()
 app = SimpleOS(root)
 root.mainloop()
```

## ----- Content of Solution.rb -----

```
ar = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", "", "\n", "\t", "\\", "\\", "\\", "#", "\~"], [{"{}": "\\"}, {"{}": "\-"}, {"{}": "\!"}, {"{}": "\^"}, {"{}": "\&"}, {"{}": "*"}, {"{}": "\("}, {"{}": "\)"}, {"{}": "_"}, {"{}": "\+"}, {"{}": "\="}, {"{}": "\A"}, {"{}": "\B"}, {"{}": "\C"}, {"{}": "\D"}, {"{}": "\E"}, {"{}": "\F"}, {"{}": "\G"}, {"{}": "\H"}, {"{}": "\I"}, {"{}": "\J"}, {"{}": "\K"}, {"{}": "\L"}, {"{}": "\M"}, {"{}": "\N"}, {"{}": "\O"}, {"{}": "\P"}, {"{}": "\Q"}, {"{}": "\R"}, {"{}": "\S"}, {"{}": "\T"}, {"{}": "\U"}, {"{}": "\V"}, {"{}": "\W"}, {"{}": "\X"}, {"{}": "\Y"}, {"{}": "\Z"}, {"{}": "\O"}]
```

print ar

`k = ar.length - 1`

$$p = k + 1$$

puts p;

row = C

**id = 0**

```
print "\n\nN
```

`nocs = gets`

`nbr_comb = (k+1)**noc`

```
f = File.open("Solution_rb")
```

$w < \text{nbr\_com}$

`id = id + 1`

```
f.print "File
```

```
f.print "\n"
```

col = noc-1

$\geq 0$  do

$$rdV = (k+1)^{-r} \det$$

cell = (row/rdiv)

#buffer = a

col = col - 1

1

1

1

`fclose()`

Content of SOLUTION\_MAX.cpp

## ----- Content of sub-



```

scanf("%d", &noc);
printf("\n\nNumber Of FILE Cells = %d", noc);
int n = noc;
int row, col;
int cell;
int rdiv;
unsigned long long id;
id = 0;
int rin;
int nbr_comb = pow(k+1, n);
if(cheque == 1){
 for (row=0; row < nbr_comb; row++){
 id++; fprintf(p,"\\n\\nF%ld\\n\\n", id);
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);
 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 fprintf(p,"%c", a[rin]);
 }
 //printf("\\n");
 }
}
if(cheque == 0){
 for (row=0; row < nbr_comb; row++){
 id++; fprintf(p,"\\n\\nF%ld\\n\\n", id);
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 cell = (row/rdiv) % (k+1);
 fprintf(p,"%c", a[cell]);
 }
 //printf("\\n");
 }
}
fprintf(p,"\\n\\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
fclose(p);
//end of adaptation
return 0;
}

```

----- Content of SOLUTION\_MAX\_Instruction\_Set.cpp -----

//SOLUTION\_MAX.cpp, written by Dominic Alexander Cooper (Database of Files from the given set of elements/ functions, that being a).

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
using namespace std;
//put together by Dominic Alexander Cooper
int main(){

```

```

//Code Adapted by DAC from lyst on https://stackoverflow.com
//k+1 = no. of elements
//n = exponent = number of cells
//the k and n values must perfectly fit the size of the set of elements in question
FILE *p; p = fopen("Instruction_Set_Rename.txt","w");
//char a[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','\n','t','\\','\\','/','<','>','?','!','@','#','~','[','{','}','^','|','!','"','$', '%','^','&','*','(',')','_','+', '='}; A,B,C,D,E,F,G,H,
'!','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9';
char a[] =
{'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','\\','\\','/','<','>','?','!','@','#','~','[','{','}','^','|','!','"','$', '%','^','&','*','(',')','_','+', '='}, A,B,C,D,E,F,G,H,
'!','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9';
int pin;

int pr, pc;
printf("\nLet us begin\n");
pc = 97;
//puts("26 = SPACE");
//puts("27 = NEWLINE");
//puts("28 = TAB");
puts("");
for(pr = 0; pr < pc; pr++){
 printf("%d %c\n", pr, a[pr]);
}

int cheque;
cout << "Enter the size of your array: ";
cin >> pin;
cout << "Do you want to create a custom array? ";
cin >> cheque;
int array[pin], inn, position;
if(cheque == 1){
 cout << "\n\n Enter the " << pin << " elements of your array: \n\n";
 for(inn = 0; inn < pin; inn++){
 cin >> array[inn];
 }
}
//char a[] = {'1','2','3','4','p1','p2','p3','p4','p5','p6'};
//char b[] = {'a','b'};
//char c[] = {'0','1','2','3'};
//int n = 4; //int k = 3; //int n = x;
//int k = 100;

//int k = strlen(a)-1;
int k;
k = pin- 1;
printf("\n\tk = %d", k);
int noc; printf("\n\tn = ");
scanf("%d", &noc);

```

```

printf("\n\tNumber Of FILE Cells = %d", noc);
int n = noc;
int row, col;
int cell;
int rdiv;
unsigned long long id;
id = 0;
int rin;
int nbr_comb = pow(k+1, n);
if(cheque == 1){
 for (row=0; row < nbr_comb; row++){
 id++;
 //fprintf(p,"\\n\\nF%d\\n\\n", id);
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 //cell = (row/rdiv) % (k+1); fprintf(p,"%c", a[cell]);
 cell = (row/rdiv) % (k+1);
 rin = array[cell];
 fprintf(p,"%c", a[inr]);
 }
 fprintf(p,"\\n");
 }
}
if(cheque == 0){
 for (row=0; row < nbr_comb; row++){
 id++;
 //fprintf(p,"\\n\\nF%d\\n\\n", id);
 for (col=n-1; col>=0; col--){ rdiv = pow(k+1, col);
 cell = (row/rdiv) % (k+1);
 fprintf(p,"%c", a[cell]);
 }
 fprintf(p,"\\n");
 }
}
//fprintf(p,"\\n\\nend.(k+1)^n = (%d + 1)^%d = %d", k, n, id);
fclose(p);
//end of adaptation
return 0;
}

```

----- Content of Statement-Informatics.py-----

```

from tkinter import *
import tkinter as tk
#from PIL.Image import core as _imaging
import random
import os
import math
import time
import sys

```



```
st_f = open(gi1, "r")
st_buff = ['0']
got = st_f.read()
while(got != ""):
 print(got)
 st_buff.extend(got)
 got = st_f.read()
del st_buff[0]
print(st_buff)

#...

conda = len(st_buff)
yields = 0
yeldo = input("Number of Characters in your Language:")
yeldo_ = int(yeldo)
iter_99 = 0
iter_98 = -1
benine = ['0']
pnaf = len(a)
#s13 = 1
while(iter_99 < conda):

 if(iter_98 < yeldo_):
 iter_98 = iter_98 + 1
 if(iter_98 == yeldo_):
 iter_98 = 0

 if(st_buff[iter_99] == a[iter_98]):
 column_ = str(iter_98 + 1)
 s_f.write(column_)
 s_f.write(' ')
 benine.append(iter_98 + 1)
 iter_99 = iter_99 + 1
 yields = yields + 1

del benine[0]

print(benine)
print("Length of array is ...")
print(conda)
```

```

print("random() : ", random.random())
master = Tk()
master.attributes('-fullscreen', True)
#a = 250
#b = 200
print("Welcome\n")
print("Tip: side length should be a factor of the image width and of the image height")
change1 = input("Enter side length of image block: ")
change = int(change1)
#a = 1880
#b = 1050
a1 = input("Enter width of image: ")
a = int(a1)
b1 = input ("Enter height of image: ")
b = int(b1)
pin_p = a/change
w = Canvas(master, width= a, height= b)
#psv = getscreen().getcanvas()

#files = 3
#files_buffer = input("Enter number of files to be generated (8^((width/side_length)*(height/side_length)):")

#files = int(files_buffer)

#c = ["red","blue","yellow","brown","purple","pink","green","orange"]
c = ["gold"]
xc = 0
zerox = 0
zeroy = 0
p = 1
range_for = int((a/change)*(b/change))
name = 1
cells = ((a//change)*(b//change))
upper = cells- 1
nbr_comb = math.pow(len(c),cells)
files = int(nbr_comb)
#print(len(c))
rown = 0
#rdiv = math.pow(len(c),cells- 1)
#print(rdiv)
#cell = (row/rdiv) % (len(c))
#print(cell)

#rdiv = math.pow(len(c),cells- 2)
#print(rdiv)
#cell = (row/rdiv) % (len(c))
#print(cell)
#def rone(h,j,l):
cell = (h/j) % l

```

```
celled = int(cell
w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = c[celled])
file_count = 0
st_f.close()
s_f.close()
#giilp = 1
aiilp = 0
#for t in range(0,files):
x_coun = 0
create_a = ['0']
z_coun = 0
dcoun = 0
lk = 0

if(conda > 100):
 print("Max Characters allowed: 100")
 time.sleep(5)
 sys.exit(1)

while(z_coun < cells):
 create_a.append(lk)
 z_coun = z_coun + 1
zcoun = 0

while(zcoun < condn):
 create_a[benine[zcoun]] = benine[dcoun]
 zcoun = zcoun + 1
 dcoun = dcoun + 1

del create_a[0]

print(create_a)
zcoun = 0
tr_ = len(create_a)
print(tr_)
for t in range(0,files):
 file_count = file_count + 1
 #if(file_count == 121):
 # break
 switch = 1
 sw = 0
 #for x in range(range_for):
 #for x in range (0,cells):
 x = 0
 col = cells- 1
 #print(rown)
 pixlp = 0
 while(x < cells):
```

```

#f = open('%s.ps' % name, 'wb')
#f.close
if(switch == 1):
 row = 1
 nxleft = 0
 nxright = change
 nyleft = 0
 nyright = change
 zerox = 0
 zeroy = 0
c_length = len(c)
switch = 0
#ran = random.randint(0,c_length- 1)

rdiv = math.pow(len(c),col)
cell = (rown/rdiv) % (len(c))
#rone(row,rdiv,len(c))
#print(cell)
#print(rdiv)
celled = int(cell)
#print(celled)
#print(cell)
#w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[ran], outline = c[ran])

kij = st_buff[aiilp]

#for inh in range(0,100*conda):

if(create_a[zcoun] > 0):
 w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[celled], outline = "green", width = 0)
 #giilp = 0
 #zcoun += 1

if(create_a[zcoun] == 0):
 w.create_rectangle(zerox, nyleft, nxright,nyright, fill = "purple", outline = "black", width = 0)

zcoun = zcoun + 1

#z_coun = z_coun + 1
#pixlp = 0
#if(giilp > 100):
giilp = 0

if(x <= cells):
 col = col- 1
#w.create_rectangle(zerox, nyleft, nxright,nyright, fill = c[ran])
#zxbuffer = zerox

```

```
#w.place(x = zerox, y = zeroy)
#w.place(bordemode = OUTSIDE, x = zerox + change, y = zeroy)
w.grid(row = zeroy, column = zerox + change)
if(p >= pin_p and p%pin_p == 0):
 zeroy = zeroy + change
 zerox -=change
 #zerox = 0
 nxleft = change
 nxright = 0
 nyleft = nyleft + change
 nyright = nyright + change
 zerox = zerox + change
 p = p + 1
 #xc + 1
 nxright = nxright + change
 if(xc == 3):
 xc = 0
 x = x + 1
 rown = rown + 1
 ce = str(name)
 w.update()
```

```
def savefirst():
 w.postscript(file = 'M' + gi3 + '.ps', colormode='color')

savefirst()

name = name + 1
#f_p = 'D:\\Kaliber\\Portfolio\\Content\\My PhotoBook\\Content\\'
#os.rename(f_p + '1.ps', f_p + '2.ps')

#process = subprocess.Popen(["ps2pdf", ce + ".ps", ce + ".pdf"], shell=True)
```

```
#Contender 1 For File Saving
"""

def savefirst():
 cnv = getscreen().getcanvas()
 global hen
 ps = cnv.postscript(colormode = 'color')
 hen = filedialog.asksaveasfilename(defaultextension = '.jpg')
 im = Image.open(io.BytesIO(ps.encode('utf-8')))
 im.save(hen + '.jpg')
#savefirst()
```

```

#####
#####
Second Contender For File Saving
def save(w):
 ps = w.canvas.postscript(colormode='color')
 img = Image.open(io.BytesIO(ps.encode('utf-8')))
 img.save('testing.jpg')
#####
#w.update()
#script = ce + ".ps"
#w.save(script)
#os.startfile(script)
#print(script)
print("Done")

#master.mainloop()

```

```

----- Content of StudyPlanner.py-----
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import random
import math
import json # For logging

Initialize cube states
cube_states = {}
for x in range(11):
 for y in range(11):
 for z in range(11):
 cube_states[(x, y, z)] = 'RED'

Colors for different states
colors = [
 [1, 0, 0], # Red for unmastered topics
 [0, 1, 0], # Green for familiar topics
 [0, 0, 1] # Blue for mastered topics
]

Function to read cube states from a text file
def read_states_from_file(filename):
 color_map = {'Red': 0, 'Green': 1, 'Blue': 2}
 with open(filename, 'r') as f:
 lines = f.readlines()
 states = {}
 for index, line in enumerate(lines):

```

```

color = line.strip()
x, y, z = get_cube_coordinates(index, 11) # Assuming side_length is 11
states[(x, y, z)] = color_map.get(color, 0) # Default to 0 (Red) if color is not recognized
return states

Function to get cube coordinates based on its index
def get_cube_coordinates(index, side_length):
 x = index // (side_length * side_length)
 y = (index % (side_length * side_length)) // side_length
 z = index % side_length
 return x, y, z

Read states from a text file
cube_states = read_states_from_file('cube_states.txt')

Function to draw a cube at (x, y, z)
def draw_cube(x, y, z, color):
 vertices = [
 [x, y, z],
 [x + 1, y, z],
 [x + 1, y + 1, z],
 [x, y + 1, z],
 [x, y, z - 1],
 [x + 1, y, z - 1],
 [x + 1, y + 1, z - 1],
 [x, y + 1, z - 1],
]
 faces = [
 (0, 1, 2, 3),
 (3, 2, 7, 6),
 (6, 7, 5, 4),
 (4, 5, 1, 0),
 (1, 5, 6, 2),
 (4, 0, 3, 7),
]
 glBegin(GL_QUADS)
 glColor3fv(color)
 for face in faces:
 for vertex in face:
 glVertex3fv(vertices[vertex])
 glEnd()

Function to change the state of a cube
def change_state(x, y, z):
 current_state = cube_states[(x, y, z)]
 if current_state == 'RED':
 cube_states[(x, y, z)] = 'GREEN'
 elif current_state == 'GREEN':
 cube_states[(x, y, z)] = 'BLUE'
 elif current_state == 'BLUE':

```

```
cube_states[(x, y, z)] = 'RED'

print("")

Initialize Pygame
pygame.init()
display = (900, 700)
pygame.display.set_mode(display, DOUBLEBUF | OPENGL)
gluPerspective(45, (display[0] / display[1]), 0.1, 50.0)
gTranslatef(0.0, 0.0, -40)

Initialize selected cube coordinates and previous selected cube
selected_x, selected_y, selected_z = 0, 0, 0
prev_selected_x, prev_selected_y, prev_selected_z = None, None, None

Function to get the index of a cube based on its coordinates
def get_cube_index(x, y, z, side_length):
 return x * side_length * side_length + y * side_length + z

Initialize rotation angles
x_angle = 0
y_angle = 0

Initialize log
state_log = []

Main Loop
while True:
 for event in pygame.event.get():
 if event.type == pygame.QUIT:
 # Save log to a file
 with open('state_log.json', 'w') as f:
 json.dump(state_log, f)
 pygame.quit()
 quit()
 elif event.type == pygame.KEYDOWN:
 if event.key == pygame.K_LEFT:
 y_angle -= 5
 elif event.key == pygame.K_RIGHT:
 y_angle += 5
 elif event.key == pygame.K_UP:
 x_angle -= 5
 elif event.key == pygame.K_DOWN:
 x_angle += 5
 elif event.key == pygame.K_w: # Move selection up
 selected_y += 1
 elif event.key == pygame.K_s: # Move selection down
 selected_y -= 1
 elif event.key == pygame.K_a: # Move selection left
```

```

 selected_x-= 1
 elif event.key == pygame.K_d: # Move selection right
 selected_x += 1
 elif event.key == pygame.K_q: # Move selection backward
 selected_z-= 1
 elif event.key == pygame.K_e: # Move selection forward
 selected_z += 1
 elif event.key == pygame.K_SPACE: # Change state of selected cube
 change_state(selected_x, selected_y, selected_z)
 state_log.append({'coords': (selected_x, selected_y, selected_z), 'new_state':
cube_states[(selected_x, selected_y, selected_z)]})

Print the index and coordinates only if the selected cube has changed
if (selected_x, selected_y, selected_z) != (prev_selected_x, prev_selected_y, prev_selected_z):
 index = get_cube_index(selected_x, selected_y, selected_z, 11) # Assuming side_length is 11
 print(f"Selected Cube Index: {index}, Coordinates: ({selected_x}, {selected_y}, {selected_z})")
 prev_selected_x, prev_selected_y, prev_selected_z = selected_x, selected_y, selected_z

glPushMatrix()
glRotatef(x_angle, 1, 0, 0)
glRotatef(y_angle, 0, 1, 0)
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

Draw smaller cubes to form a larger cube
for x in range(11):
 for y in range(11):
 for z in range(11):
 color_index = cube_states.get((x, y, z), 0) # Default to 0 (Red) if key is not found
 color = colors[color_index] # Map the state to its corresponding color
 draw_cube(x * 2 - 10, y * 2 - 10, z * 2 - 10, color)

glPopMatrix()

pygame.display.flip()
pygame.time.wait(50)

```

----- Content of Tally.py-----

```

import tkinter as tk
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pickle

```

```

class TallyCounterApp:
 def __init__(self, master):
 self.master = master
 master.title("Tally Counter")

```

# Variables

```
self.tally = 0
self.tally_history = [0]
self.time_step = 0

User inputs for increment and decrement
self.increment_value = tk.IntVar(value=1)
self.decrement_value = tk.IntVar(value=1)

Layout
self.label = tk.Label(master, text="Tally: 0")
self.label.pack()

self.increment_entry = tk.Entry(master, textvariable=self.increment_value)
self.increment_entry.pack()

self.decrement_entry = tk.Entry(master, textvariable=self.decrement_value)
self.decrement_entry.pack()

self.increase_button = tk.Button(master, text="Increase", command=self.increase_tally)
self.increase_button.pack()

self.decrease_button = tk.Button(master, text="Decrease", command=self.decrease_tally)
self.decrease_button.pack()

Save and Load buttons
self.save_button = tk.Button(master, text="Save Session", command=self.save_session)
self.save_button.pack()

self.load_button = tk.Button(master, text="Load Session", command=self.load_session)
self.load_button.pack()

Chart
self.fig = Figure(figsize=(5, 4), dpi=100)
self.plot = self.fig.add_subplot(111)
self.canvas = FigureCanvasTkAgg(self.fig, master=master)
self.canvas_widget = self.canvas.get_tk_widget()
self.canvas_widget.pack()

self.update_plot()

def increase_tally(self):
 self.tally += self.increment_value.get()
 self.update_tally()

def decrease_tally(self):
 self.tally -= self.decrement_value.get()
 self.update_tally()

def update_tally(self):
```

```

self.label.config(text=f"Tally: {self.tally}")
self.tally_history.append(self.tally)
self.time_step += 1
self.update_plot()

def update_plot(self):
 self.plot.clear()
 self.plot.plot(range(self.time_step + 1), self.tally_history, marker='o')
 self.plot.set_title("Tally Over Time")
 self.canvas.draw()

def save_session(self):
 with open('tally_session.pkl', 'wb') as f:
 pickle.dump({'tally': self.tally, 'tally_history': self.tally_history, 'time_step': self.time_step}, f)
 print("Session saved!")

def load_session(self):
 try:
 with open('tally_session.pkl', 'rb') as f:
 session_data = pickle.load(f)
 self.tally = session_data['tally']
 self.tally_history = session_data['tally_history']
 self.time_step = session_data['time_step']
 self.update_tally()
 except FileNotFoundError:
 print("No saved session found.")

root = tk.Tk()
app = TallyCounterApp(root)
root.mainloop()

```

----- Content of Terminal.py -----

```

import tkinter as tk
from tkinter import scrolledtext, ttk
import subprocess
import os
import platform

class CustomTerminal(tk.Tk):
 def __init__(self):
 super().__init__()

 self.title("Python Terminal")
 self.geometry("1000x700")

 # Command history list and current directory
 self.command_history = []
 self.history_pointer = 0
 self.current_dir = os.getcwd()

```

```
Identify the OS type
self.os_type = platform.system()

Menu bar
self.menu_bar = tk.Menu(self)
self.file_menu = tk.Menu(self.menu_bar, tearoff=0)
self.file_menu.add_command(label="New Tab", command=self.add_tab, accelerator="Ctrl+T")
self.file_menu.add_command(label="Exit", command=self.on_closing)
self.menu_bar.add_cascade(label="File", menu=self.file_menu)
self.config(menu=self.menu_bar)

self.notebook = ttk.Notebook(self)
self.notebook.pack(expand=True, fill='both')

self.add_tab()
self.bind("<Control-t>", lambda event: self.add_tab())

self.protocol("WM_DELETE_WINDOW", self.on_closing)

def add_tab(self):
 tab = ttk.Frame(self.notebook)
 text_widget = scrolledtext.ScrolledText(tab, wrap='word')
 text_widget.pack(expand=True, fill='both')
 command_entry = tk.Entry(tab)
 command_entry.pack(fill='x')
 command_entry.bind('<Return>', self.execute_command)
 command_entry.bind('<Up>', self.prev_command)
 command_entry.bind('<Down>', self.next_command)

 tab.text_widget = text_widget

 self.notebook.add(tab, text="Terminal")
 self.notebook.select(tab)
 command_entry.focus()

def execute_command(self, event):
 command_entry = event.widget
 command = command_entry.get()
 if not command:
 return

 # Choose the shell based on OS type
 if self.os_type == "Windows":
 cmd = ['cmd.exe', '/c', command]
 else:
 cmd = ['/bin/bash', '-c', command]

 try:
 process = subprocess.Popen(cmd, cwd=self.current_dir, stdout=subprocess.PIPE,
```

```

stderr=subprocess.PIPE, text=True)
stdout, stderr = process.communicate()
output = stdout + stderr

Update the current directory if 'cd' command is used
if command.startswith('cd '):
 path = command.split(' ', 1)[1]
 self.current_dir = os.path.join(self.current_dir, path)

except Exception as e:
 output = f'Error: {e}\n'

self.command_history.append(command)
self.history_pointer = len(self.command_history)
command_entry.delete(0, tk.END)
tab = self.notebook.nametowidget(self.notebook.select())
text_widget = tab.text_widget
text_widget.insert(tk.END, f'{command}\n{output}')
text_widget.see(tk.END)

def prev_command(self, event):
 command_entry = event.widget
 if self.command_history and self.history_pointer > 0:
 self.history_pointer -= 1
 command_entry.delete(0, tk.END)
 command_entry.insert(tk.END, self.command_history[self.history_pointer])

def next_command(self, event):
 command_entry = event.widget
 if self.command_history and self.history_pointer < len(self.command_history) - 1:
 self.history_pointer += 1
 command_entry.delete(0, tk.END)
 command_entry.insert(tk.END, self.command_history[self.history_pointer])
 else:
 command_entry.delete(0, tk.END)

def on_closing(self):
 self.destroy()

if __name__ == "__main__":
 terminal = CustomTerminal()
 terminal.mainloop()

----- Content of textFileEdit2.cpp-----
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
```

```
int main() {
 std::string filename;
 std::cout << "Enter the text file name: ";
 std::cin >> filename;

 std::ifstream inputFile(filename);
 if (!inputFile.is_open()) {
 std::cerr << "Error opening file: " << filename << std::endl;
 return 1;
 }

 std::vector<std::string> lines;
 std::string line;
 while (std::getline(inputFile, line)) {
 lines.push_back(line);
 }
 inputFile.close();

 int option;
 while (true) {
 std::cout << "\nChoose an option:\n"
 << "1. Edit a line\n"
 << "2. Delete a line\n"
 << "3. Add a line\n"
 << "4. View a specific line\n"
 << "5. Save and exit\n"
 << "Option: ";
 std::cin >> option;

 if (option == 1) {
 size_t lineNo;
 std::cout << "Enter the line number to edit: ";
 std::cin >> lineNo;
 std::cin.ignore();

 if (lineNo > 0 && lineNo <= lines.size()) {
 std::cout << "Enter the new text for the line: ";
 std::getline(std::cin, line);
 lines[lineNo - 1] = line;
 } else {
 std::cout << "Invalid line number!\n";
 }
 } else if (option == 2) {
 size_t lineNo;
 std::cout << "Enter the line number to delete: ";
 std::cin >> lineNo;

 if (lineNo > 0 && lineNo <= lines.size()) {
 lines.erase(lines.begin() + lineNo - 1);
 }
 }
 }
}
```

```

 } else {
 std::cout << "Invalid line number!\n";
 }
} else if (option == 3) {
 std::cout << "Enter the text for the new line: ";
 std::cin.ignore();
 std::getline(std::cin, line);
 lines.push_back(line);
} else if (option == 4) {
 size_t lineNo;
 std::cout << "Enter the line number to view: ";
 std::cin >> lineNo;

 if (lineNo > 0 && lineNo <= lines.size()) {
 std::cout << "Line " << lineNo << ":" << lines[lineNo - 1] << std::endl;
 } else {
 std::cout << "Invalid line number!\n";
 }
} else if (option == 5) {
 break;
} else {
 std::cout << "Invalid option!\n";
}
}

std::ofstream outputFile(filename, std::ios::trunc);
if (!outputFile.is_open()) {
 std::cerr << "Error opening file: " << filename << std::endl;
 return 1;
}

for (const auto& l : lines) {
 outputFile << l << std::endl;
}
outputFile.close();

std::cout << "Changes saved successfully.\n";

return 0;
}

```

----- Content of ULang-integers.py-----

```
Unicode escape sequence
```

```
unicode_escape = input("Unicode for equivalent integer code: e.g. '\\uE031': ")
```

```
Remove the '\\u' prefix and convert the remaining hexadecimal number to an integer
integer_number = int(unicode_escape[2:], 16)
```

```
print(f"The integer representation of '{unicode_escape}' is {integer_number}.")
```

