class ParkData():

SUBROUTINE \_\_init\_\_(self, childCost: INTEGER, adultCost: INTEGER, seniorCost: INTEGER, wristbandCost: INTEGER, personsInPark: INTEGER, maxPersonsInPark: INTEGER):

self.childCost<- childCost

self.adultCost<- adultCost

self.seniorCost<- seniorCost

self.wristbandCost<- wristbandCost

self.personsInPark<- personsInPark

self.maxPersonsInPark<- maxPersonsInPark

ENDSUBROUTINE

SUBROUTINE getIterableData(self):

RETURN {

"childCost": self.childCost,

"adultCost": self.adultCost,

"seniorCost": self.seniorCost,

"wristbandCost": self.wristbandCost,

"personsInPark": self.personsInPark,

"maxPersonsInPark": self.maxPersonsInPark

}

ENDSUBROUTINE

SUBROUTINE addPersons(self, amount: INTEGER):

self.personsInPark <- self.personsInPark + amount

ENDSUBROUTINE

SUBROUTINE removePersons(self, amount: INTEGER):

self.personsInPark <- self.personsInPark - amount

ENDSUBROUTINE

SUBROUTINE changeData(self, dataToChange: STRING, value: INTEGER):

IF dataToChange = "childCost" THEN

self.childCost<- value

ELSEIF dataToChange = "adultCost" THEN

self.adultCost<- value

ELSEIF dataToChange = "seniorCost" THEN

self.seniorCost<- value

ELSEIF dataToChange = "wristbandCost" THEN

self.wristbandCost<- value

ELSEIF dataToChange = "personsInPark" THEN

self.personsInPark<- value

ELSEIF dataToChange = "maxPersonsInPark" THEN

self.maxPersonsInPark<- value

ENDIF

IF "Cost" in dataToChange THEN

OUTPUT f"{dataToChange} changed to £{value}.00"

ELSE

OUTPUT f"{dataToChange} changed to {value}"

ENDIF

ENDSUBROUTINE

import datetime

class ParkingPass():

SUBROUTINE \_\_init\_\_(self, name):

self.name<- name

ENDSUBROUTINE

SUBROUTINE getPass(self):

time<- datetime.datetime.now()

return f"""

TEMPORARY PARKING PASS FOR

{self.name}

ACTIVE FROM {time.day}/{time.month} {time.hour}:{time.minute}

FOR 12 HOURS

"""

ENDSUBROUTINE

class Ticket():

SUBROUTINE \_\_init\_\_(self, userData):

self.userData<- userData

ENDSUBROUTINE

SUBROUTINE getTicket(self):

name<- self.userData.surname.split(" ")

childTicket<- self.userData.children > 0 and f"Child ticket: {self.userData.children} £{12\*self.userData.children}.00" or ""

adultTicket<- self.userData.adults > 0 and f"\n Adult ticket: {self.userData.adults} £{20\*self.userData.adults}.00" or ""

seniorTicket<- self.userData.seniors > 0 and f"\n Senior ticket: {self.userData.seniors} £{11\*self.userData.seniors}.00" or ""

bands<- self.userData.bands > 0 and f"\n Wristbands: {self.userData.bands} £{20\*self.userData.bands}.00" or ""

ENDSUBROUTINE

ticket<- f"""

Copington Theme Park

Date: {date.today().strftime("%d/%m/%y")}

Item// QTY PRICE

--------------------------------------

{childTicket}{adultTicket}{seniorTicket}{bands}

--------------------------------------

SUBTOTAL: £{self.userData.calculateCost()}.00

AMOUNT PAYED £{self.userData.amountPayed()}.00

CHANGE: £{max(self.userData.amountPayed()-self.userData.calculateCost(), 0)}.00

LEAD BOOKER: {self.userData.surname.upper()}

"""

RETURN ticket

class UserData():

SUBROUTINE \_\_init\_\_(self, children: INTEGER, adults: INTEGER, seniors: INTEGER, bands: INTEGER, parking: BOOLEAN, surname: STRING, notes10: INTEGER, notes20: INTEGER):

self.children<- children

self.adults<- adults

self.seniors<- seniors

self.bands<- bands

self.parking<- parking

self.surname<- surname

self.notes10<- notes10

self.notes20<- notes20

ENDSUBROUTINE

SUBROUTINE setNotes(self, type: STRING, amount: INTEGER):

IF type = "notes10" THEN

self.notes10<- amount

ELSEIF type = "notes20" THEN

self.notes20<- amount

ELSE

raise("Invalid type submitted.")

ENDIF

ENDSUBROUTINE

SUBROUTINE calculateCost(self):

RETURN (self.children \* 12) + (self.seniors \* 11) + (self.adults \* 20) + (self.bands \* 20)

ENDSUBROUTINE

SUBROUTINE amountPayed(self):

RETURN self.notes10 \* 10 + self.notes20 \* 20

ENDSUBROUTINE

SUBROUTINE change(self):

RETURN self.amountPayed() - self.calculateCost()

ENDSUBROUTINE

SUBROUTINE main:

WHILE True DO

OUTPUT "Welcome to Copington Theme Park"

splitName<- getName().split(" ")

surname<- splitName[len(splitName)-1]

children<- getInt(

f"How many children are there? (£{PARKDATA.childCost} each): ")

adults<- getInt(

f"How many adults are there? (£{PARKDATA.adultCost} each): ")

seniors<- getInt(

f"How many seniors are there? (£{PARKDATA.seniorCost} each): ")

bands<- getInt(

f"How many wristbands do you want? (£{PARKDATA.wristbandCost} each): ")

IF tooManyPeople(children, adults, seniors)

THEN

OUTPUT "Sorry we are at capacity right now.\nPlease try again later."

ELSE

parking<- yesOrNo("Do you need a parking pass (y or n): ")

userData<- UserData(children, adults, seniors,

bands, parking, surname)

pay(userData)

ticket<- Ticket(userData)

OUTPUT ticket.getTicket(

IF userData.parking

THEN

OUTPUT ParkingPass(surname.getPass(

ENDIF

OUTPUT "Thank you FOR coming to Copington Theme Park" //Pseudocode can't handle this

ENDIF

WAIT 2 seconds

ENDWHILE

ENDSUBROUTINE

SUBROUTINE getInt(message):

valid<- False

WHILE not valid DO

INPUT integer<-INPUT(message)

TRY:

integer<- INTEGER(integer)

IF integer < 0 THEN

RAISE ValueError

ENDIF

valid<- True

EXCEPT ValueError:

OUTPUT "You must enter a whole number greater than 0"

ENDWHILE

RETURN integer

ENDSUBROUTINE