

FOUNDATIONS OF PROGRAMMING ASSIGNMENT 3

**Submitted by: Submitted to:**

Ravideep Kaur (U1140816) DR. TIANNING LI

Parul Sharma (U1149906) **Lecturer** (Computing)

**Course Code:** CSC5020

**SOURCE CODE**

records =[]

# ---------------check concurrent Time ---------------

def checkConcurrentTime(sTime,eTime,targetStartTime,targetEndTime):

'''

This function is a helper function that is finds if the time is concurrent to any other function.

Parameters:

sTime(string) : starting time of the appointment time.

eTime(string) : ending time of the appointment time.

targetStartTime(string) : starting time of the other appointment time on the same date.

targetEndTime(string) : ending time of the other appointment time on the same date.

'''

if int(sTime) in range(int(targetStartTime),int(targetEndTime)+1):

print("This time slot on the same date is occupied by other appointment.So Please enter different time slot\nor select any other date")

return True

elif int(eTime) in range(int(targetStartTime),int(targetEndTime)+1):

print("This time slot on the same date is occupied by other appointment.So Please enter different time slot\nor select any other date")

return True

return False

def isConcurrentAppointment(date,sTime,eTime,records):

'''

This function is used to check if the given time is concurrent to any other appointment time on a particular date

This function also takes the help of the other helper function i.e checkConcurrentTime

Parameters:

date(string) : a particular date against which is to be checked.

sTime(string) : starting time of the appointment

eTime(string) : ending time of the appointment

records(list) : list of all the records.

Returns:

boolean : true if the particular time is concurrent to any other time on the same date.

'''

for record in records:

data = record.split(";")

if date == data[1]:

if checkConcurrentTime(sTime,eTime,data[2],data[3]):

return True

return False

# ---------- show record--------------

def showRecord(records):

'''

This Function Prints All the Records in the format that is described in the figure 2.

Parameters:

records(list) : List of All records

'''

if len(records) == 0:

print()

print("No Records with this Keyword found.")

print("Try with some other keyword.")

return

print("Priority\tDate\t\tStart\t\tEnd\t\tSubject")

print("--------\t----\t\t-----\t\t---\t\t-------------")

for record in records:

data = record.split(";")

print(data[0] if data[0].lower()=='high' else ' Low',end="\t\t")

print(data[1],end="\t")

print(data[2] if int(data[2])>9 else '0'+data[2],end="\t\t")

print(data[3] if int(data[3])>9 else '0'+data[3],end="\t\t")

print(data[4])

# ------------search record -------------

def showRecordWithKeyword(keyword,records):

'''

Helper function for the searchRecord function that segregate out records on the basis of particular keyword.

Parameteres:

keyword(string) : This value represents on the bases of which value(date,subject) we want to search the record.

records(list) : All records

'''

filtered\_data =[]

key = input("Please enter the keyword: ")

if keyword == 1:

for record in records:

data = record.split(";")

date = data[keyword]

dateSplit = date.split("/")

for datedata in dateSplit:

if key == datedata:

filtered\_data.append(record)

showRecord(filtered\_data)

return

for record in records:

data = record.split(";")

if key in data[keyword]:

filtered\_data.append(record)

showRecord(filtered\_data)

def searchRecord(records):

'''

This function search particular records on the basis of keyword given by the user.

Parameters:

records(list) : List of all records

'''

while True:

inp = input("Enter the keyword to search. eg. date,subject or end : ")

if inp.lower() =='date':

showRecordWithKeyword(1,records)

elif inp.lower()=='subject':

showRecordWithKeyword(4,records)

elif inp.lower()=='end':

print("Iteration has stopped.")

break

else:

print("Please enter a valid keyword fom [date,end,subject]")

# -------------sorting function --------------

def sortRecords(records):

'''

This function sort out the records on the basis of priority of the appointment.

Records with high priority are sorted first and the records with lower priority are sorted out later in the list.

Parameters:

records(list) : List of all the records.

'''

size = len(records)

for i in range(size):

for j in range(0,size-i-1):

""" Here i am comparing the rcords on basis of the priority"""

if records[j][0][0].lower() > records[j+1][0][0].lower():

records[j],records[j+1] = records[j+1],records[j]

# ---------------validation functions ---------------

def isValidTime(stime,etime):

'''

This function checks the time is valid or not based on the start time and end time provided by the user.

Parameters:

stime(string) : starting time of the appointment.

etime(string) : ending time of the appointment.

Returns:

boolean : if the time is valid it returns true else returns false.

'''

if int(stime) < int(etime) and 7 <= int(stime) <= 22 and 7<= int(etime) <=22:

return True

else:

print("-------------------------")

print("You Entered an invalid time. Please Enter time between 7 and 22.")

return False

days\_in\_month = [31,28,31,30,31,30,31,31,30,31,30,31]

def isLeapYear(year):

'''

This function is to check whether the year of the date is Leap year of not a leap year.

Parameters:

year(string) : year, which is to be checked

Returns:

boolean : if the year is leap it returns true else return false

'''

if (year % 400 == 0) and (year % 100 == 0):

return True

elif (year % 4 ==0) and (year % 100 != 0):

return True

return False

def isValidDate(st):

'''

This function is used to check whether the given date is valid or not.

Parameteres:

st(string) : given date

Returns:

boolean : return true if the given date is valid and return false if the given date is not valid.

'''

date = st.split("/")

if len(date)==3:

if int(date[2]) <10000 and int(date[2]) > 2022:

if 1 <= int(date[1]) <=12:

if int(date[1]) ==2:

if isLeapYear(int(date[2])):

if 1<= int(date[0]) <= 29:

return True

else:

return False

else:

if 1<= int(date[0]) <= 28:

return True

else:

return False

else:

if 1 <= int(date[0]) <= days\_in\_month[int(date[1])-1]:

return True

else:

return False

else:

return False

else:

return False

else:

return False

# --------Input function--------------

def getPriority():

priority = int(input("Enter 1 --> High Priority and 2 --> Low Priority."))

if priority == 1:

return "High"

elif priority == 2:

return "Low"

else:

print("You entered wrong input. Please Enter 1 or 2.")

return getPriority()

def getDate():

date = input("Enter appointment date. Please Enter date from next Year. eg. 23/5/2023 : ")

if isValidDate(date):

return date

else:

print("You have entered and Invalid date.\nPlease enter a valid date. eg. 23/5/2023")

return getDate()

def getStartingTime():

start\_time = input("Enter starting time = ")

return start\_time

def getEndingTime():

ending\_time = input("Enter the ending time = ")

return ending\_time

def getSubject():

subject = input("Enter the subject = ")

return subject

#------------- input data of specific user --------------

def addRecord():

global records

"""

This function will get the data from the user and store it in the list and return list.

"""

data = [getPriority(),getDate(),getStartingTime(),getEndingTime(),getSubject()]

if (isValidTime(data[2],data[3])==True) and (isConcurrentAppointment(data[1],data[2],data[3],records) == False):

return ";".join(data)

else:

print("Please enter the record again.")

return addRecord()

def printTable(key,data,freq):

'''

This function prints the tallyed records.

Paramters:

key(string) : keyword to print in heading

data(list) : segregated records values to be printed.

freq(list) : freq of the segregated values that is printed to the corresponding value from data.

'''

heading = ""

if key == 0:

heading =" Priority"

elif key==1:

heading = " Date"

else:

heading = " Year"

print(heading,"\t\tAppointments")

print("---------\t\t-----------")

for d,f in zip(data,freq):

#Maintaing the spaces by considering the length of data

#The length of string is set to 10 characters here.

st = ' '\*(10-len(d)) + str(d)

print(st,"\t\t",f)

def showTallyRecords(key,records):

'''

This is a helper function to tally out records from the given records on the basis of the given keyword by the user.

Parameters:

key(string) : Keyword on the basis of which the records are to be tallyed(date,priority,year)

records(list) : all records/appointments

Call another function printTable to print the segregated records.

'''

arr =[]

for record in records:

data = record.split(";")

if key==2:

arr.append(data[1].split("/")[2])

else:

arr.append(data[key])

arr.sort()

data =[]

freq =[]

i=-1

for item in arr:

if item in data:

freq[i]+=1

else:

i+=1

data.append(item)

freq.append(1)

printTable(key,data,freq)

def tallyRecords(records):

'''

This function the main function that tally records.

Parameters:

records(list) : list of all records.

'''

while True:

inp = input("Please enter the keywords from [date,priority,year,end] = ")

if inp.lower() == 'date':

showTallyRecords(1,records)

elif inp.lower() =='priority':

showTallyRecords(0,records)

elif inp.lower() =='year':

showTallyRecords(2,records)

elif inp.lower() =='end':

print("Iteration has stopped.")

break

else:

print("Please type a valid keyword from [date,priority,year,end]")

# ---------------- main function ----------------------

def main():

'''

Main function that is the root of the whole program.

'''

global records

print("Welcome to the schedule assignment.")

while True:

print("Press 1 to enter the record, 2 to exit.")

a = input()

if a=='1':

data = addRecord()

records.append(data)

elif a=='2':

print("Exiting Program.")

break

else:

print("Please press the valid key.")