Project Progress 1

โมกข์ มาอาจ 65-040626-2015-9
อธิชา เล็กสรรเสริญ 65-040626-2017-5

Containers

history.csv: เก็บข้อมูลประวัติการเล่น

- date: เวลาที่เกมนั่นจบลง (Epoch time)
- target_word: คำตอบของเกมนั่น
- guess_count: จำนวนที่ผู้เล่นตอบ (-1 ถ้าแพ้)
- guess_word1 - guess_word6: คำที่ผู้เล่นเดา

ข้อมูลตัวอย่าง:

```
1665291629.7163048,fells,6,crane,epoxy,bezel,devil,jewel,fells
1665291684.4712517,jacks,3,crane,wacky,jacks,,,
1665291827.2184944,twats,-1,crane,plaza,quash,staff,toast,staid
```

- **entryList**: ไว้เก็บ Entry ที่เป็นตารางการเล่น (5 x 6) สำหรับแก้ไขสี

ข้อมูลตัวอย่าง: entryList = [
[<tkinter.Entry object .!frame29.!entry>, ...], # ขนาด 5
[<tkinter.Entry object .!frame29.!entry>, ...], # ขนาด 5
...
] # ขนาด 6

- **buttonList**: ไว้เก็บ Button ที่เป็นคีย์บอร์ดสำหรับแก้ไขสี (เก็บตามตัวอักษร)

ข้อมูลตัวอย่าง: buttonList = {
'q': <tkinter.Button object .!frame.!button2>,
'w': <tkinter.Button object .!frame2.!button2>,
...
}

- **textVariableList**: เหมือนกับ entryList แต่เก็บ textVariable แทนไว้ใช้เปลี่ยนค่าที่แสดงอยู่
- **wordsList**: ไว้เก็บคำศัพท์ของ Wordle ทั้งหมด (เปิดมาจากไฟล์)
- **guessList**: ไว้เก็บคำที่ผู้เล่นเดา ในเกมนั่นๆ (รีเซ็ตทุกๆ เกม)

ข้อมูลตัวอย่าง: guessList = [‘crane’, ‘plaza’, ‘quash’, ‘staff’, ‘toast’, ‘staid’]

- **targetWordCount**: จำนวนของแต่ละตัวอักษรของคำตอบ (รีเซ็ตทุกๆ เกม)

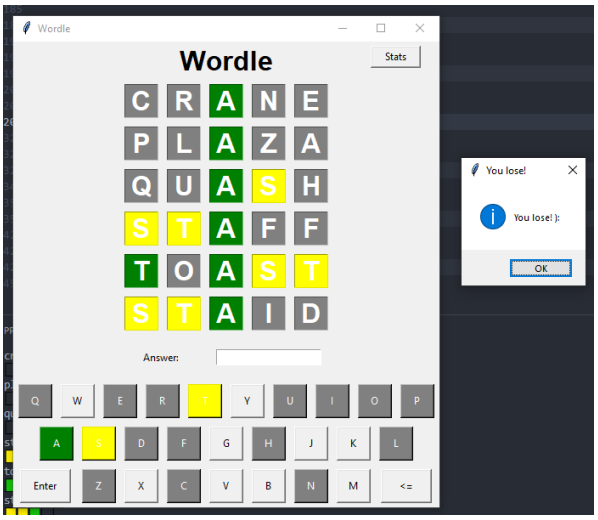
ข้อมูลตัวอย่าง: ถ้าคำตอบ = ‘fells’, targetWordCount = {
‘f’: 1, ‘e’: 1, ‘l’: 2, ‘s’: 1
}

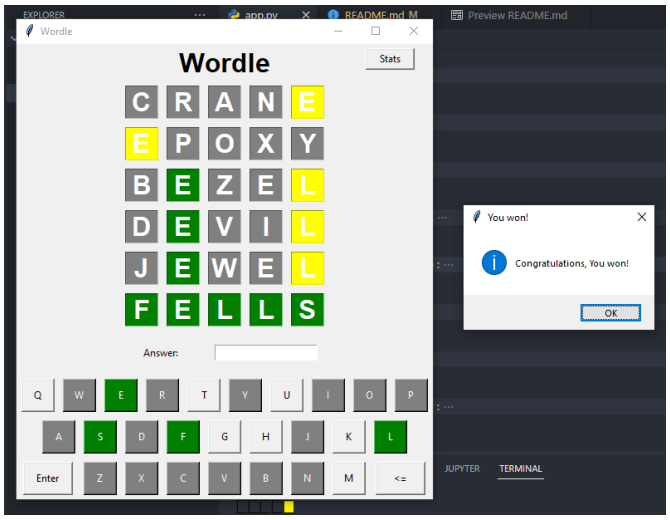
- **currWordState**: ไว้เก็บข้อมูล (ว่าเป็นสีเหลือง หรือเขียว) ของคำที่ผู้เล่นเดา (รีเซ็ตทุกๆ ครั้งที่ผู้เล่นพิมพ์)

ข้อมูลตัวอย่าง: ถ้าคำตอบ = ‘wonts’, ผู้เล่นเดาว่า ‘pains’, จะได้ currWordState = {
0: {'char': 'p', 'color': 'gray'},
1: {'char': 'a', 'color': 'gray'},
2: {'char': 'i', 'color': 'gray'},
3: {'char': 'n', 'color': 'yellow'},
4: {'char': 's', 'color': 'green'}
}

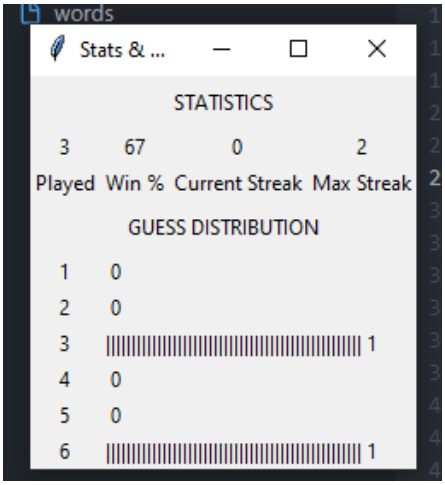
GUI (Graphics & Coding)

- หน้า GUI ระหว่างเล่น (ชนะ / แพ้)





- หน้า GUI ของ stats & History



Program (Python Coding)

```
from math import floor
from tkinter import *
from tkinter import messagebox

import random
import time
import csv

# Global Game Variable
entryList = []          # Store all Entry
buttonList = {}         # Store all Keyboard's Button
textVariableList = []   # Store all Entry's TextVariable
wordsList = []          # Wordle's words list
answerEntry = None      # Answer Entry box
answerVariable = None   # Answer Entry box's TextVariable
guessList = []          # Current Game's guess words
targetWord = ''         # Current Game's target word
currRow = 0             # Current Game's playing row


# Constants
HISTORY_HEADER = [
    'date',
    'target_word',
    'guess_count',
    'guess_word1',
    'guess_word2',
    'guess_word3',
    'guess_word4',
    'guess_word5',
    'guess_word6'
]


def getHistory():
    try:
        f = open('history.csv', 'r+', newline='')
    except:
        try:
            # File does not existed
            f = open('history.csv', 'w+', newline='')
        except:
            print('Error can\'t create history.csv File, getHistory()')
```

```
            return []

    csvReader = csv.reader(f)
    historyData = [row for row in csvReader]

    # File existed, but it's empty
    if len(historyData) == 0:
        csvWriter = csv.writer(f)
        csvWriter.writerow(HISTORY_HEADER)
    else:
        # Removed first row (header row)
        historyData = historyData[1:]

    f.close()
    return historyData


def updateHistory(data):
    if(len(data) != len(HISTORY_HEADER)):
        print('Invalid History data!, (updateHistory(data))')
        return False

    # Call for header checking ...
    getHistory()

    try:
        f = open('history.csv', 'a', newline='')
    except:
        print('Error can\'t open history.csv File')
        return False

    csvWriter = csv.writer(f)
    csvWriter.writerow(data)
    f.close()

    return True


def drawButton(text='', row=0, rowspan=1, column=0, columnspan=1, width=100,
height=100, command=None, keyboard=False):
    frame = Frame(root, width=width, height=height)
    button = Button(frame, text=text, command=command)
    if(keyboard):
        button = Button(frame, text=text,
                        command=lambda: onKeyboardClick(text))
```

```
    frame.grid_propagate(False)         # Disables resizing of frame
    frame.columnconfigure(0, weight=1)  # Enables button to fill frame
    frame.rowconfigure(0, weight=1)

    frame.grid(row=row, column=column, rowspan=rowspan,
               columnspan=columnspan, padx=5, pady=5)
    button.grid(sticky='wens')
    return button


def drawSquareEntry(textvariable, row=0, rowspan=1, column=0, columnspan=1,
width=100, height=100):
    frame = Frame(root, width=width, height=height)
    entry = Entry(frame, textvariable=textvariable, justify='center',
                  foreground='white', font='Helvetica 24 bold')

    frame.grid_propagate(False)         # Disables resizing of frame
    frame.columnconfigure(0, weight=1)  # Enables button to fill frame
    frame.rowconfigure(0, weight=1)

    frame.grid(row=row, column=column, rowspan=rowspan,
               columnspan=columnspan, padx=5, pady=5)
    entry.grid(sticky='wens')
    return entry


def initKeyboardGUI():
    keyboardLayout = [
        'QWERTYUIOP',
        'ASDFGHJKL',
        'ZXCVBNM'
    ]

    offset = [0, 1, 3]
    startRow, startColumn = 19, 1

    # Draw Keyboard Key
    for inxRow, row in enumerate(keyboardLayout):
        placeRow = startRow + (2 * inxRow)
        for inxCol, text in enumerate(list(row)):
            placeColumn = startColumn + (2 * inxCol) + offset[inxRow]

            btn = drawButton(text, width=40, height=40,
                             row=placeRow, rowspan=2,
```

```
                                column=placeColumn, columnspan=2, keyboard=True
                                )
            buttonList[text.lower()] = btn

    # Enter Button
    drawButton('Enter', row=23, rowspan=2, column=1,
               columnspan=3, width=40 / 2 * 3, height=40, command=checkWord)

    # Return Button
    drawButton('<=', row=23, rowspan=2, column=18,
               columnspan=3, width=40 / 2 * 3, height=40, command=onReturn)


def initDisplay():
    startRow, startColumn = 4, 6

    # Display 6 x 5
    for inxRow in range(6):
        placeRow = startRow + (2 * inxRow)
        textVariableRow = []
        entryRow = []
        for inxCol in range(5):
            placeColumn = startColumn + (2 * inxCol)

            str = StringVar()
            textVariableRow.append(str)

            entry = drawSquareEntry(str,  width=40, height=40,
                                    row=placeRow, rowspan=2,
                                    column=placeColumn, columnspan=2
                                    )
            entry['state'] = DISABLED
            entry['disabledbackground'] = 'white'
            entry['disabledforeground'] = 'white'
            entryRow.append(entry)
        textVariableList.append(textVariableRow)
        entryList.append(entryRow)

    # Answer Box
    Label(root, text='Answer: ').grid(
        row=17, column=6, columnspan=4, pady=15)

    global answerVariable
    answerVariable = StringVar()
```

```
    entryAnswer = Entry(root, textvariable=answerVariable)
    entryAnswer.grid(row=17, column=10, columnspan=6, pady=15)
    entryAnswer.bind('<Return>', checkWord)
    entryAnswer.focus()

    global answerEntry
    answerEntry = entryAnswer


def onReturn():
    currWord = answerVariable.get()
    currWord = currWord[:-1]  # Remove last element

    answerVariable.set(currWord)
    answerEntry.icursor(len(currWord))


def onKeyboardClick(key):
    currWord = answerVariable.get()
    currWord += key.lower()

    answerVariable.set(currWord)
    answerEntry.icursor(len(currWord))


def checkWord(event=None):
    currWord = answerVariable.get().strip().lower()

    # Is word empty
    if(len(currWord) == 0):
        messagebox.showinfo('Please enter again', 'Word can\'t be emptied!')
        return

    # Is word wrong size
    if(len(currWord) != 5):
        messagebox.showinfo('Please enter again', 'Word size must be 5!')
        return

    # Is word a word
    if(currWord not in wordsList):
        messagebox.showinfo('Please enter again', 'Not in word list!')
        return

    print(currWord, targetWord)
    guessList.append(currWord)
```

```
# Create dict of each letter count of Target Word
targetWordCount = {}
for c in targetWord:
    if(c in targetWordCount):
        targetWordCount[c] += 1
    else:
        targetWordCount[c] = 1

currWordState = {}
# Check for exact match
for idx, char in enumerate(currWord):
    if(char == targetWord[idx]):
        # Remove Exact Match from Target Word's letter count
        targetWordCount[char] -= 1

        # Exact Match, green color
        currWordState[idx] = {
            'char': char,
            'color': 'green'
        }
    else:
        # Not Exact Match, can be yellow, or gray
        currWordState[idx] = {
            'char': char,
            'color': 'gray'
        }

for idx, char in enumerate(currWord):
    # Is there is any char in Target Word
    if(char in targetWord):
        if(targetWordCount[char] != 0):
            # If not Exact Match but exist in word, yellow color
            if(currWordState[idx]['color'] != 'green'):
                currWordState[idx]['color'] = 'yellow'

                targetWordCount[char] -= 1
            # No more words left, gray color
            elif(targetWordCount[char] < 1):
                currWordState[idx]['color'] = 'gray'

global currRow
# Set Color, and Char
for idx in currWordState:
    if(currWordState[idx]['color'] == 'green'):
```

```
            print('🟩', end='')
        elif(currWordState[idx]['color'] == 'yellow'):
            print('🟨', end='')
        elif(currWordState[idx]['color'] == 'gray'):
            print('⬛', end='')

        textVariableList[currRow][idx].set(currWordState[idx]['char'].upper())
        # entryList[currRow][idx]['background'] = currWordState[idx]['color']
        entryList[currRow][idx]['disabledbackground'] =
currWordState[idx]['color']
        entryList[currRow][idx]['state'] = DISABLED

        buttonList[currWordState[idx]['char']
                   ]['background'] = currWordState[idx]['color']
        buttonList[currWordState[idx]['char']]['foreground'] = 'white'
    print('')

    answerVariable.set('')
    currRow += 1

    # Answer is correct
    if(currWord == targetWord):
        messagebox.showinfo('You won!', 'Congratulations, You won!')
        history = [
            time.time(),  # date
            targetWord,   # target_word
            currRow,      # guess_count
        ]

        # guess_word1 - guess_word6
        for i in range(6):
            if(i >= len(guessList)):
                history.append('')
            else:
                history.append(guessList[i])
        print(history)

        updateHistory(history)
        gameCycle()
        return

    # You lose ):
    if(currRow == 6):
        messagebox.showinfo('You lose!', 'You lose! ):')
        history = [
```

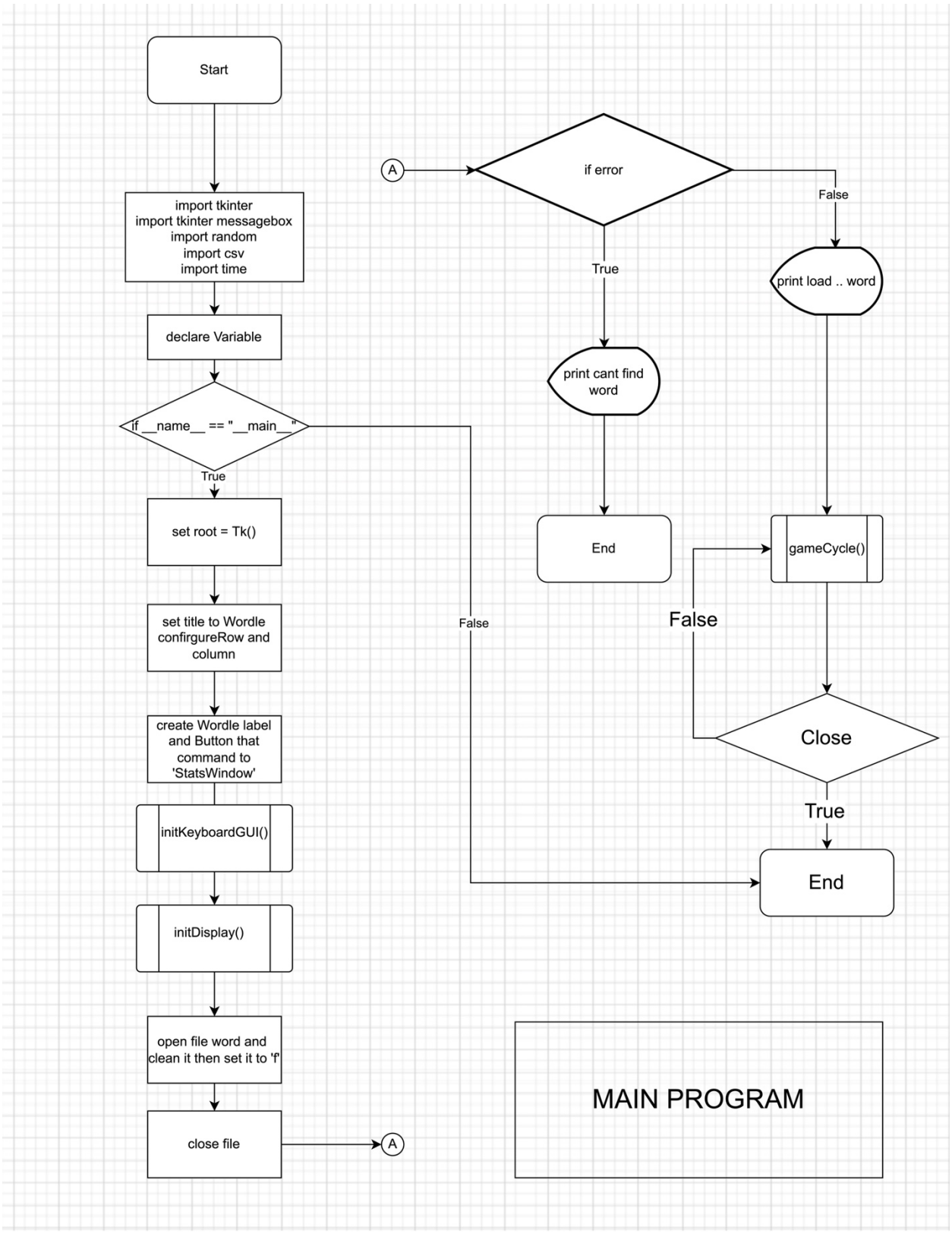
```
            time.time(),   # date
            targetWord,    # target_word
            -1,            # guess_count,
            guessList[0],  # guess_word1
            guessList[1],  # guess_word2
            guessList[2],  # guess_word3
            guessList[3],  # guess_word4
            guessList[4],  # guess_word5
            guessList[5],  # guess_word6
        ]

        updateHistory(history)
        gameCycle()
        return


def gameCycle():
    # Pick random words
    # random.seed('Can I get A dai mai, Ajarn')  # Just for testing
    global targetWord
    targetWord = random.choice(wordsList)

    global guessList
    guessList = []

    # Reset Counter
    global currRow
    currRow = 0
    for idxRow, row in enumerate(textVariableList):
        for idxCol, textVar in enumerate(row):
            textVar.set('')

            entryList[idxRow][idxCol]['disabledbackground'] = 'white'

    for btn in buttonList:
        # Default Button Color
        buttonList[btn]['background'] = 'SystemButtonFace'
        buttonList[btn]['foreground'] = 'black'


def StatsWindow():
    root2 = Tk()
    root2.title('Stats & History | Wordle')

    historyData = getHistory()
```

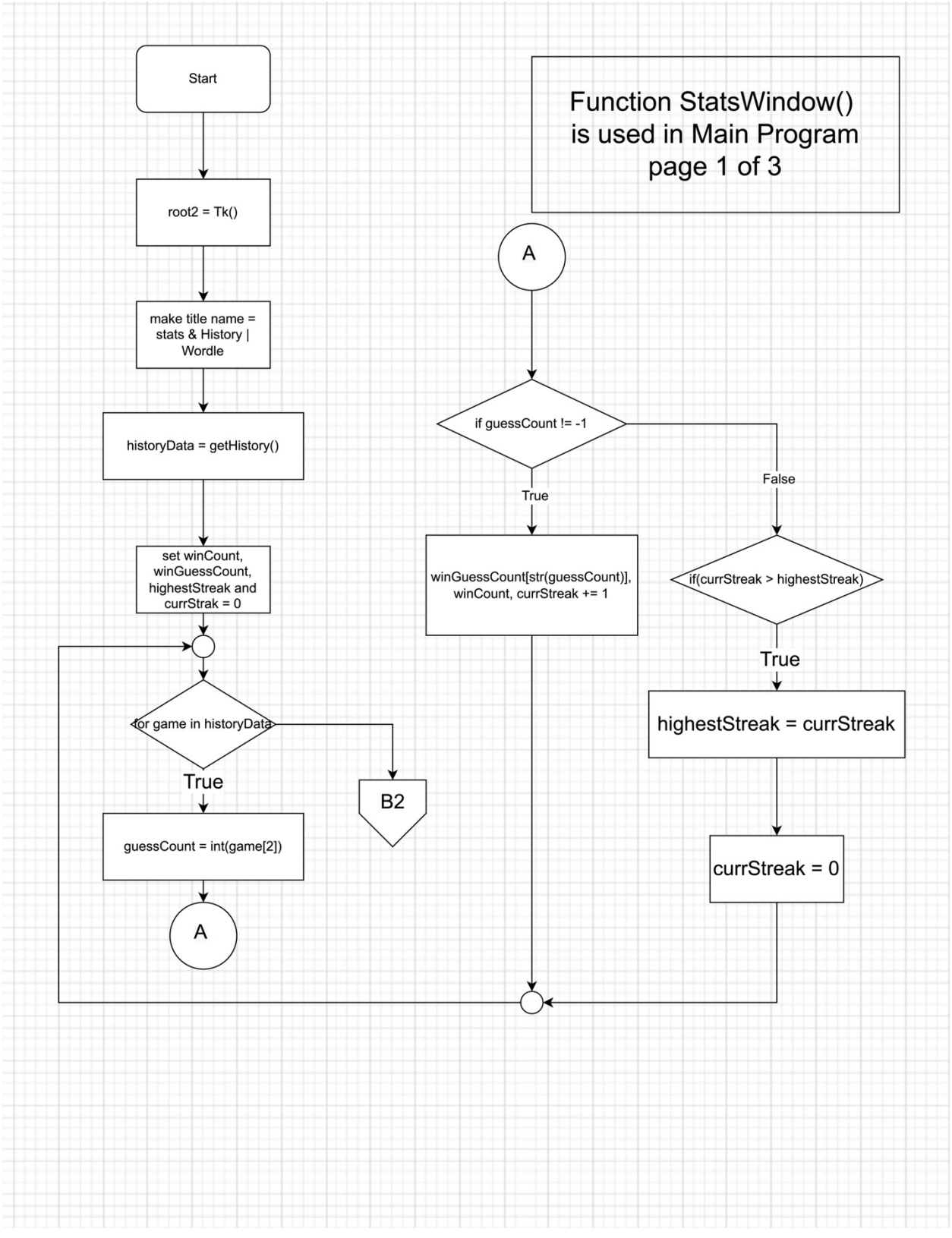
```
# print(len(historyData))              # For testing
# [print(row) for row in historyData]  # For testing

# One loop calculate all
winCount = 0
winGuessCount = {
    '1': 0,
    '2': 0,
    '3': 0,
    '4': 0,
    '5': 0,
    '6': 0
}
highestStreak = 0
currStreak = 0
for game in historyData:
    guessCount = int(game[2])

    # Winning
    if(guessCount != -1):
        winGuessCount[str(guessCount)] += 1
        winCount += 1

        currStreak += 1
    else:  # Losing
        if(currStreak > highestStreak):
            highestStreak = currStreak

        currStreak = 0

# Draw 'STATISTICS' Label
Label(root2, text='STATISTICS').grid(row=0, column=0, columnspan=4, pady=5)

# Draw Play count
playCount = len(historyData)
Label(root2, text=playCount).grid(row=1, column=0)
Label(root2, text='Played').grid(row=2, column=0)

# Draw Win rate
winRate = winCount / playCount
Label(root2, text=str(round(winRate * 100))).grid(row=1, column=1)
Label(root2, text='Win %').grid(row=2, column=1)

# Draw Current Streak
Label(root2, text=str(currStreak)).grid(row=1, column=2)
```

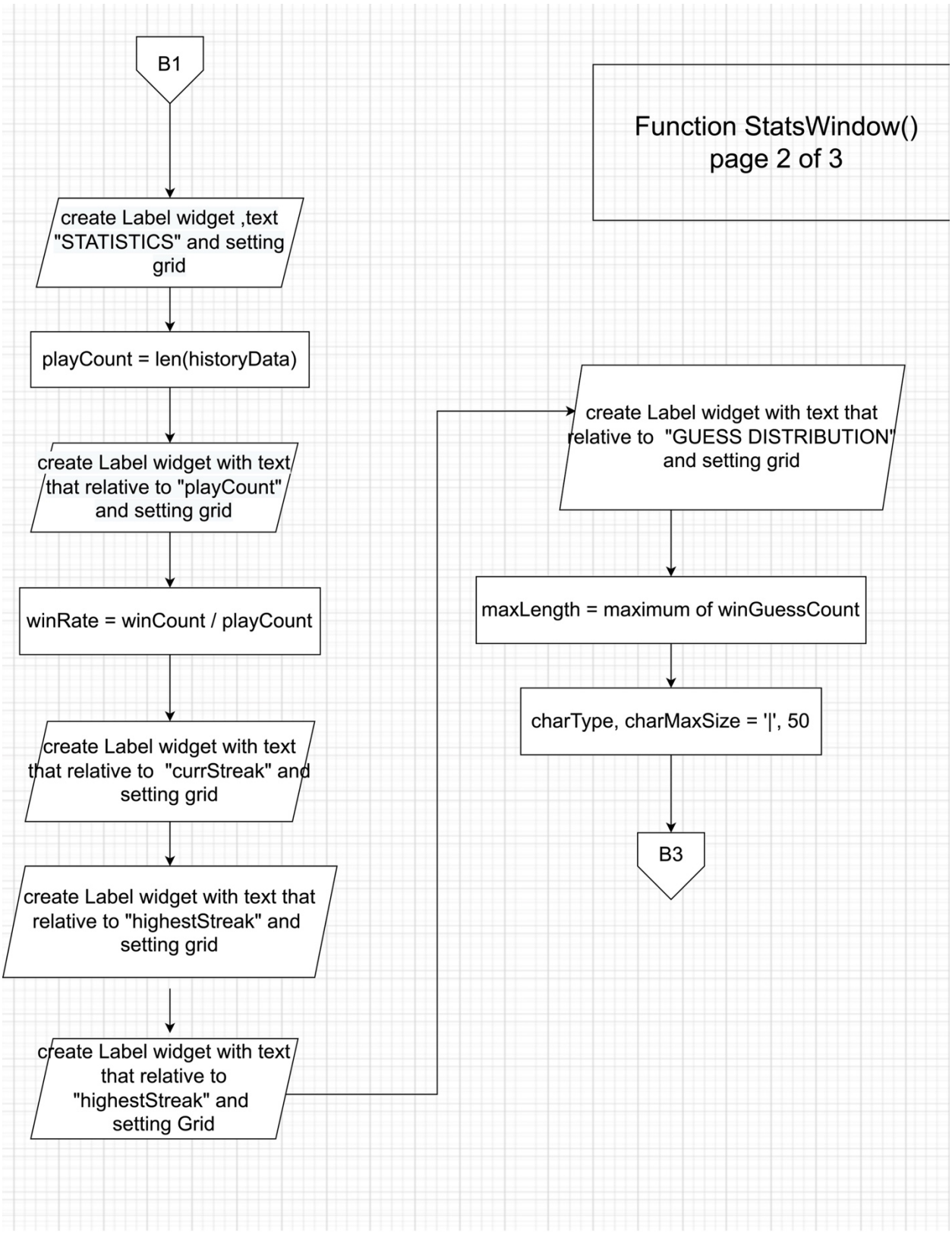
```
    Label(root2, text='Current Streak').grid(row=2, column=2)

    # Draw Max Streak
    Label(root2, text=str(highestStreak)).grid(row=1, column=3)
    Label(root2, text='Max Streak').grid(row=2, column=3)

    # Draw 'GUESS DISTRIBUTION' Label
    Label(root2, text='GUESS DISTRIBUTION').grid(
        row=3, column=0, columnspan=4, pady=5)
    maxLength = max([winGuessCount[key] for key in winGuessCount])
    charType, charMaxSize = '|', 50
    for i in winGuessCount:
        graphBar = charType * floor(charMaxSize * winGuessCount[i] / maxLength)
        idx = int(i) - 1

        Label(root2, text=i).grid(row=4+idx, column=0)
        Label(root2, text=f'{graphBar} {winGuessCount[i]}').grid(
            row=4+idx, column=1, columnspan=3, sticky='W')

    f.close()
    root2.mainloop()


if __name__ == '__main__':
    root = Tk()
    root.title('Wordle')

    root.rowconfigure(tuple(range(22)), weight=1, minsize=1)
    root.columnconfigure(tuple(range(22)), weight=1, minsize=1)

    # Draw Title
    Label(root, text='Wordle', font='Helvetica 24 bold').grid(
        row=1, column=8, rowspan=2, columnspan=6)

    # Stats Button
    Button(root, text='Stats', width=7, command=StatsWindow).grid(
        row=1, column=17, columnspan=4)

    # Draw Components
    initKeyboardGUI()
    initDisplay()

    # Load Words List
    try:
        f = open('words', 'r')
```

```
    wordsList = f.read().split('\n')
    f.close()
except:
    print('Can\'t Find Words list File, exiting...')
    exit()

print(f'Loaded {len(wordsList)} words')

# Main Game Cycle
gameCycle()

root.mainloop()
```

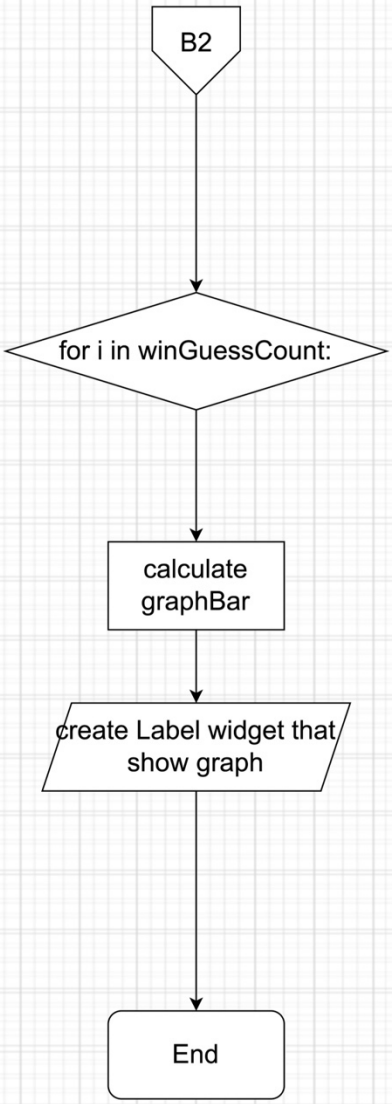
Flow Chart

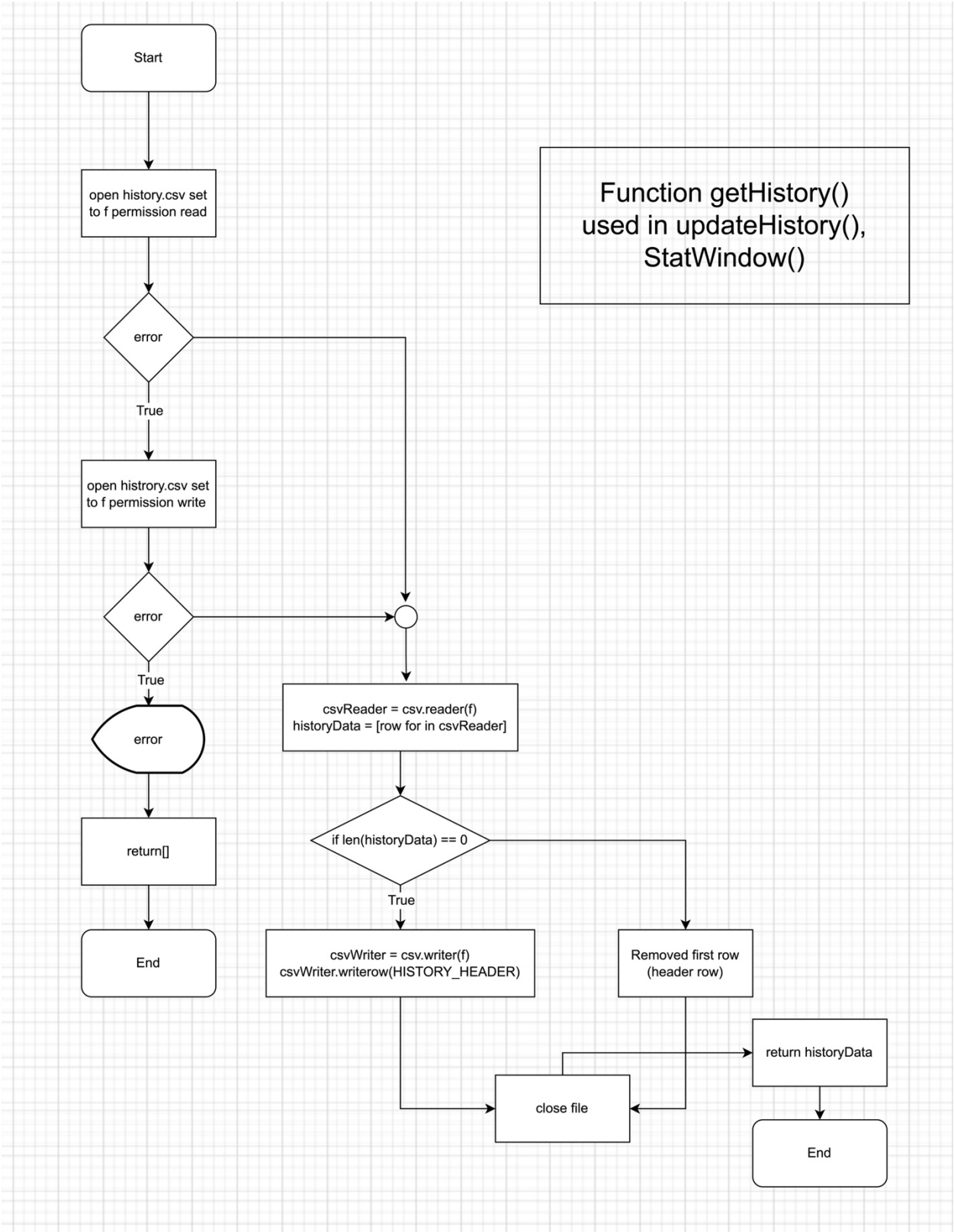

Start
import tkinter
import tkinter messagebox
import random
import csv
import time
declare Variable
if __name__ == "__main__"
True
False
set root = Tk()
set title to Wordle
confirgureRow and
column
create Wordle label
and Button that
command to
'StatsWindow'
initKeyboardGUI()
initDisplay()
open file word and
clean it then set it to 'f'
close file
A
A
if error
True
False
print cant find
word
print load .. word
End
gameCycle()
False
Close
True
End
MAIN PROGRAM

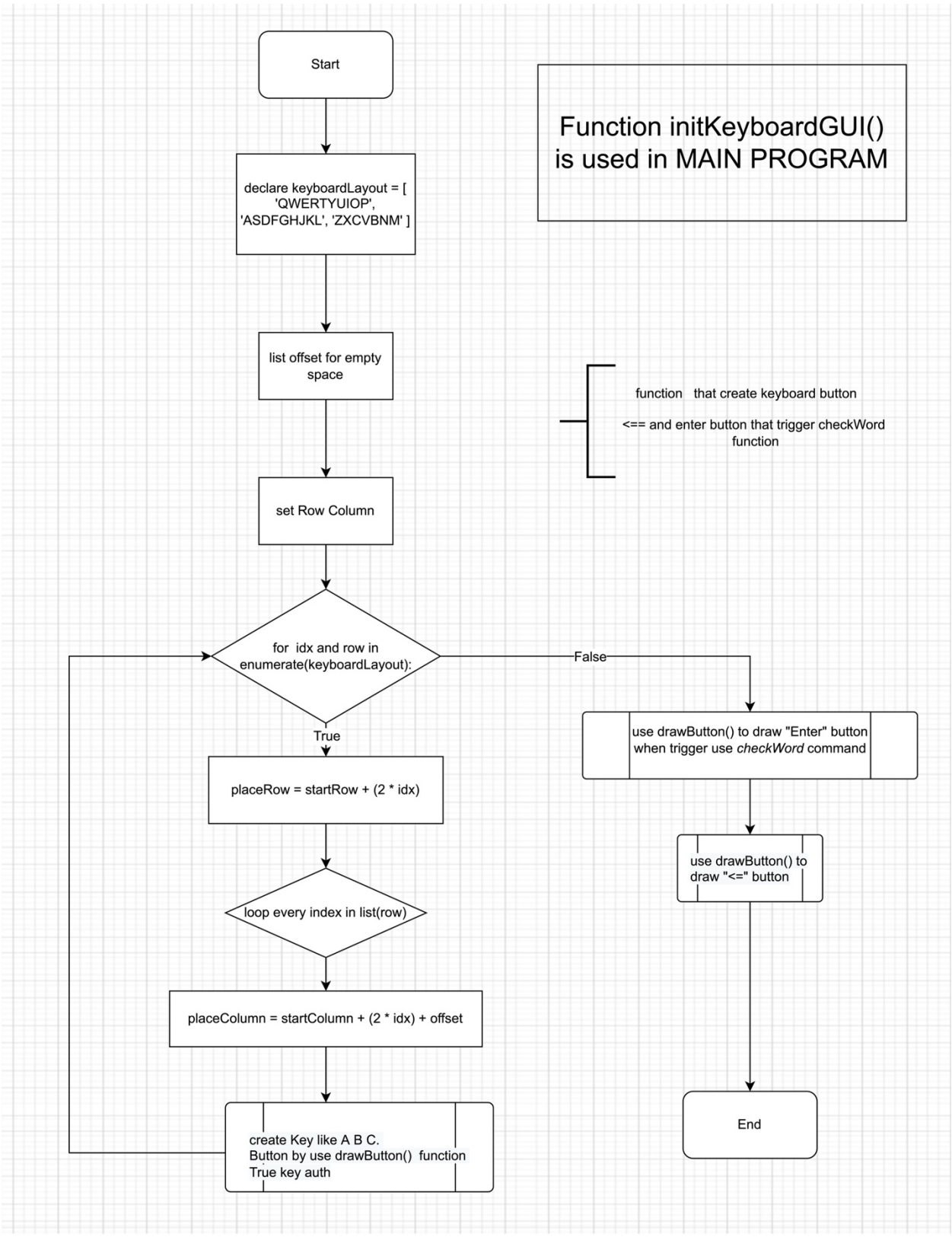

Function StatsWindow()
is used in Main Program
page 1 of 3
Start
root2 = Tk()
make title name = stats & History | Wordle
historyData = getHistory()
set winCount, winGuessCount, highestStreak and currStrak = 0
for game in historyData
True
guessCount = int(game[2])
A
B2
A
if guessCount != -1
True
False
winGuessCount[str(guessCount)], winCount, currStreak += 1
if(currStreak > highestStreak)
True
highestStreak = currStreak
currStreak = 0

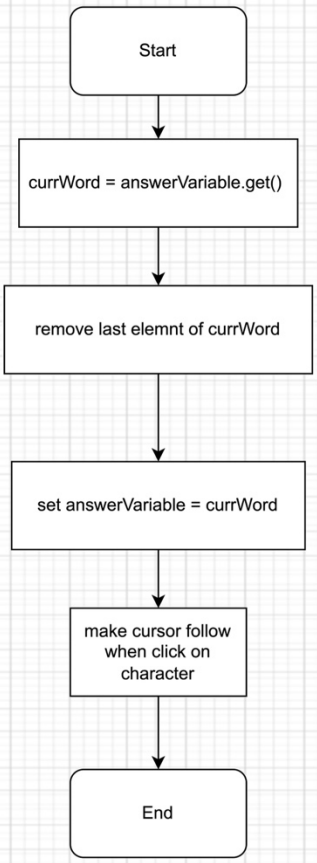

B1
Function StatsWindow()
page 2 of 3
create Label widget ,text "STATISTICS" and setting grid
playCount = len(historyData)
create Label widget with text that relative to "playCount" and setting grid
winRate = winCount / playCount
create Label widget with text that relative to "currStreak" and setting grid
create Label widget with text that relative to "highestStreak" and setting grid
create Label widget with text that relative to "highestStreak" and setting Grid
create Label widget with text that relative to "GUESS DISTRIBUTION" and setting grid
maxLength = maximum of winGuessCount
charType, charMaxSize = '|', 50
B3

Function StatsWindow()
page 3 of 3

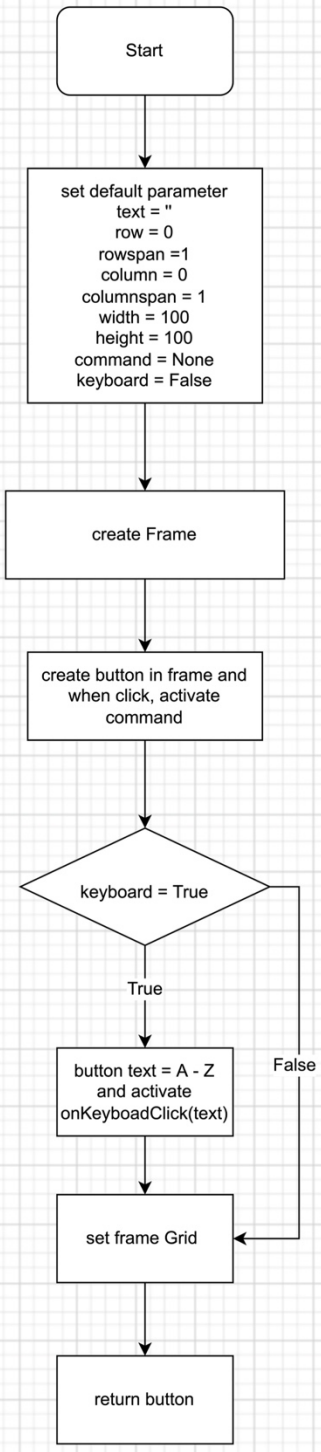



Start
open history.csv set to f permission read
error
True
open histrory.csv set to f permission write
error
True
error
return[]
End
csvReader = csv.reader(f)
historyData = [row for in csvReader]
if len(historyData) == 0
True
csvWriter = csv.writer(f)
csvWriter.writerow(HISTORY_HEADER)
Removed first row (header row)
close file
return historyData
End
Function getHistory() used in updateHistory(), StatWindow()

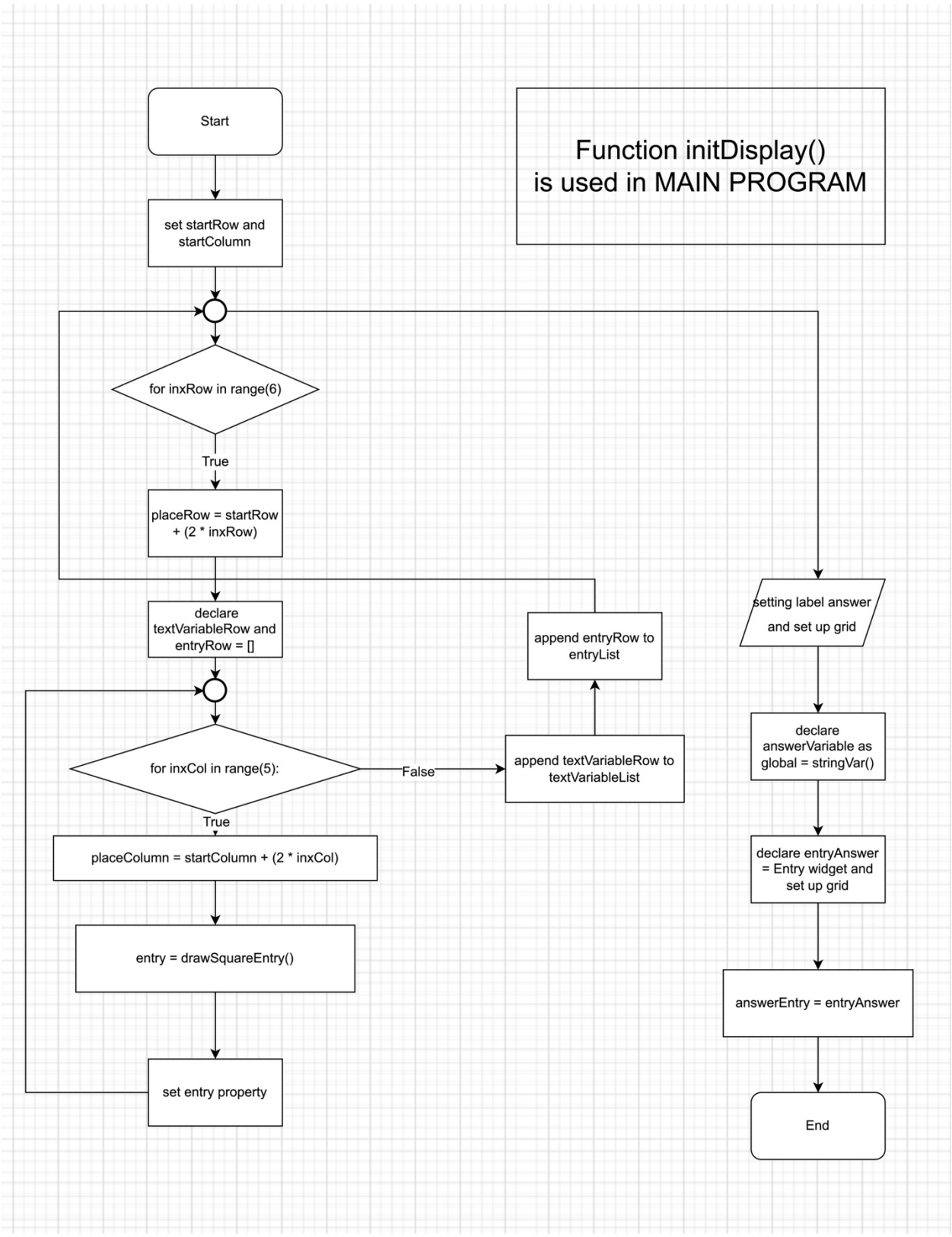

Start
declare keyboardLayout = ['QWERTYUIOP', 'ASDFGHJKL', 'ZXCVBNM']
list offset for empty space
set Row Column
for idx and row in enumerate(keyboardLayout):
True
placeRow = startRow + (2 * idx)
loop every index in list(row)
placeColumn = startColumn + (2 * idx) + offset
create Key like A B C.
Button by use drawButton() function
True key auth
False
use drawButton() to draw "Enter" button when trigger use checkWord command
use drawButton() to draw "<=" button
End
Function initKeyboardGUI() is used in MAIN PROGRAM
function that create keyboard button
<== and enter button that trigger checkWord function

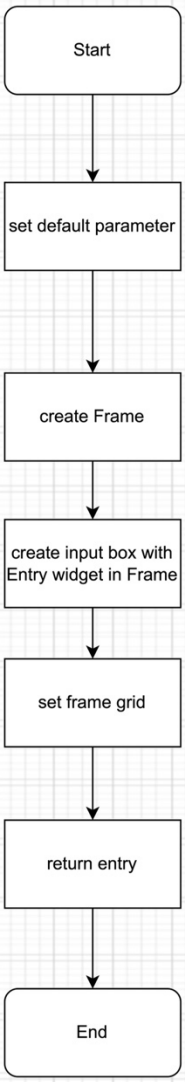


Function onReturn():
is used in initKeyboardGUI()

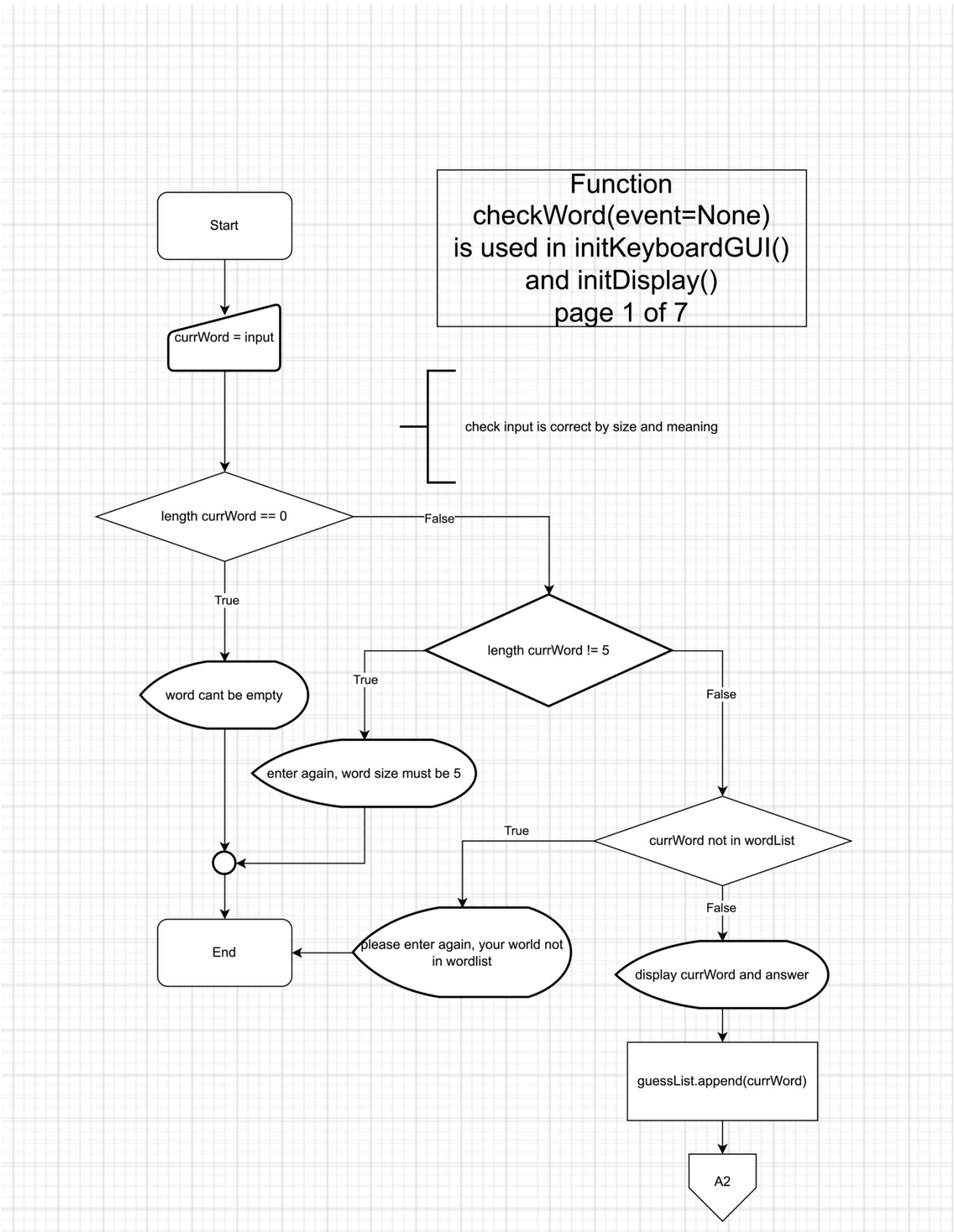


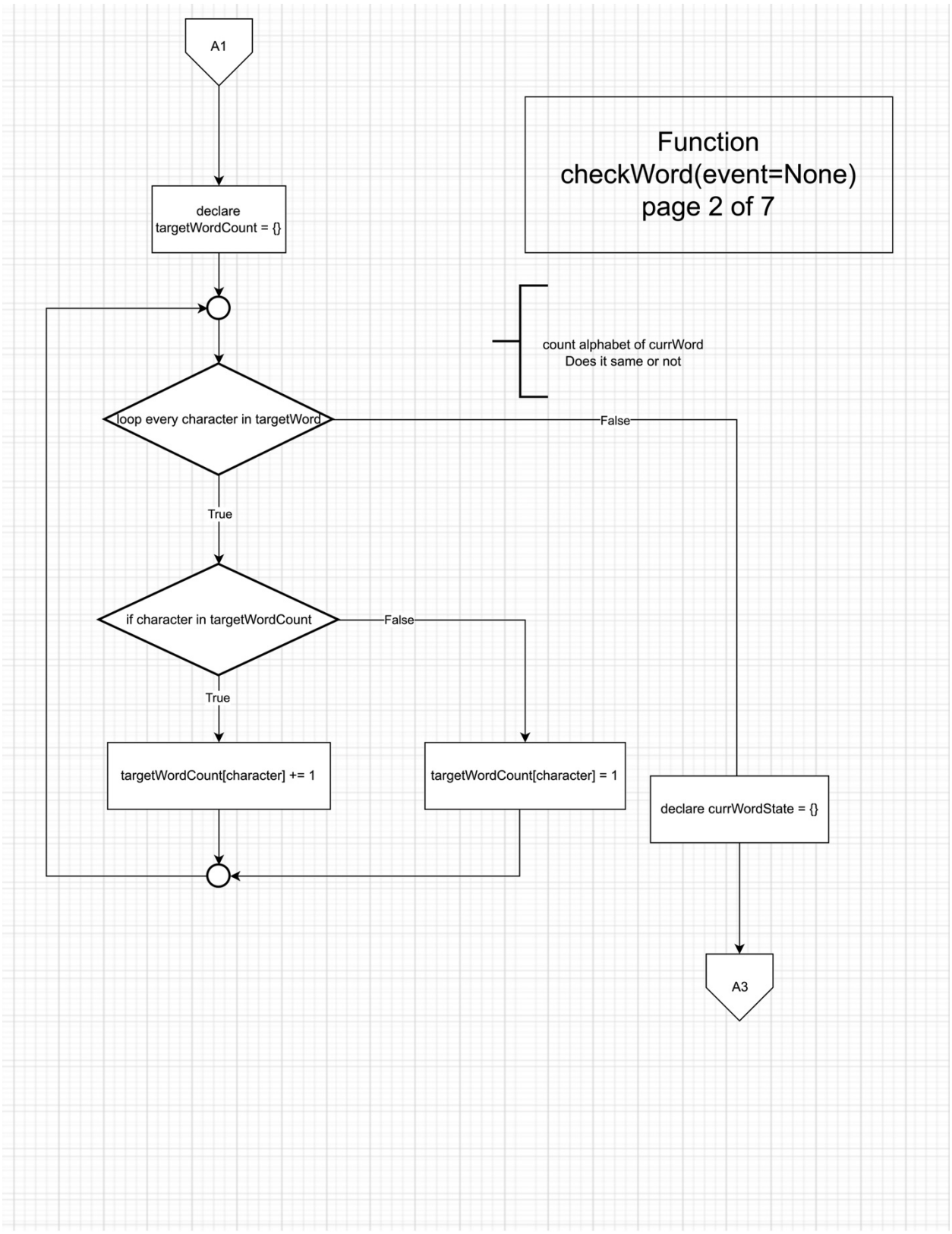
Function drawButton()
is used in initKeyboaedGui()
function

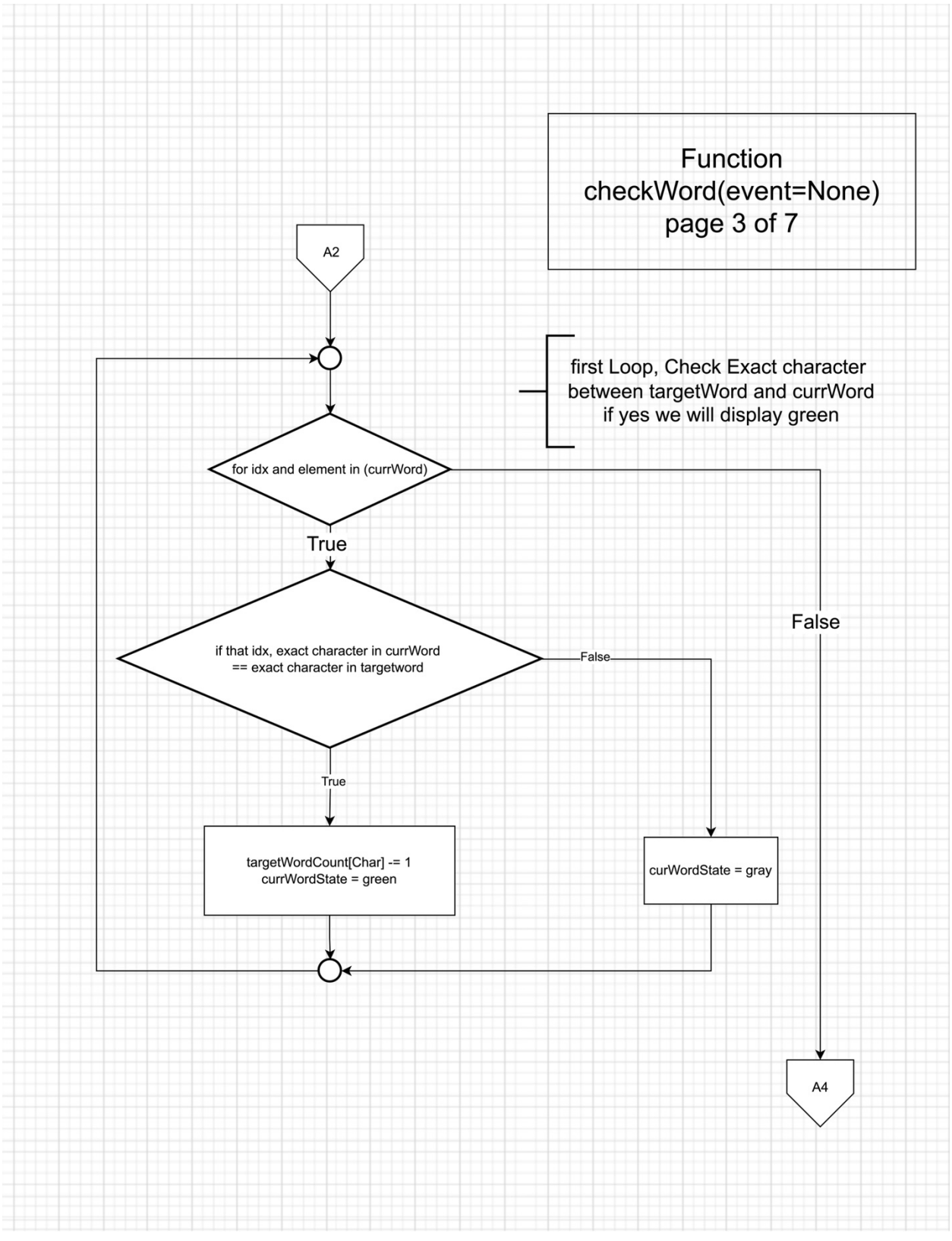

Start
set startRow and startColumn
for inxRow in range(6)
True
placeRow = startRow + (2 * inxRow)
declare textVariableRow and entryRow = []
for inxCol in range(5):
False
True
placeColumn = startColumn + (2 * inxCol)
entry = drawSquareEntry()
set entry property
append entryRow to entryList
append textVariableRow to textVariableList
Function initDisplay() is used in MAIN PROGRAM
setting label answer and set up grid
declare answerVariable as global = stringVar()
declare entryAnswer = Entry widget and set up grid
answerEntry = entryAnswer
End

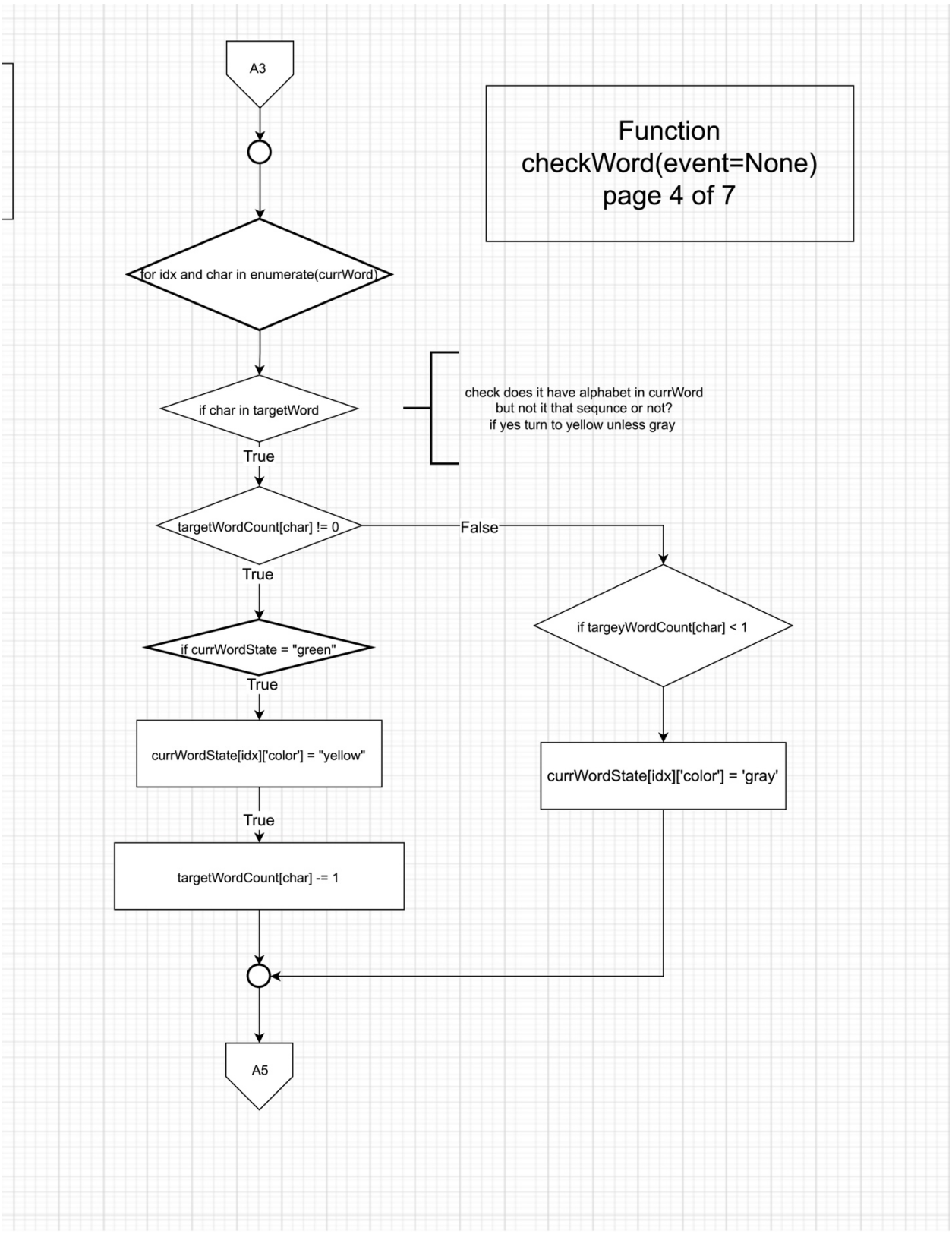


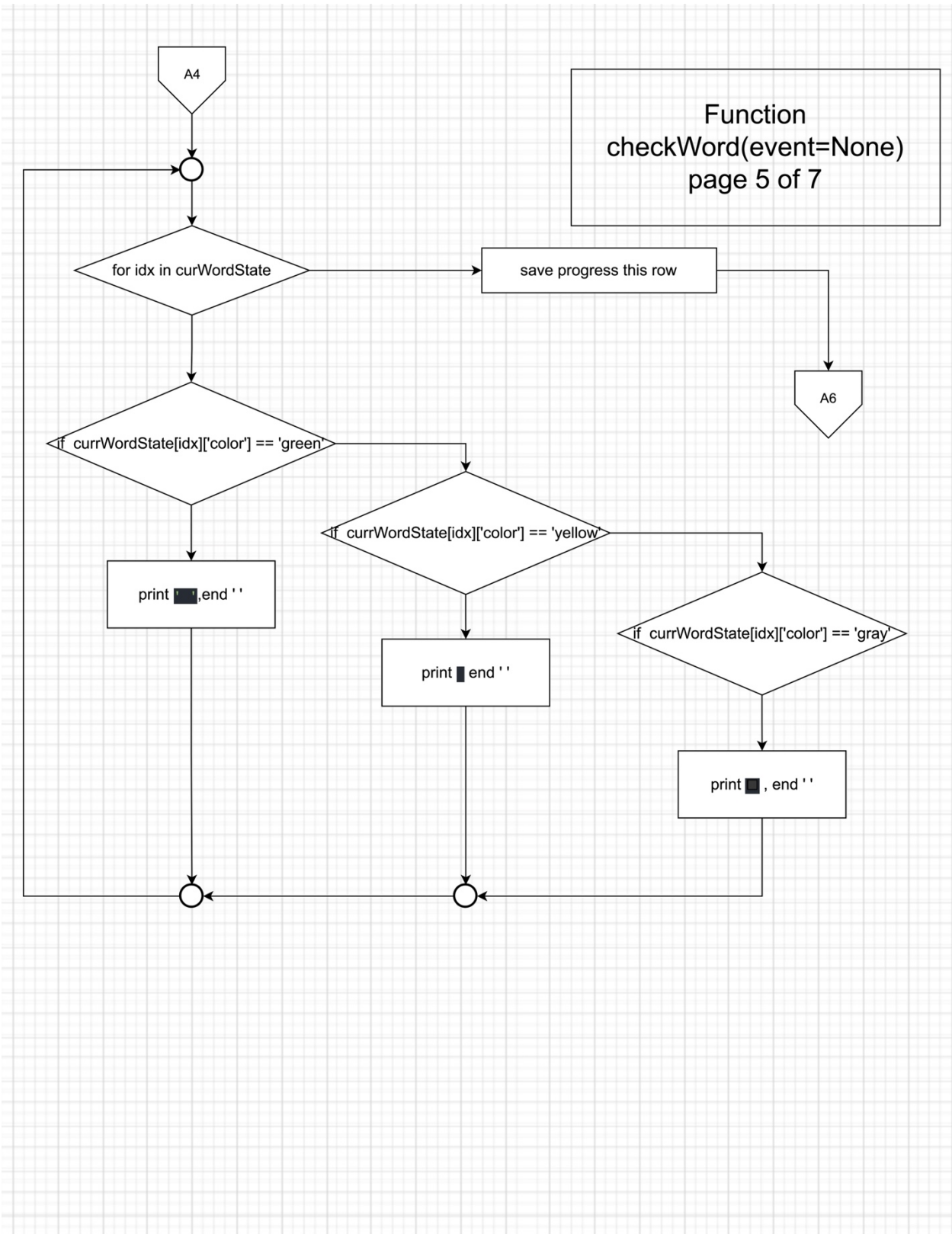
Function drawSquareEntry() is used in initDisplay()

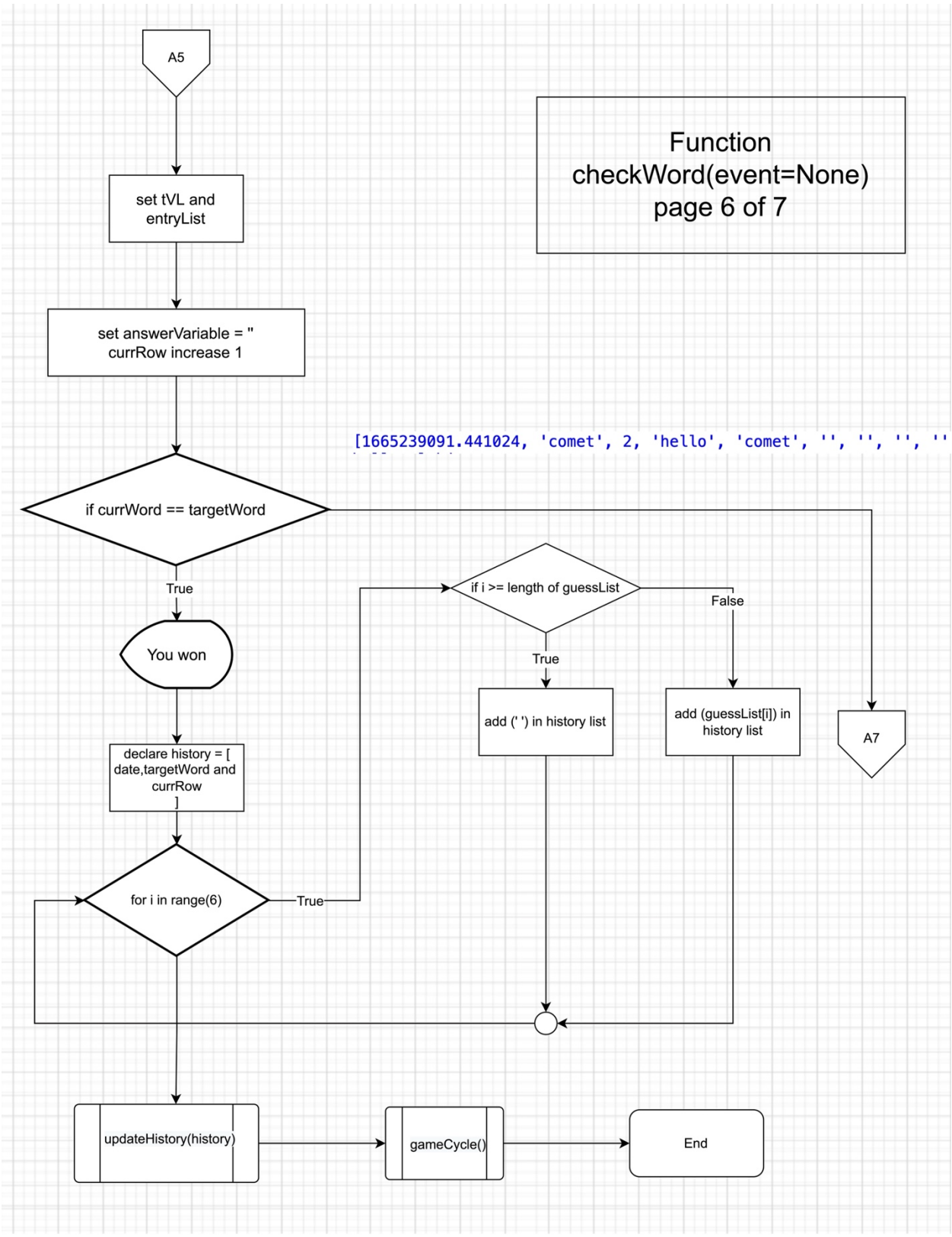

Function
checkWord(event=None)
is used in initKeyboardGUI()
and initDisplay()
page 1 of 7
Start
currWord = input
check input is correct by size and meaning
length currWord == 0
False
True
word cant be empty
length currWord != 5
True
False
enter again, word size must be 5
currWord not in wordList
True
False
End
please enter again, your world not in wordlist
display currWord and answer
guessList.append(currWord)
A2


A1
Function
checkWord(event=None)
page 2 of 7
declare
targetWordCount = {}
count alphabet of currWord
Does it same or not
loop every character in targetWord
False
True
if character in targetWordCount
False
True
targetWordCount[character] += 1
targetWordCount[character] = 1
declare currWordState = {}
A3

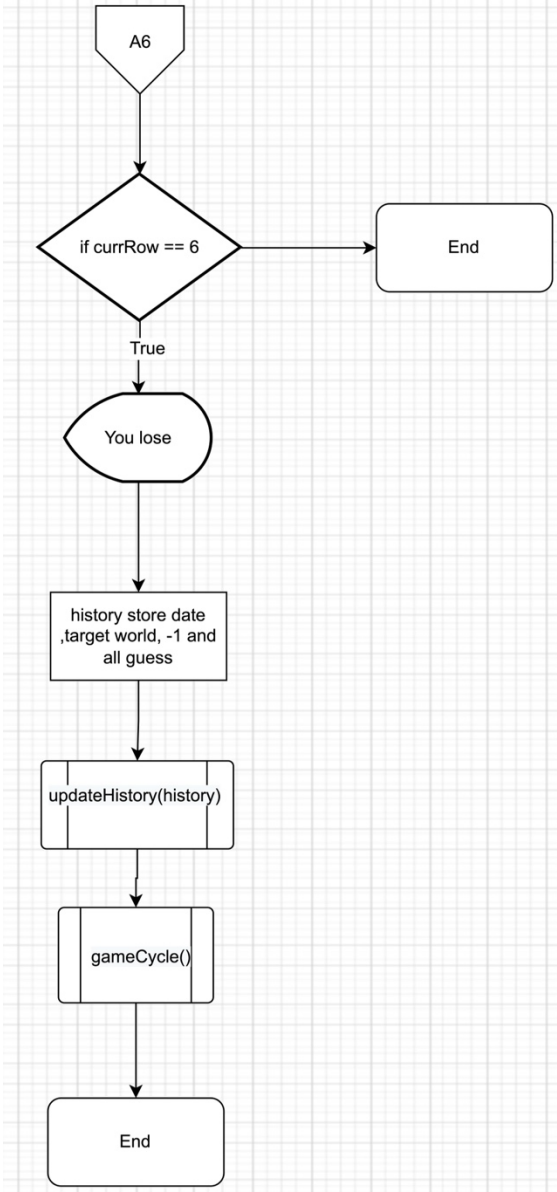

Function
checkWord(event=None)
page 3 of 7
A2
first Loop, Check Exact character
between targetWord and currWord
if yes we will display green
for idx and element in (currWord)
True
False
if that idx, exact character in currWord
== exact character in targetword
False
True
targetWordCount[Char] -= 1
currWordState = green
curWordState = gray
A4

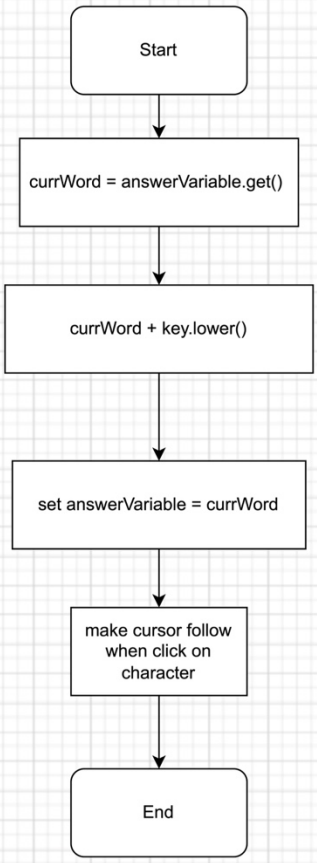

A3
Function
checkWord(event=None)
page 4 of 7
for idx and char in enumerate(currWord)
if char in targetWord
check does it have alphabet in currWord
but not it that sequnce or not?
if yes turn to yellow unless gray
True
targetWordCount[char] != 0
False
True
if targeyWordCount[char] < 1
if currWordState = "green"
True
currWordState[idx]['color'] = "yellow"
currWordState[idx]['color'] = 'gray'
True
targetWordCount[char] -= 1
A5


A4
Function
checkWord(event=None)
page 5 of 7
for idx in curWordState
save progress this row
A6
if currWordState[idx]['color'] == 'green'
if currWordState[idx]['color'] == 'yellow'
if currWordState[idx]['color'] == 'gray'
print ' ',end ' '
print end ' '
print , end ' '

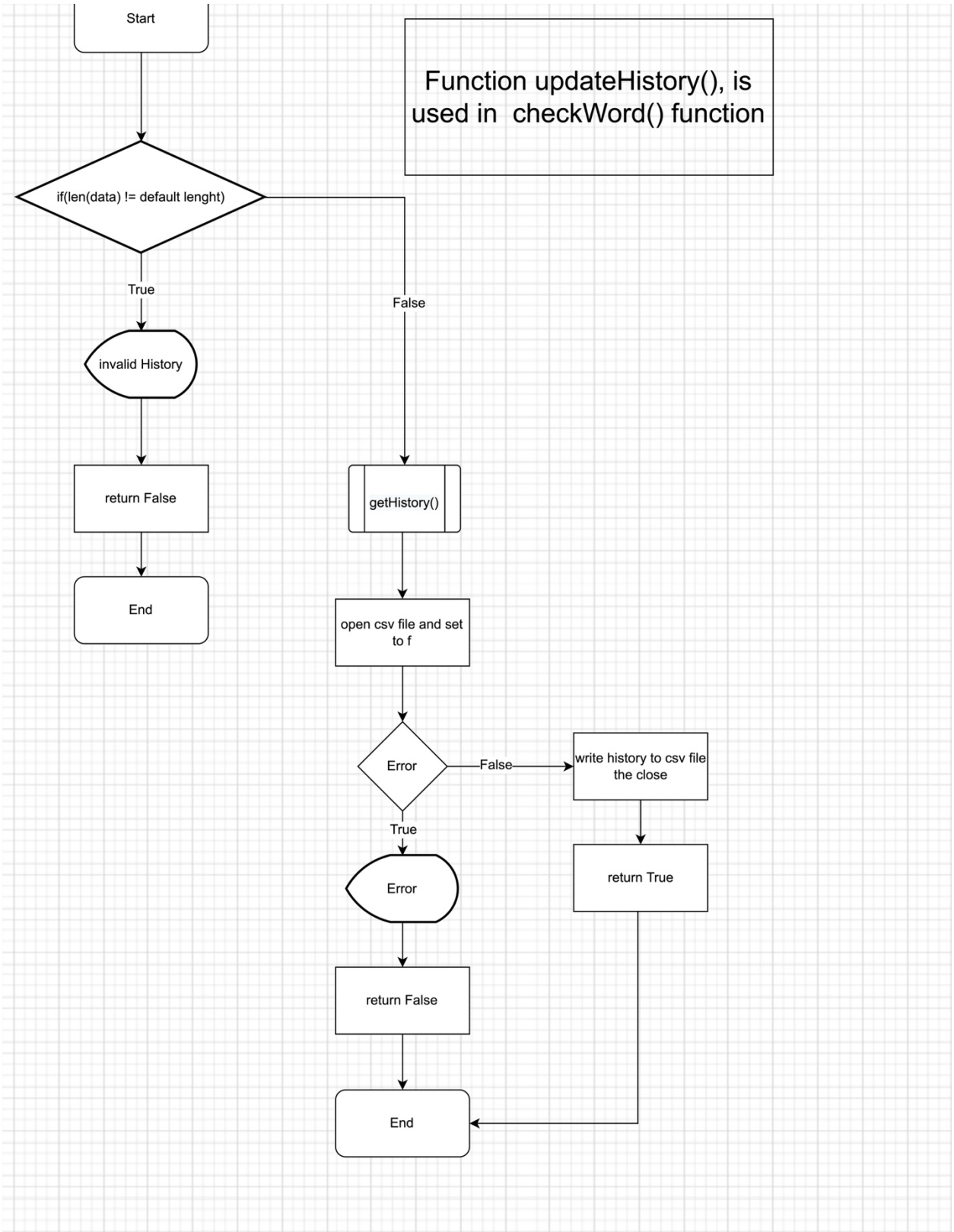

A5
Function checkWord(event=None) page 6 of 7
set tVL and entryList
set answerVariable = '' currRow increase 1
[1665239091.441024, 'comet', 2, 'hello', 'comet', '', '', '', ''
if currWord == targetWord
True
You won
declare history = [date,targetWord and currRow]
for i in range(6)
True
if i >= length of guessList
True
False
add (' ') in history list
add (guessList[i]) in history list
A7
updateHistory(history)
gameCycle()
End

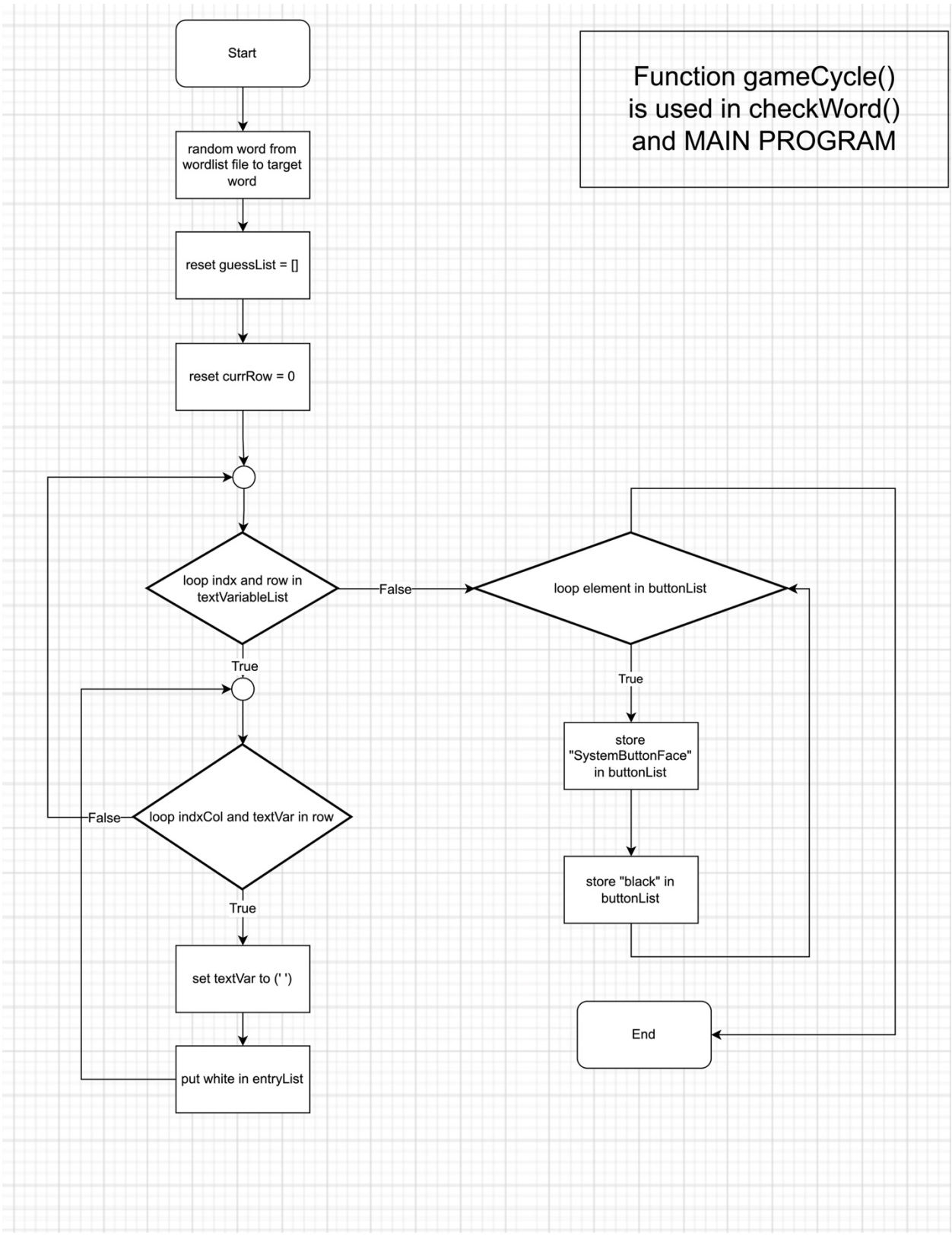
Function checkWord(event=None)
page 7 of 7





Function onKeyboardClick(): is used in drawButton().


Start
Function updateHistory(), is used in checkWord() function
if(len(data) != default lenght)
True
False
invalid History
return False
End
getHistory()
open csv file and set to f
Error
False
write history to csv file the close
True
Error
return True
return False
End


Start
random word from wordlist file to target word
reset guessList = []
reset currRow = 0
loop indx and row in textVariableList
False
loop element in buttonList
True
True
store "SystemButtonFace" in buttonList
store "black" in buttonList
End
loop indxCol and textVar in row
False
True
set textVar to (' ')
put white in entryList
Function gameCycle() is used in checkWord() and MAIN PROGRAM