```python
import tkinter.scrolledtext
from data import *
from process import process
from err import errCheckWithGui
from tkinter import filedialog
from tkinter import messagebox
import tkinter
import threading
from statusPanel import progressPanel
from statusPanel import errCheckProgressPanel
import sys

datas  = None
winnerShown = False
panel:progressPanel = None


def initMainFunc():
    global datas
    filePath = filedialog.askopenfilename()

    errPanel = errCheckProgressPanel()
    errFlag = False
    try:
        with open(filePath,"rt",encoding="UTF-8") as blob:
            errFlag = errCheckWithGui(blob,errPanel)
            if(not errFlag):
                datas = data(blob)
    except:
        errPanel.root.quit()
        messagebox.askyesno("err occured","during opening the data file, err occured.")
        errFlag = True

    if(errFlag):
        exit(0)


def getWinner(root:tkinter.Tk):
    global datas,winnerShown,panel
    if(winnerShown): return
    else: winnerShown = True

    def threadFunc(datas:data,panel:progressPanel):
        winner = process(datas,panel)
        if(winner != None):
            tkinter.Label(root,text="The winner is " + str(winner["id"])).pack()
        else:
            tkinter.Label(root,text="All candidates are excluded!").pack()

    processThread = threading.Thread(target=threadFunc,args=(datas,panel))
    processThread.start()



if __name__ == "__main__":
    initMainFunc()
    root = tkinter.Tk()

    root.title("Vote system")
    root.geometry("400x250")

    commandList = tkinter.LabelFrame(root,text="command button")
    commandList.pack(padx=10,pady=10)

    start = tkinter.Button(commandList,text="start process")
    start.bind("<1>",lambda event:getWinner(root))
    start.pack(padx=10,pady=10,side=tkinter.LEFT)

    end = tkinter.Button(commandList,text="exit")
    end.bind("<1>",lambda event:root.destroy())
    end.pack(padx=10,pady=10)

    panel = progressPanel(root)
    panel.showPanel()

    root.mainloop()



from io import TextIOWrapper
import string
from multiprocessing import Pool
import os
import tkinter
```

```python
from tkinter import ttk
from statusPanel import errCheckProgressPanel
import threading


# error check functions
# return:
# false -> no error found
# true  -> error found

def wordCheck(text:str,panel:errCheckProgressPanel) -> bool:
    panel.ProgressUpdate("wordCheck",{"progress":0})

    if("CANDIDATES = " in text[:256] and
       "VOTES = " in text[:256]): return False

    panel.ProgressUpdate("wordCheck",{"progress":100})
    return True

def valueCheck(text:str,panel:errCheckProgressPanel) -> bool:
    panel.ProgressUpdate("valueCheck",{"status":"start","progress":0})

    maxLength = ("BFGHJKLMPQRUWXYZ-" + string.ascii_lowercase).__len__()
    for index,Achar in enumerate("BFGHJKLMPQRUWXYZ-" + string.ascii_lowercase):
        panel.ProgressUpdate("valueCheck",{"status":"onProcess","index":index,"max":maxLength})
        if(Achar in text): return True

    panel.ProgressUpdate("valueCheck",{"status":"end","progress":100})
    return False

def checkVoteData(text:str,panel:errCheckProgressPanel) -> bool:
    panel.ProgressUpdate("checkVoteData",{"status":"start","progress":0})
    lines = text.split("\n")

    counter = 0
    numberOfCandidate = 0
    numberOfVote = 0
    for line in lines:
        element = line.split(" ")

        if(counter == 0):
            try:
                numberOfCandidate = int(element[2])
            except:
                return True
            counter += 1
            continue

        if(counter == 1):
            try:
                numberOfVote = int(element[2])
            except:
                return True
            counter += 1
            continue

        if(counter == 2):
            counter += 1
            continue

        votes = []
        for i in range(numberOfCandidate):
            try:
                votes.append(int(element[i]))
            except:
                return True

        checked = []
        for index,aVote in enumerate(votes):
            panel.ProgressUpdate("checkVoteData",{"status":"onProcessVoteRow","index":index,"numberOfCandidate":numberOfCandidate})
            for check in checked:
                if(aVote == check): return True
            checked.append(aVote)

        if(counter == numberOfVote): break
        panel.ProgressUpdate("checkVoteData",{"status":"onProcessEntire","counter":counter,"numberOfVote":numberOfVote})
        counter += 1

    panel.ProgressUpdate("checkVoteData",{"status":
```

```
"end","progress":100})
    return False


def errCheckWithGui(blob:TextIOWrapper,panel:errCheckProgressPanel):
    # panel = errCheckProgressPanel()
    status = {"status":"None","err":False}
    check = threading.Thread(target=errCehck,args=(blob,panel,status))
    check.start()
    panel.ProgressShow()
    return status["err"]

# each error check should be separeted into function
# and then executed as multiprocess
def errCehck(blob:TextIOWrapper,panel:errCheckProgressPanel,status:dict):
    text = blob.read()

    # check include "CANDIDATES" and "VOTES" value
    if(wordCheck(text,panel)):
        status["err"] = True
        status["status"] = "wordCheck"
        panel.showErrMessage()
        panel.root.quit()
        return

    # check unwanted char are included
        # check is there negative value
        # check format correctness
    if(valueCheck(text,panel)):
        status["err"] = True
        status["status"] = "valueCheck"
        panel.showErrMessage()
        panel.root.quit()
        return

    # check dupulicate data in vote data row
    # check are there alphabets in vote data row
    if(checkVoteData(text,panel)):
        status["err"] = True
        status["status"] = "checkVoteData"
        panel.showErrMessage()
        panel.root.quit()
        return

    # check are there worng amount of data in vote date row

    blob.seek(0)
    panel.root.destroy()
    status["status"] = "end"
    return False
import csv
from io import TextIOWrapper
import asyncio
import tkinter
import threading
from tkinter import ttk

class data:

    def __init__(self,blob:TextIOWrapper) -> None:
        self.candidateList      = []
        self.voteList           = []

        self.numberOfCandidate  = 0
        self.numberOfVote       = 0

        self.initProgressPanel()

        threading.Thread(target=self.getData,args=(blob,)).start()
        self.showProgress()

    def initProgressPanel(self):
        self.progressPanel = tkinter.Tk()
        self.progressPanel.title("reading data...")

        self.progressStatus = {
            "readVote"          : tkinter.StringVar(self.progressPanel,"0"),
            "currentData"       : tkinter.StringVar(self.progressPanel,""),
            "numberOfCandidate" : tkinter.StringVar(self.progressPanel,"wait"),
            "numberOfVote"      :
```

```python
tkinter.StringVar(self.progressPanel,"wait"),
            "status"            :tkinter.StringVar(self.progressPanel,"None")
        }

        self.progressBarState = {
            "initStep"              :10,
            "countVoteStepLength"   :50,
            "initCandidateLength"   :40,
            "prevStep"              :tkinter.IntVar(self.progressPanel,0),
            "step"                  :tkinter.IntVar(self.progressPanel,0)
        }
        self.progressBar = ttk.Progressbar(self.progressPanel,maximum=100,variable=self.progressBarState["step"])

    def showProgress(self) -> None:
        tkinter.Label(self.progressPanel,textvariable=self.progressStatus["status"]).pack()
        frame = tkinter.Frame(self.progressPanel)
        frame.pack()

        #------
        infoPanel = tkinter.LabelFrame(frame,text="info")
        infoPanel.pack(padx=50,pady=10,side=tkinter.RIGHT)

        numberOfCandidate = tkinter.LabelFrame(infoPanel,text="number of candidate")
        numberOfCandidate.pack(padx=10,pady=10)
        tkinter.Label(numberOfCandidate,textvariable=self.progressStatus["numberOfCandidate"]).pack()

        numberOfVote = tkinter.LabelFrame(infoPanel,text="number of vote")
        numberOfVote.pack(padx=10,pady=10)
        tkinter.Label(numberOfVote,textvariable=self.progressStatus["numberOfVote"]).pack()
        #------


        #------
        progress = tkinter.LabelFrame(frame,text="Progress")
        progress.pack(padx=50,pady=10)

        readVote = tkinter.LabelFrame(progress,text="number of read vote")
        readVote.pack(padx=10,pady=10)
        tkinter.Label(readVote,textvariable=self.progressStatus["readVote"]).pack()

        currentData = tkinter.LabelFrame(progress,text="current read data")
        currentData.pack(padx=10,pady=10)
        tkinter.Label(currentData,textvariable=self.progressStatus["currentData"]).pack()
        #------

        self.progressBar.pack(fill=tkinter.X)
        self.progressPanel.mainloop()


    def updateProgress(self,mode:str) -> None:

        self.updateProgressBar(mode)

        if(mode == "countVote"):
            self.progressStatus["status"].set(mode)
            self.progressStatus["readVote"].set(str(self.counter))
            currentData = ""
            for index,vote in enumerate(self.voteList[-1]):
                if(index == 6):
                    currentData += "..."
                    break
                currentData += str(vote) + ","
            self.progressStatus["currentData"].set(currentData)
            return

        if(mode == "initCandidate"):
            self.progressStatus["status"].set(mode)
            return

        if(mode == "init"):
            self.progressStatus["status"].set(mode)
            self.progressStatus["numberOfVote"].set(str(self.numberOfVote))
            self.progressStatus["numberOfCandidate"].set(str(self.numberOfCandidate))
            return


    def updateProgressBar(self,mode):
        if(mode == "countVote"):
            progress = self.progressBarState["initStep"] + self.progressBarState["countVoteStepLength"] * float(self.counter) / self.numberOfVote
            self.progressBarState["step"].set(int(progress))
            self.progressBarState["prevStep"].set(self.progressBarState["step"].get())
            return

        if(mode == "initCandidate"):
            progress = self.progressBarState["initStep"] + self.progressBarState["prevStep"].get() + (self.progressBarState["countVoteStepLength"] * float(self.candidateIndex) / self.numberOfCandidate)
            self.progressBarState["step"].set(int(progress))
            return

        if(mode == "init"):
            self.progressBarState["step"].set(int(self.progressBarState["initStep"]))
            return


    def getData(self,blob:TextIOWrapper) -> None:
        reader = csv.reader(blob,delimiter=" ")

        self.counter = 0
        for index,row in enumerate(reader):
            if(index == 0):
                self.numberOfCandidate = int(row[2])
                self.updateProgress("init")
                continue

            if(index == 1):
                self.numberOfVote = int(row[2])
                self.updateProgress("init")
                continue

            if(index == 2):
                # do nothing
                continue

            new = []
            for i in range(self.numberOfCandidate):
                new.append(int(row[i]))
            self.voteList.append(new)

            if(self.counter == self.numberOfVote):
                break

            self.counter += 1
            self.updateProgress("countVote")


        self.candidateIndex = 0
        self.updateProgress("initCandidate")
        for i in range(self.numberOfCandidate):
            self.candidateIndex = i
            self.updateProgress("initCandidate")
            new = {"id":i + 1,"count":0,"exclude":False}
            self.candidateList.append(new)

        self.progressPanel.destroy()

    def resetCandidateListCount(self) -> None:
        for aCandidate in self.candidateList:
            aCandidate["count"] = 0

    def findAllCandidateExcluded(self) -> bool:
        for aCandidateInfo in self.candidateList:
            if(not aCandidateInfo["exclude"]): False
        return True

    def showInfo(self) -> None:
        print(self.voteList)
        print(self.candidateList)
        print(self.numberOfVote)
        print(self.numberOfCandidate)
from data import data
from multiprocessing import Pool
import os
import tkinter
from tkinter import ttk
import threading
from statusPanel import progressPanel

def findCandidate(data:data,id:int) -> dict:
    for AcandidateInfo in data.candidateList:
        if(AcandidateInfo["id"] == id):
            return AcandidateInfo

    return None

def findNoneExculde(data:data,vote) -> int:
    for aCandidate in vote:
        info = findCandidate(data,aCandidate)
        if(not info["exclude"]): return aCandidate
    return None

def countCaindidate(data:data,voteList) -> None:
    for aCandidateInfo in data.candidateList:
        for aCandidate in voteList:
            if(aCandidate == aCandidateInfo["id"]):
                aCandidateInfo["count"] += 1

def findWinner(data:data) -> dict:
    threshold = int(data.numberOfVote / 2) + 1
    for aCandidateInfo in data.candidateList:
        if(aCandidateInfo["count"] >= threshold and not aCandidateInfo["exclude"]):
            return aCandidateInfo
    return None

def findExclude(data:data) -> None:
    minVote = data.candidateList[0]

    counter = 1
    while(minVote["exclude"]):
        minVote = data.candidateList[counter]
        counter += 1

    for aCandidateInfo in data.candidateList[counter:]:
        if(minVote["count"] > aCandidateInfo["count"] and not aCandidateInfo["exclude"]):
            minVote = aCandidateInfo

    minVote["exclude"] = True



def process(data:data,statusPanel:progressPanel) -> str:

    winner = None

    counter = 0
    while(1):
        voteList = []

        for vote in data.voteList:
            voteList.append(findNoneExculde(data,vote))
        print("find non excluded candidate")
        statusPanel.updateProgress("findNonExclude")

        # slow
        countCaindidate(data,voteList)
        print("count candidate")
        statusPanel.updateProgress("countCandidate")

        winner = findWinner(data)
        print("found winner")
        statusPanel.updateProgress("foundWinner")

        if(winner != None): break
        findExclude(data)
        print("find to exclude")
        statusPanel.updateProgress("findToExclude")

        data.resetCandidateListCount()
        print("reset vote counter of candidates")
        statusPanel.updateProgress("resetVote")

```

```python
        if(not data.findAllCandidateExcluded()):
            return None
        print("check all candidates are excluded")
        statusPanel.updateProgress("checAllCandidate")

        print("progress: " + str(counter) + "\n\n")
        counter += 1

    return winner

import tkinter
from data import *
import threading
from tkinter import messagebox



class errCheckProgressPanel:
    def __init__(self) -> None:
        self.ProgressInit()

    def ProgressInit(self) -> None :
        self.root = tkinter.Tk()
        self.root.title("error check status")
        self.root.geometry("400x200")

        self.progressStatus = {
            "status": tkinter.StringVar(self.root,"please wait..."),
            "progress": tkinter.IntVar(self.root,0),
            "checkVoteData":tkinter.IntVar(self.root,0)
        }

        self.progressBar = ttk.Progressbar(self.root,variable=self.progressStatus["progress"],maximum=100)

        self.checkVoteDataBar = ttk.Progressbar(self.root,variable=self.progressStatus["checkVoteData"],maximum=100)
        self.checkVoteDataLabel = tkinter.Label(self.root,text="vote row check progress")


    def ProgressShow(self) -> None :
        status = tkinter.LabelFrame(self.root,text="status")
        status.pack(padx=10,pady=10)
        tkinter.Label(status,textvariable=self.progressStatus["status"]).pack()

        self.progressBar.pack(fill=tkinter.X)
        threading.Thread(self.root.mainloop()).start()

    def ProgressHide(self) -> None :
        self.root.destroy()

    def showCheckVoteDataInfo(self) -> None:
        self.checkVoteDataLabel.pack()
        self.checkVoteDataBar.pack(fill=tkinter.X)

    def hideCheckVoteDataInfo(self) -> None:
        self.checkVoteDataLabel.forget()
        self.checkVoteDataBar.forget()

    def showErrMessage(self):
        text = "err occured in " + self.progressStatus["status"].get() +" err check process\n"
        text += "Please check your data file is correct."
        messagebox.askokcancel(title="error",message=text)

    def ProgressUpdate(self,mode:str,data:dict) -> None :
        if(mode == "wordCheck"):
            self.progressStatus["status"].set(mode)
            self.progressStatus["progress"].set(data["progress"])

        # ------
        if(mode == "valueCheck"):
            if(data["status"] == "start"):
                self.progressStatus["status"].set(mode)
                self.progressStatus["progress"].set(data["progress"])

            if(data["status"] == "onProcess"):
                index = data["index"]
                maxVal = data["max"]
                progress = int(100 * (index / float(maxVal)))
                self.progressStatus["progress"].set(progress)

            if(data["status"] == "end"):
                self.progressStatus["progress"].set(data["progress"])
        # ------

        if(mode == "checkVoteData"):
            if(data["status"] == "start"):
                self.progressStatus["status"].set(mode)
                self.progressStatus["progress"].set(data["progress"])
                self.showCheckVoteDataInfo()

            if(data["status"] == "onProcessEntire"):
                self.progressStatus["status"].set(mode + " " + str(data["counter"]) + "/" + str(data["numberOfVote"]))
                counter = data["counter"]
                numberOfVote = data["numberOfVote"]
                progress = int(100 * ((counter - 2) / float(numberOfVote)))
                self.progressStatus["progress"].set(progress)

            if(data["status"] == "onProcessVoteRow"):
                index = data["index"]
                maxVal = data["numberOfCandidate"]
                progress = int(100 * (index / float(maxVal)))
                self.progressStatus["checkVoteData"].set(progress)

            if(data["status"] == "end"):
                self.progressStatus["status"].set(mode)
                self.progressStatus["progress"].set(data["progress"])
                self.hideCheckVoteDataInfo()


class progressPanel:

    def __init__(self,root:tkinter.Tk) -> None:
        self.root = root
        self.initProgress()

    def initProgress(self) -> None:
        self.processProgressPanel = tkinter.LabelFrame(self.root,text="process progress status")

        self.processStatus =  {
            "status": tkinter.StringVar(self.processProgressPanel,"None"),
            "progress": tkinter.IntVar(self.processProgressPanel,0),
            "findNonExclude": 10,
            "countCandidate": 20,
            "foundWinner": 30,
            "findToExclude": 40,
            "resetVote": 50,
            "checAllCandidate": 60,
            "total": 60
        }

        self.progressBar = ttk.Progressbar(self.processProgressPanel,variable=self.processStatus["progress"],maximum=self.processStatus["total"])

        #-----
        frameRow1 = tkinter.Frame(self.processProgressPanel)
        frameRow1.pack(padx=10,pady=10)

        self.status = tkinter.LabelFrame(frameRow1,text="status")
        self.status.pack()
        tkinter.Label(self.status,textvariable=self.processStatus["status"]).pack()
        #-----
        self.progressBar.pack(fill=tkinter.X)


    def updateProgress(self,mode:str) -> None:
        self.processStatus["progress"].set(self.processStatus[mode])
        self.processStatus["status"].set(mode)

    def showPanel(self) -> None:
        self.processProgressPanel.pack(fill=tkinter.X,padx=10,pady=10)




```

```python














def updatePanel(state:tkinter.LabelFrame,refList:list,datas:data) -> None:
    interval = int(1.0 / 5 * 1000)
    for ref,AcandidateInfo in zip(refList,datas.candidateList):
        # code of first line for the debug purpose
        ref["id"].set("id: " + "%4d" % (AcandidateInfo["id"]))
        ref["count"].set("count: " + "%4d" % (AcandidateInfo["count"]))
        ref["exclude"].set("exclude: " + str(AcandidateInfo["exclude"]))
    state.after(interval,lambda :updatePanel(state,refList,datas))


def candidateStatePanel(root:tkinter.Tk,datas:data) -> None:
    mainFrame = tkinter.Frame(root)
    mainFrame.pack()

    scrollContainer = tkinter.Canvas(mainFrame)
    scrollContainer.grid(padx=20,pady=20,row=0,column=0)

    scroll = tkinter.Scrollbar(mainFrame,command=scrollContainer.yview)
    scroll.grid(row=0,column=1,sticky=tkinter.NS)
    scrollContainer.configure(yscrollcommand=scroll.set)

    state = tkinter.LabelFrame(scrollContainer,text="candidate state")
    state.pack(padx=20,pady=20)

    refList = []
    # add candidate
    for AcandidateInfo in datas.candidateList:
        ref = {}
        frame = tkinter.Frame(state)
        frame.pack()

        candidateID = tkinter.StringVar(state,"id: " + "%4d" % (AcandidateInfo["id"]))
        count       = tkinter.StringVar(state,"count: " + "%4d" % (AcandidateInfo["count"]))
        exclude     = tkinter.StringVar(state,"exclude: " + str(AcandidateInfo["exclude"]))

        ref["id"]       = (candidateID)
        ref["count"]    = (count)
        ref["exclude"]  = (exclude)

        tkinter.Label(frame,textvariable=candidateID).pack(side=tkinter.LEFT)
        tkinter.Label(frame,textvariable=count).pack(side=tkinter.LEFT)
        tkinter.Label(frame,textvariable=exclude).pack()

        refList.append(ref)

    state.update_idletasks()
    root.update_idletasks()
    scrollContainer.create_window(0,0,window=state,anchor="nw")
    scrollContainer.config(scrollregion=(0,0,state.winfo_reqwidth(),state.winfo_reqheight()))
    if(state.winfo_reqheight() > 200):
        scrollContainer.configure(width=state.winfo_reqwidth(),height=200)
    else:
        scrollContainer.configure(width=state.winfo_reqwidth(),height=state.winfo_reqheight())

    updater = threading.Thread(target=updatePanel,args=(state,refList,datas))
    updater.start()
```