AQA GCSE Computer Science 8520 Programming Project Task Report

Component 3 – Final Report

Project Title:

Flight Planning Software

Student Name
ADITI BHAT

Supervisor: Mrs Ghali

Department of Computer Science
Tonbridge Grammar School

Declaration

Student Name: **ADITI BHAT** Year: **11**

Date of Submission: **27/12/20**

Table of Contents

Project Specification

AQA

GCSE
COMPUTER SCIENCE

Component 3 Programming Project Task

For candidates entering for the **June 2021** 8520 examination.

To be issued to candidates on 1 April 2020 or as soon as possible after that date.

Time allowed
- 20 hours

Instructions
- *Evidence must include a complete listing of all program code together with a report. The report* should describe the design of the solution, the testing and any potential enhancements and refinements to the solution.
- Students must use one of the following programming languages:
  - C#
  - Java
  - Pascal/Delphi
  - Python
  - VB.Net

Information
- The project is designed to be completed in 20 hours.
- The allocated time is not required to be continuous.
- There are restrictions on when and where students can work on this problem. Please see the Teachers' Notes which accompany this task for more information about these restrictions.
- Students may need to use the Internet to research certain parts of the problem. This must be within the 20 hours.
- Submission may be paper based or electronic using CD/DVD.
- Students will need to complete and sign a Candidate Record Form which declares that the work is their own. This must be countersigned by the teacher.
- Copyright permission is granted by AQA to use the copyright in the materials on the condition that such use is limited strictly to the personal use by each teacher and their students for the purpose of the preparation for and conduct of the programming project task only. The materials are not to be provided to anyone other than the teacher and the students undertaking the task. The teacher must collect this task back from the students at the end of each session. The use of the materials for the production and publication in any format of teaching materials or any other such material (other than for the teacher's personal use) is strictly forbidden.

Flight Planning

A new airline wants to start running commercial passenger flights. In order to assess the feasibility of proposed flights they want a program that can help them calculate the likely profitability of running a flight between a UK airport and an overseas airport. The UK airport will be either Liverpool John Lennon or Bournemouth International.

A comma separated (csv) file called **Airports.txt** has been provided that contains the name and code for each overseas airport, the distance from Liverpool John Lennon airport and the distance from Bournemouth International airport. The contents of this comma separated file are shown in **Figure 1**.

Figure 1

Overseas airport code	Overseas airport name	Distance from Liverpool John Lennon (km)	Distance from Bournemouth International (km)
JFK	John F Kennedy International	5326	5486
ORY	Paris-Orly	629	379
MAD	Adolfo Suarez Madrid-Barajas	1428	1151
AMS	Amsterdam Schiphol	526	489
CAI	Cairo International	3779	3584

When the program is used, the following details will need to be entered:

- UK airport
- overseas airport
- type of aircraft
- number of first-class seats
- price of a first-class seat
- price of a standard-class seat.

The airline owns three types of aircraft. **Figure 2** shows the characteristics of each type of aircraft.

Figure 2

Type	Running cost per seat per 100 km	Maximum flight range (km)	Capacity if all seats are standard-class	Minimum number of first-class seats (if there are any)
Medium narrow body	£8	2650	180	8
Large narrow body	£7	5600	220	10
Medium wide body	£5	4050	406	14

3

From the details provided by the user, the program will calculate the potential profit or loss of running the proposed flight assuming it is at maximum capacity. **Figure 3** shows how the profit or loss is calculated. Each first-class seat takes up space for two standard-class seats.

Figure 3

Number of standard-class seats	Capacity if all seats are standard-class – Number of first-class seats x 2
Flight cost per seat	running cost per seat per 100 km (for the selected type of aircraft) × distance between the UK airport and the overseas airport / 100
Flight cost	flight cost per seat × (number of first-class seats + number of standard-class seats)
Flight income	number of first-class seats × price of a first-class seat + number of standard-class seats × price of a standard-class seat
Flight profit	flight income - flight cost

Figure 4 shows an example.

Figure 4

UK airport	Liverpool John Lennon
Overseas airport	John F Kennedy International
Distance	5326 kilometres
Type of aircraft	Large narrow body
Maximum flight range	5600 kilometres
Running cost per seat per 100 kilometres	£7
Capacity if all seats are standard-class	220
Number of first-class seats	40
Number of standard-class seats: 220 - 40 × 2	140
Price of a standard-class seat	£400
Price of a first-class seat	£1200
Flight cost per seat: 7 × 5326/100	£372.82
Flight cost: 372.82 × (140 + 40)	£67 107.60
Flight income: 140 × 400 + 40 × 1200	£56 000 + £48 000 = £104 000
Flight profit: 104 000 - 67 107.6	£36 892.40

The user will also have the option to clear all the data they have entered and start again.

Turn over ▶

4

The program should work in the following way:

1. The program should read in the comma separated file **Airports.txt**

2. A menu should be displayed allowing the user to select from the following options:
   - Enter airport details
   - Enter flight details
   - Enter price plan and calculate profit
   - Clear data
   - Quit

3. If the user selects the 'Quit' option then a suitable message should be displayed and the program ends.

4. If the user selects the 'Enter airport details' option:
   a. the user should be asked to enter the three-letter airport code for the UK airport
   b. if the code entered is not for Liverpool John Lennon (LPL) or Bournemouth International (BOH) then a suitable error message should be displayed and the user returned to the main menu
   c. the user should then be asked to enter the three-letter airport code for the overseas airport
   d. the program should check that the code for the overseas airport is valid based on the data read from the csv file
      - if the code for the overseas airport is valid then the full name of the overseas airport should be displayed
      - if the code for the overseas airport is not valid then a suitable error message should be displayed
   e. the user should be returned to the main menu.

5. If the user selects the 'Enter flight details' option:
   a. the user should be asked to enter the type of aircraft to be used. The allowed types are shown in **Figure 2**
   b. if the type of aircraft is not valid then a suitable error message should be displayed and the user returned to the main menu
   c. the data in **Figure 2** for that type of aircraft should then be displayed
   d. the user should then be asked to enter the number of first-class seats on the aircraft
   e. if the number of first-class seats entered is not 0 then:
      - if the number of first-class seats is less than the 'minimum number of first-class seats' then a suitable error message should be displayed and the user returned to the main menu
      - if the number of first-class seats is greater than half the 'capacity if all seats are standard-class' then a suitable error message should be displayed and the user returned to the main menu.
   f. the program should then calculate the number of standard-class seats on the aircraft using the formula:

      'Capacity if all seats are standard-class – Number of first-class seats x 2'

   g. the user should be returned to the main menu.

5

6. If the user selects the 'Enter price plan and calculate profit' option:
   a. the program should check that codes for the UK and overseas airports have been entered. If not then a suitable error message should be displayed and the user returned to the main menu
   b. the program should check if the type of aircraft has been entered. If not then a suitable error message should be displayed and the user returned to the main menu
   c. the program should check that the number of first-class seats has been entered. If not then a suitable error message should be displayed and the user returned to the main menu
   d. the program should check that the maximum flight range for the type of aircraft is greater than or equal to the distance between the airports. If not then a suitable error message should be displayed and the user returned to the main menu
   e. the user should be asked to enter the price of a standard-class seat and the price of a first-class seat
   f. the program should then calculate the flight cost per seat, flight cost, flight income and flight profit using the formulae shown in **Figure 3**
   g. the results of these calculations should be displayed and the user returned to the main menu.

7. If the user selects the 'Clear data' option:
   - the program should clear all data that has been entered by the user and then return the user to the main menu.

END OF PROGRAMMING PROJECT TASK

Task completed:

DATE STARTED	DATE COMPLETED	TOTAL TIME TAKEN
8/11/20	11/12/20	11 hours

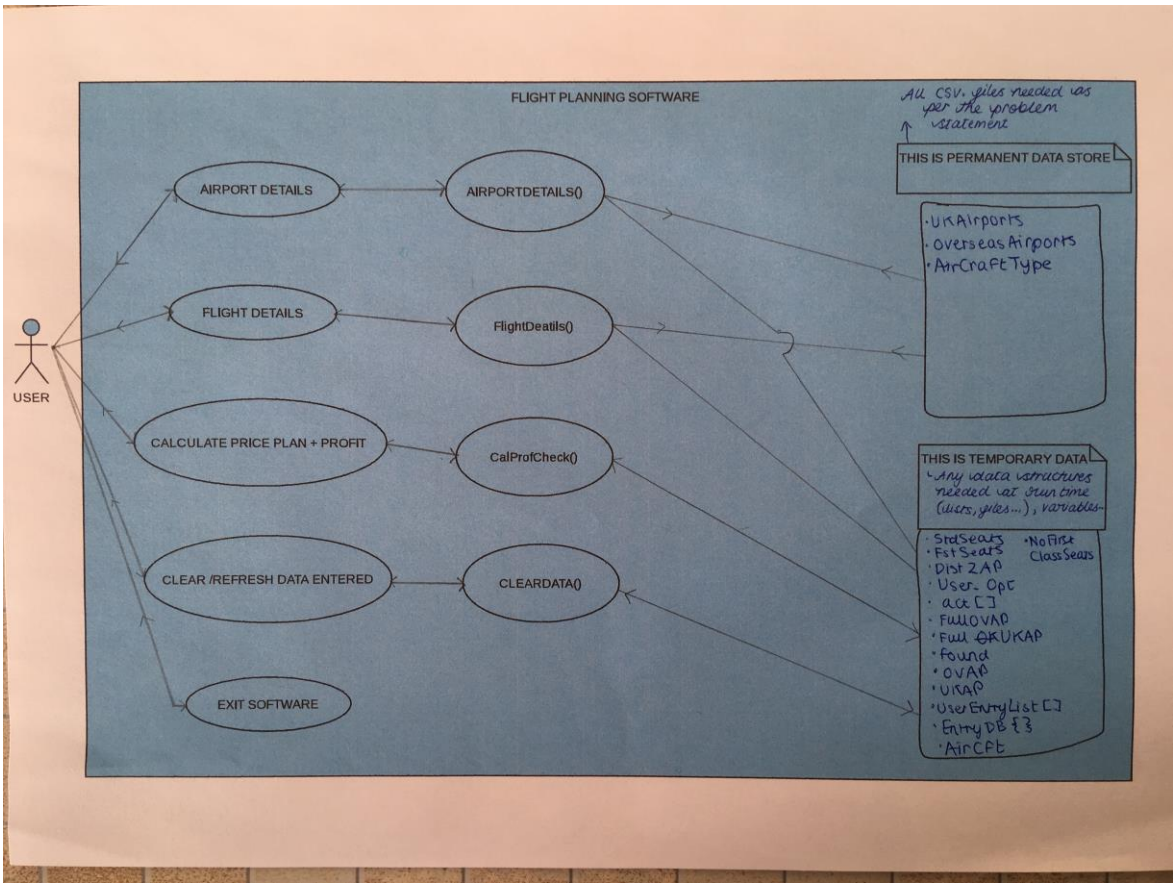
Analysis

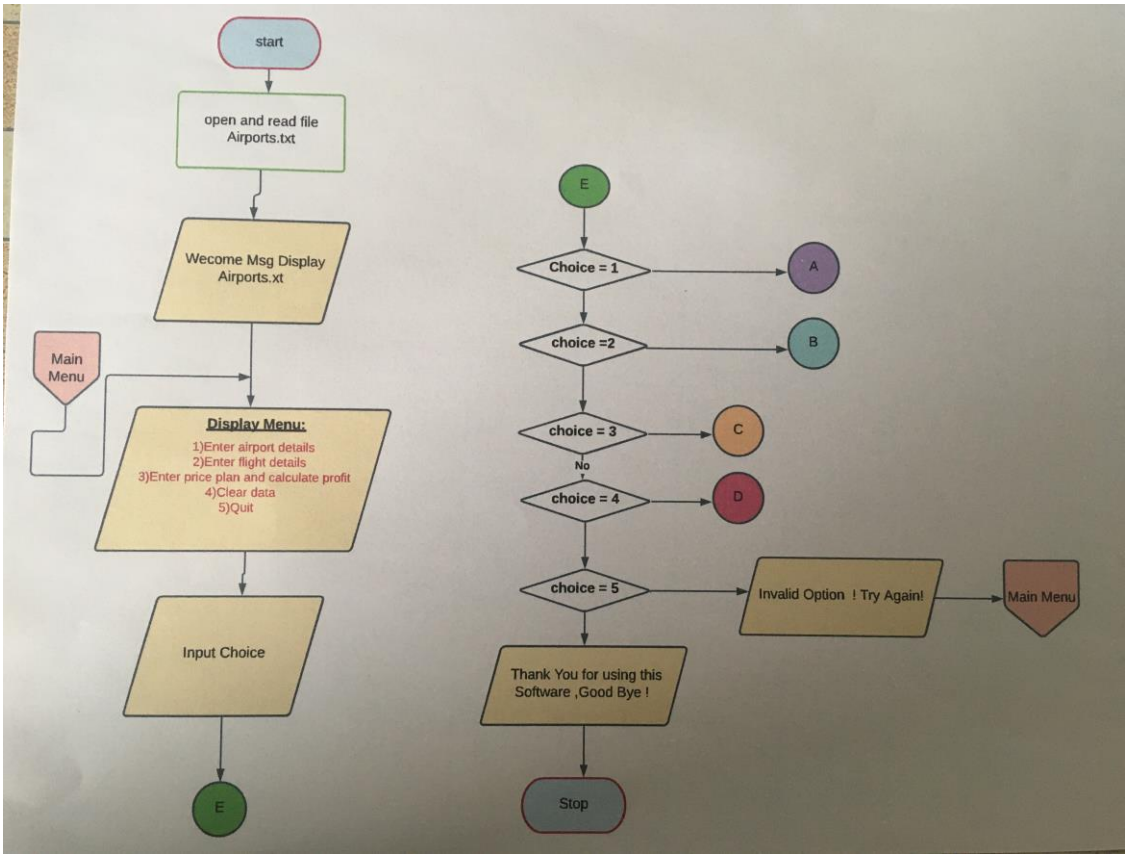
A new airline wants to start running commercial passenger flights. In order to assess the feasibility of proposed flights they want a program that can help them calculate the likely profitability of running a flight between a UK airport and an overseas airport. The UK airport will be either Liverpool John Lennon or Bournemouth International.

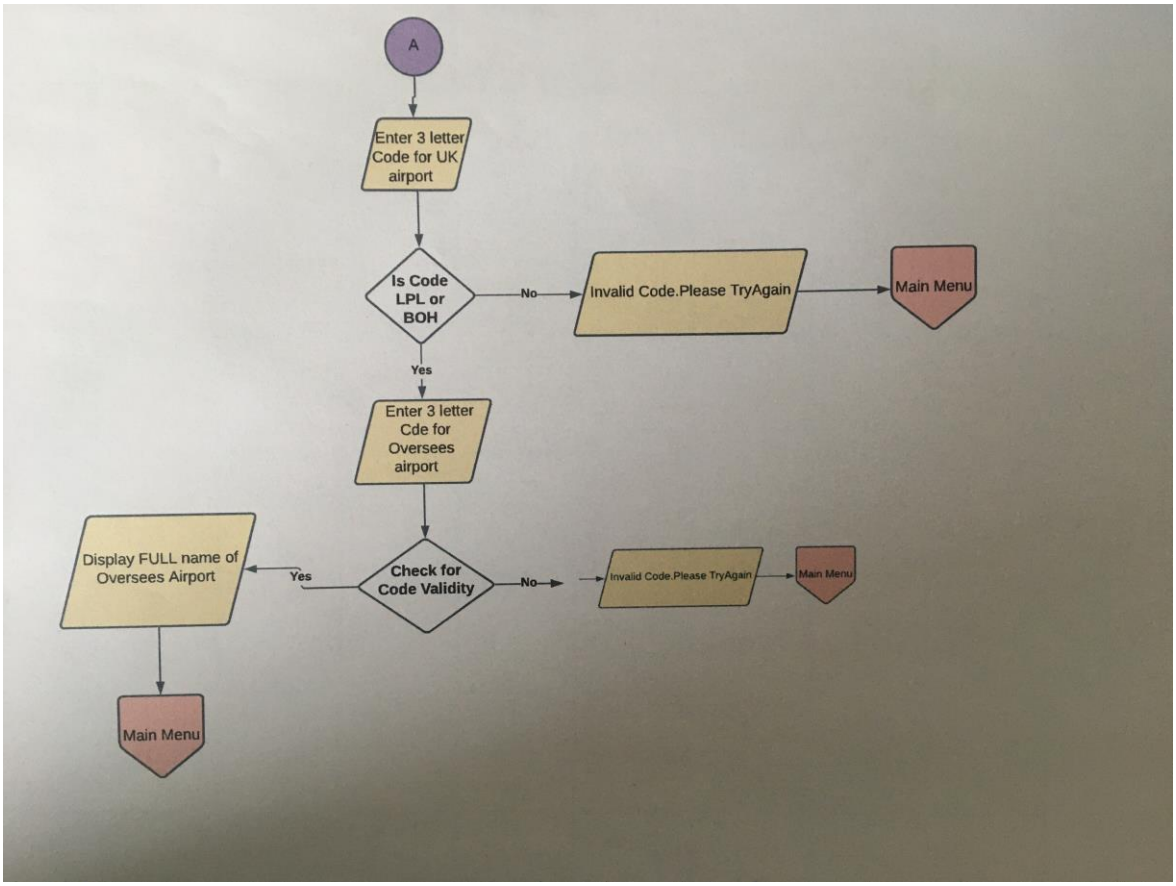
Success Criteria:

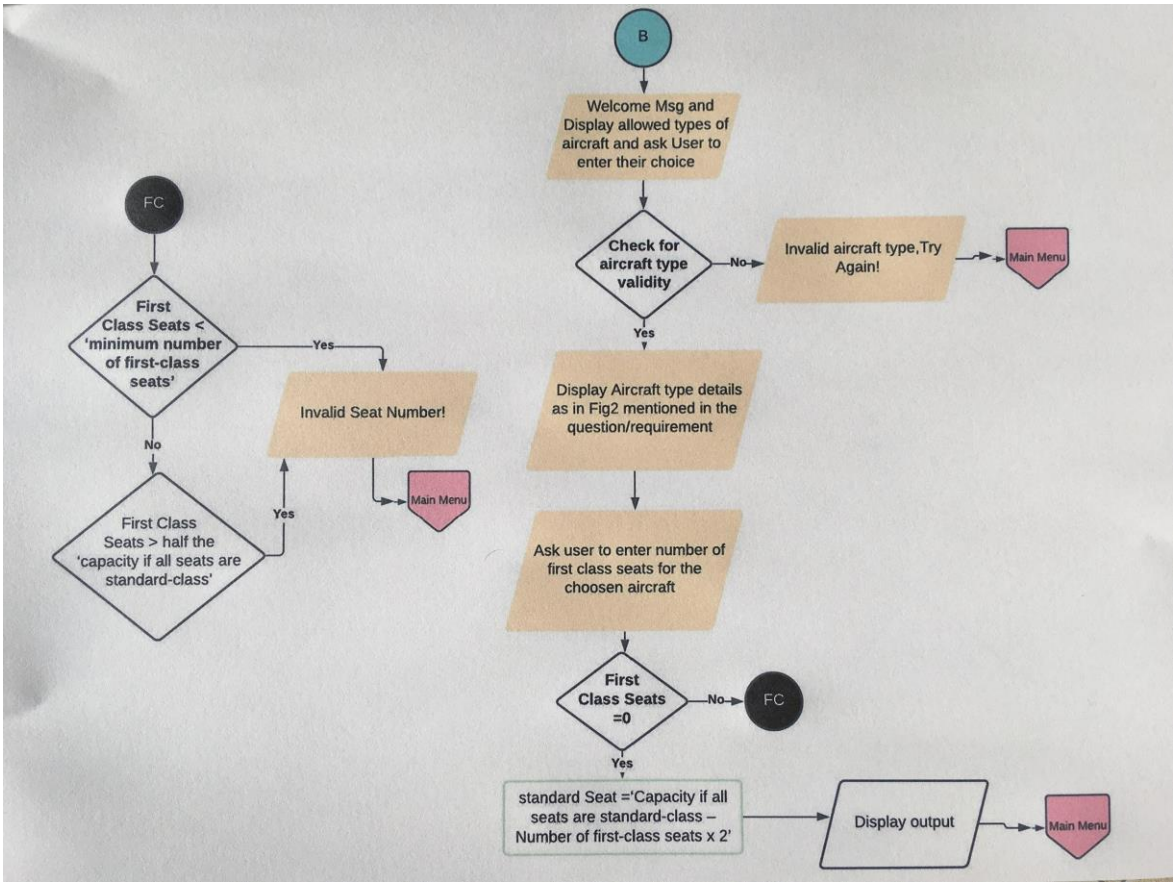
1. Be user friendly
2. Help the user calculate the profitability of their chosen flight path
3. Should calculate and display the flight cost, flight income, flight cost per seat and the total flight profit.
4. Possible error handling

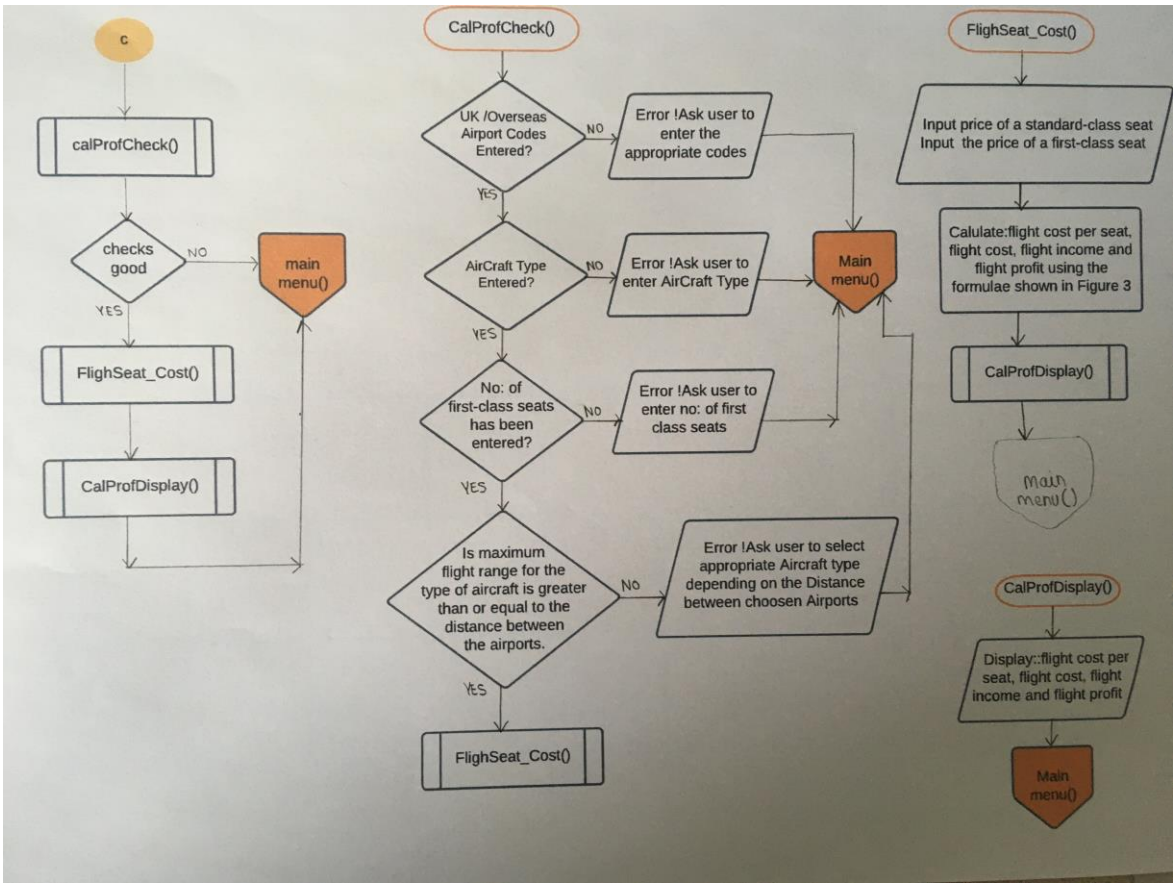
Design – Flow chart / Use Case Diagram

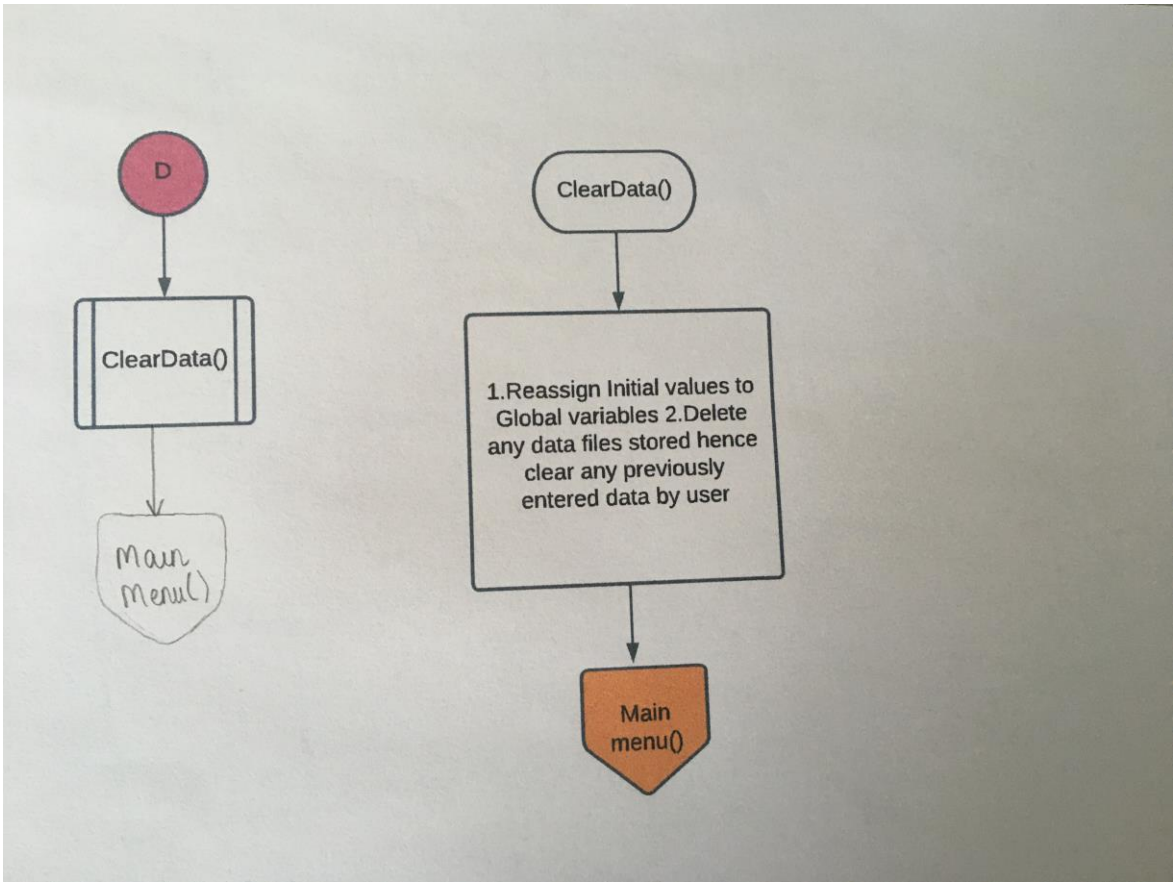



start
open and read file Airports.txt
Wecome Msg Display Airports.xt
Main Menu
Display Menu:
1)Enter airport details
2)Enter flight details
3)Enter price plan and calculate profit
4)Clear data
5)Quit
Input Choice
E
E
Choice = 1
A
choice =2
B
choice = 3
C
No
choice = 4
D
choice = 5
Invalid Option ! Try Again!
Main Menu
Thank You for using this Software ,Good Bye !
Stop

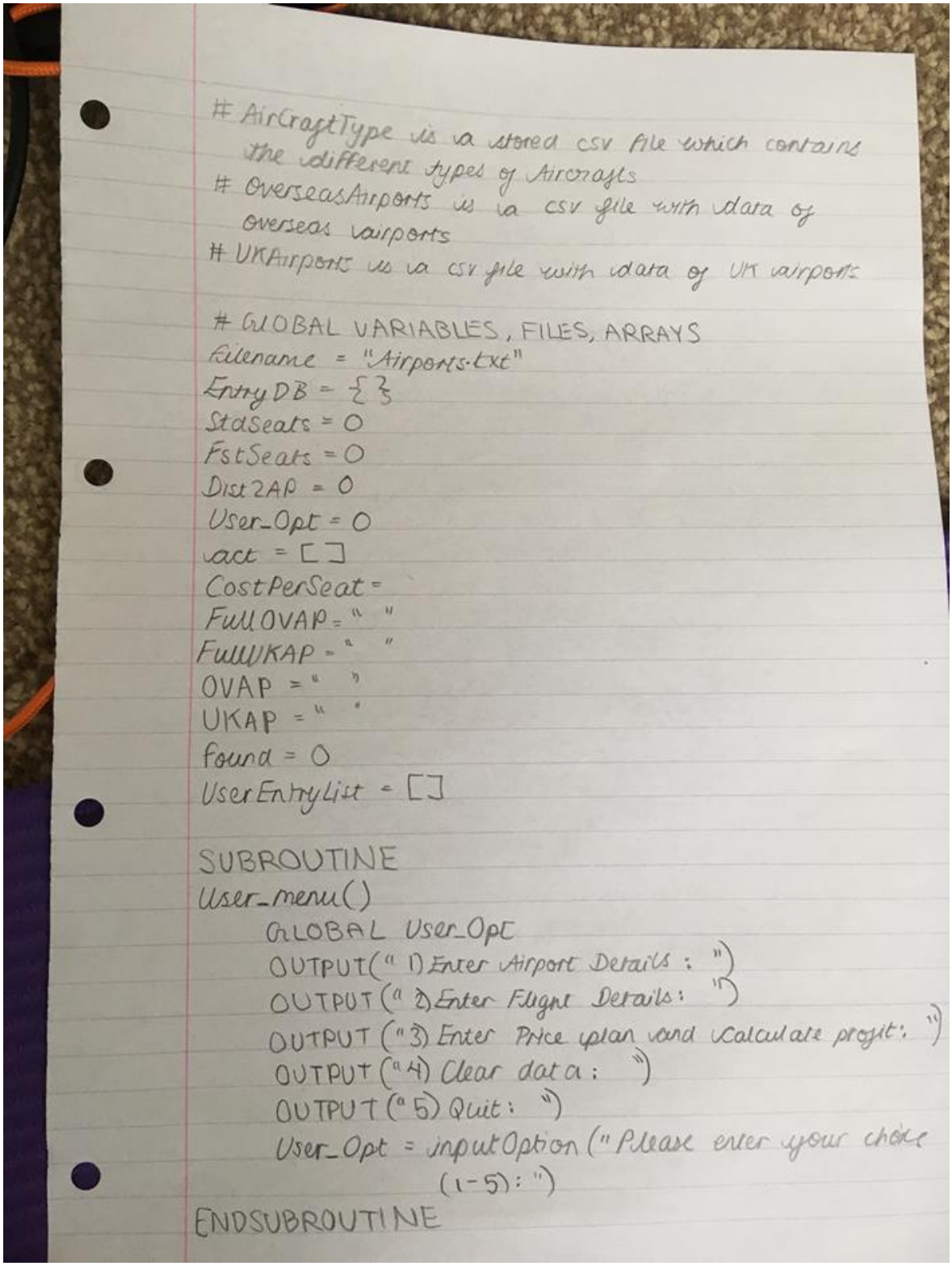

A
Enter 3 letter Code for UK airport
Is Code LPL or BOH
No
Invalid Code.Please TryAgain
Main Menu
Yes
Enter 3 letter Cde for Oversees airport
Display FULL name of Oversees Airport
Yes
Check for Code Validity
No
Invalid Code.Please TryAgain
Main Menu
Main Menu


B
Welcome Msg and Display allowed types of aircraft and ask User to enter their choice
Check for aircraft type validity
No
Invalid aircraft type,Try Again!
Main Menu
Yes
Display Aircraft type details as in Fig2 mentioned in the question/requirement
Ask user to enter number of first class seats for the choosen aircraft
First Class Seats =0
No
FC
Yes
standard Seat ='Capacity if all seats are standard-class – Number of first-class seats x 2'
Display output
Main Menu
FC
First Class Seats < 'minimum number of first-class seats'
Yes
Invalid Seat Number!
Main Menu
No
First Class Seats > half the 'capacity if all seats are standard-class'
Yes

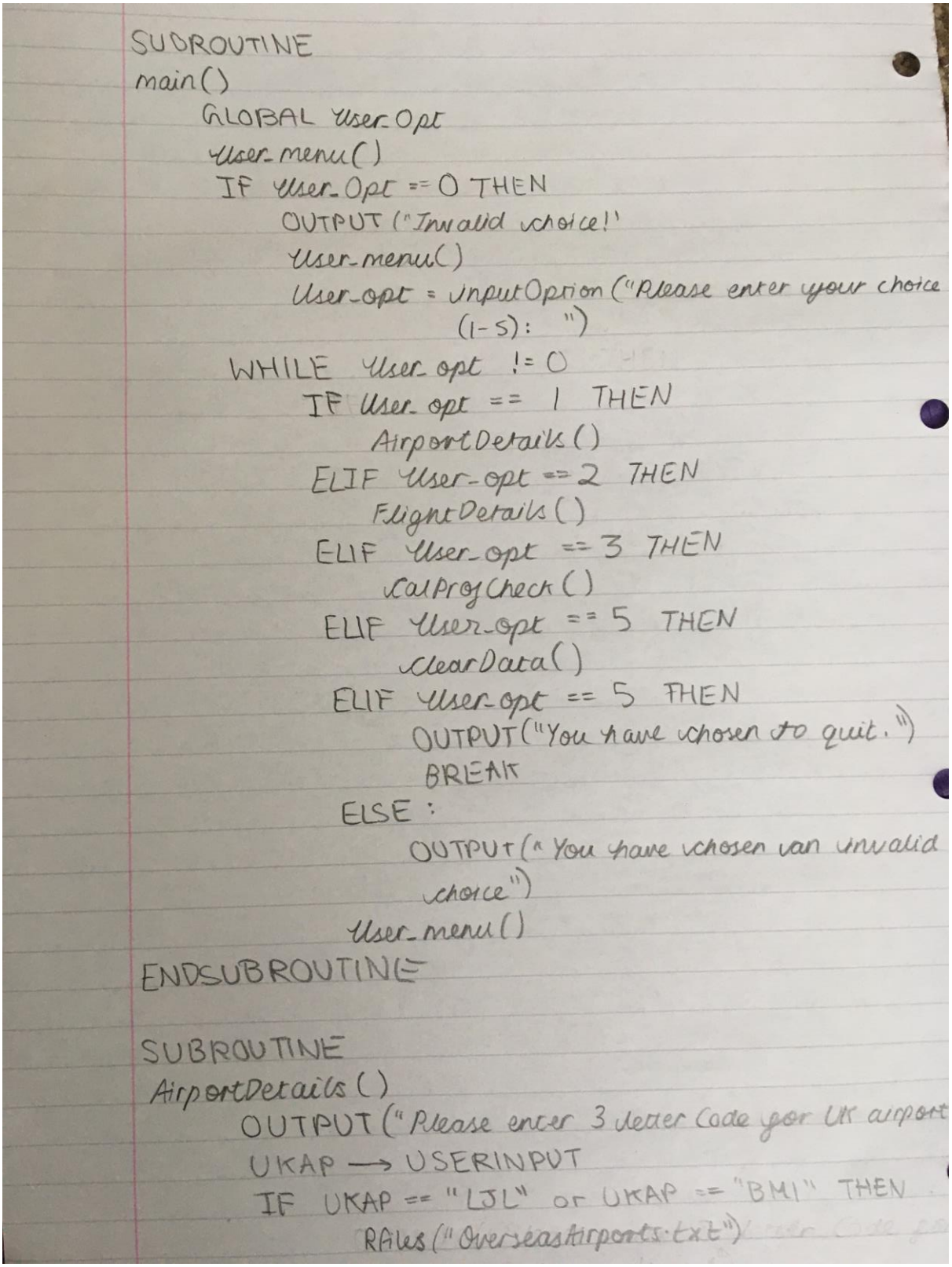

C
calProfCheck()
checks good
NO
main menu()
YES
FlighSeat_Cost()
CalProfDisplay()
CalProfCheck()
UK /Overseas Airport Codes Entered?
NO
Error !Ask user to enter the appropriate codes
YES
AirCraft Type Entered?
NO
Error !Ask user to enter AirCraft Type
Main menu()
YES
No: of first-class seats has been entered?
No
Error !Ask user to enter no: of first class seats
YES
Is maximum flight range for the type of aircraft is greater than or equal to the distance between the airports.
No
Error !Ask user to select appropriate Aircraft type depending on the Distance between choosen Airports
YES
FlighSeat_Cost()
FlighSeat_Cost()
Input price of a standard-class seat Input the price of a first-class seat
Calulate:flight cost per seat, flight cost, flight income and flight profit using the formulae shown in Figure 3
CalProfDisplay()
Main menu()
CalProfDisplay()
Display::flight cost per seat, flight cost, flight income and flight profit
Main menu()


D
ClearData()
Main Menu()
ClearData()
1.Reassign Initial values to Global variables 2.Delete any data files stored hence clear any previously entered data by user
Main menu()

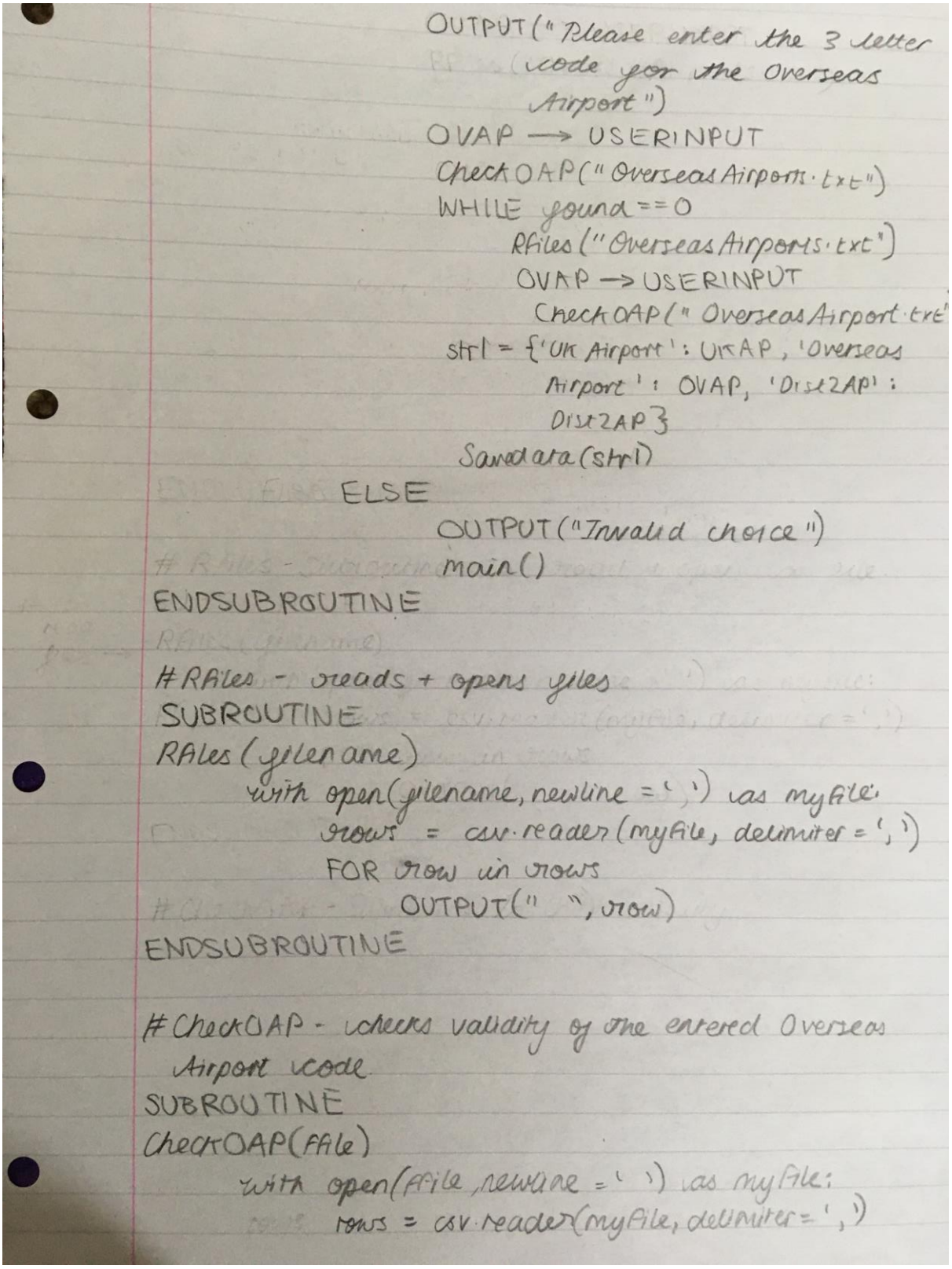
Design – Algorithm



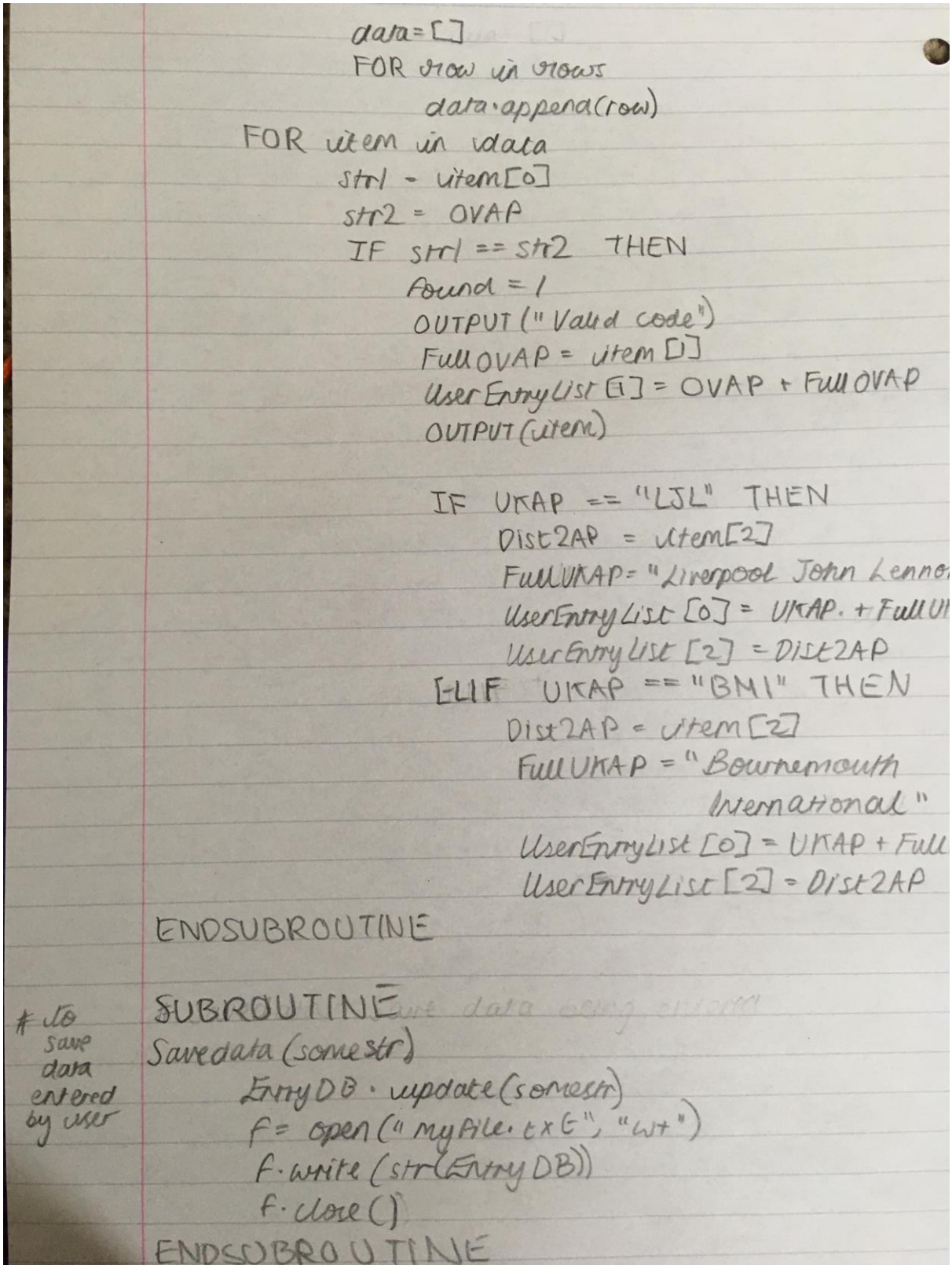
```
# AirCraftType is a stored csv file which contains
  the different types of Aircrafts
# OverseasAirports is a csv file with data of
  overseas airports
# UKAirports is a csv file with data of UK airports

# GLOBAL VARIABLES, FILES, ARRAYS
Filename = "Airports.txt"
EntryDB = {}
StdSeats = 0
FstSeats = 0
Dist2AP = 0
User_Opt = 0
act = []
CostPerSeat =
FullOVAP = " "
FullUKAP = " "
OVAP = " "
UKAP = " "
Found = 0
UserEntryList = []

SUBROUTINE
User_menu()
    GLOBAL User_Opt
    OUTPUT(" 1) Enter Airport Details : ")
    OUTPUT(" 2) Enter Flight Details: ")
    OUTPUT ("3) Enter Price plan and Calculate profit: ")
    OUTPUT ("4) Clear data : ")
    OUTPUT ("5) Quit: ")
    User_Opt = inputOption ("Please enter your choice
                (1-5): ")
ENDSUBROUTINE
```



```
SUBROUTINE
main()
    GLOBAL User_Opt
    User_menu()
    IF User_Opt == 0 THEN
        OUTPUT("Invalid choice!"
        User_menu()
        User_opt = InputOption("Please enter your choice
                   (1-5): ")
    WHILE User_opt != 0
        IF User_opt == 1 THEN
            AirportDetails()
        ELIF User_opt == 2 THEN
            FlightDetails()
        ELIF User_opt == 3 THEN
            CalProgCheck()
        ELIF User_opt == 5 THEN
            clearData()
        ELIF User_opt == 5 THEN
            OUTPUT("You have chosen to quit.")
            BREAK
        ELSE:
            OUTPUT("You have chosen an invalid
            choice")
        User_menu()
ENDSUBROUTINE

SUBROUTINE
AirportDetails()
    OUTPUT("Please enter 3 letter Code for UK airport
    UKAP ⟶ USERINPUT
    IF UKAP == "LJL" or UKAP == "BMI" THEN
        RAise("OverseasAirports.txt")
```

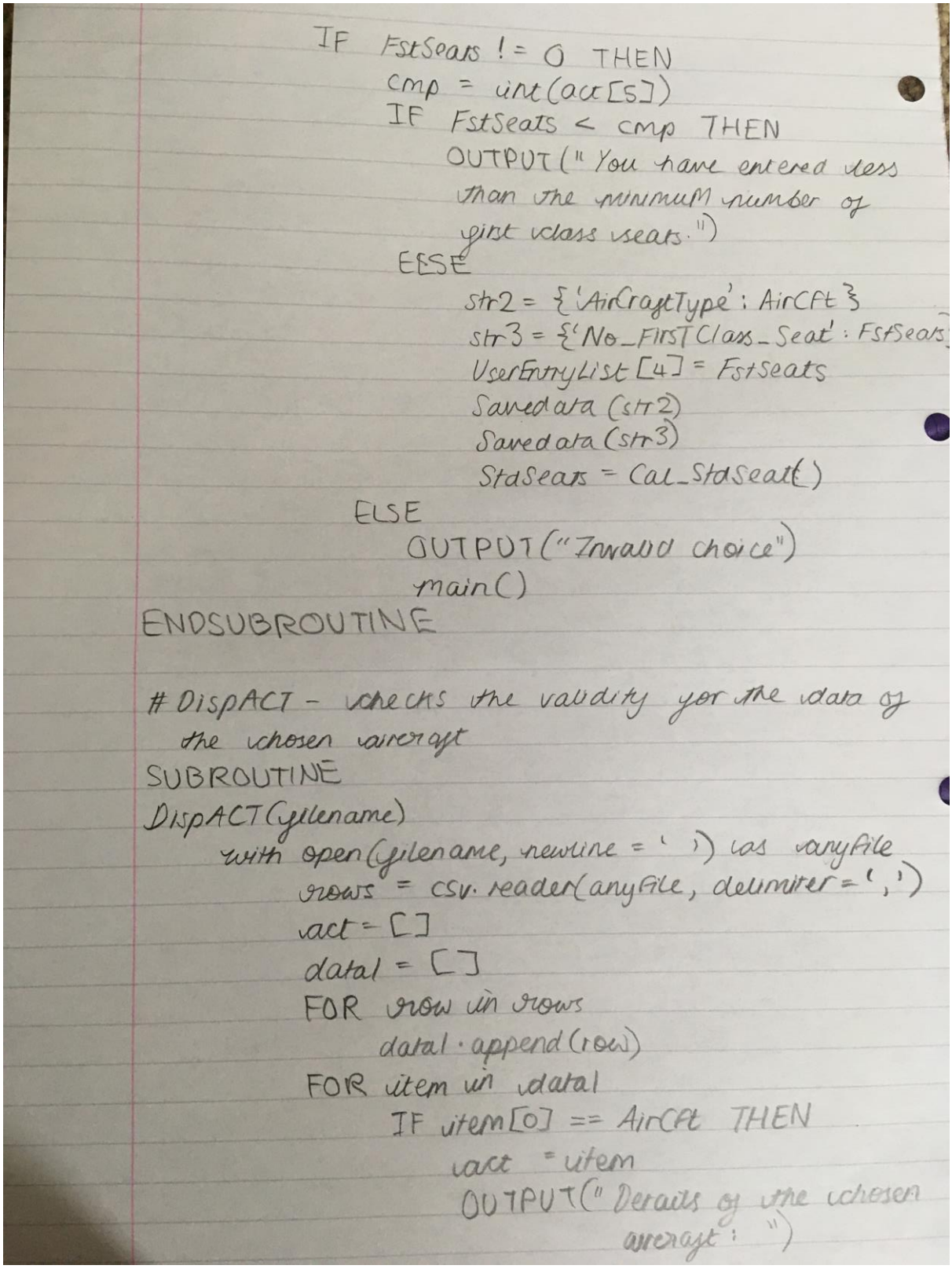


```
                OUTPUT("Please enter the 3 letter
                    (code for the Overseas
                    Airport")
                OVAP ⟶ USERINPUT
                CheckOAP("OverseasAirports.txt")
                WHILE found == 0
                    RFiles("OverseasAirports.txt")
                    OVAP ⟶ USERINPUT
                    CheckOAP("OverseasAirport.txt"
                str1 = {'UK Airport': UKAP, 'Overseas
                    Airport': OVAP, 'Dist2AP':
                    Dist2AP}
                SaveData(str1)
            ELSE
                OUTPUT("Invalid choice")
                main()
ENDSUBROUTINE

#RAles - reads + opens files
SUBROUTINE
RAles(filename)
    with open(filename, newline = '') as myfile:
        rows = csv.reader(myfile, delimiter = ',')
        FOR row in rows
            OUTPUT(" ", row)
ENDSUBROUTINE

#CheckOAP - checks validity of the entered Overseas
  Airport code
SUBROUTINE
CheckOAP(File)
    with open(File, newline = '') as myfile:
        rows = csv.reader(myfile, delimiter = ',')
```

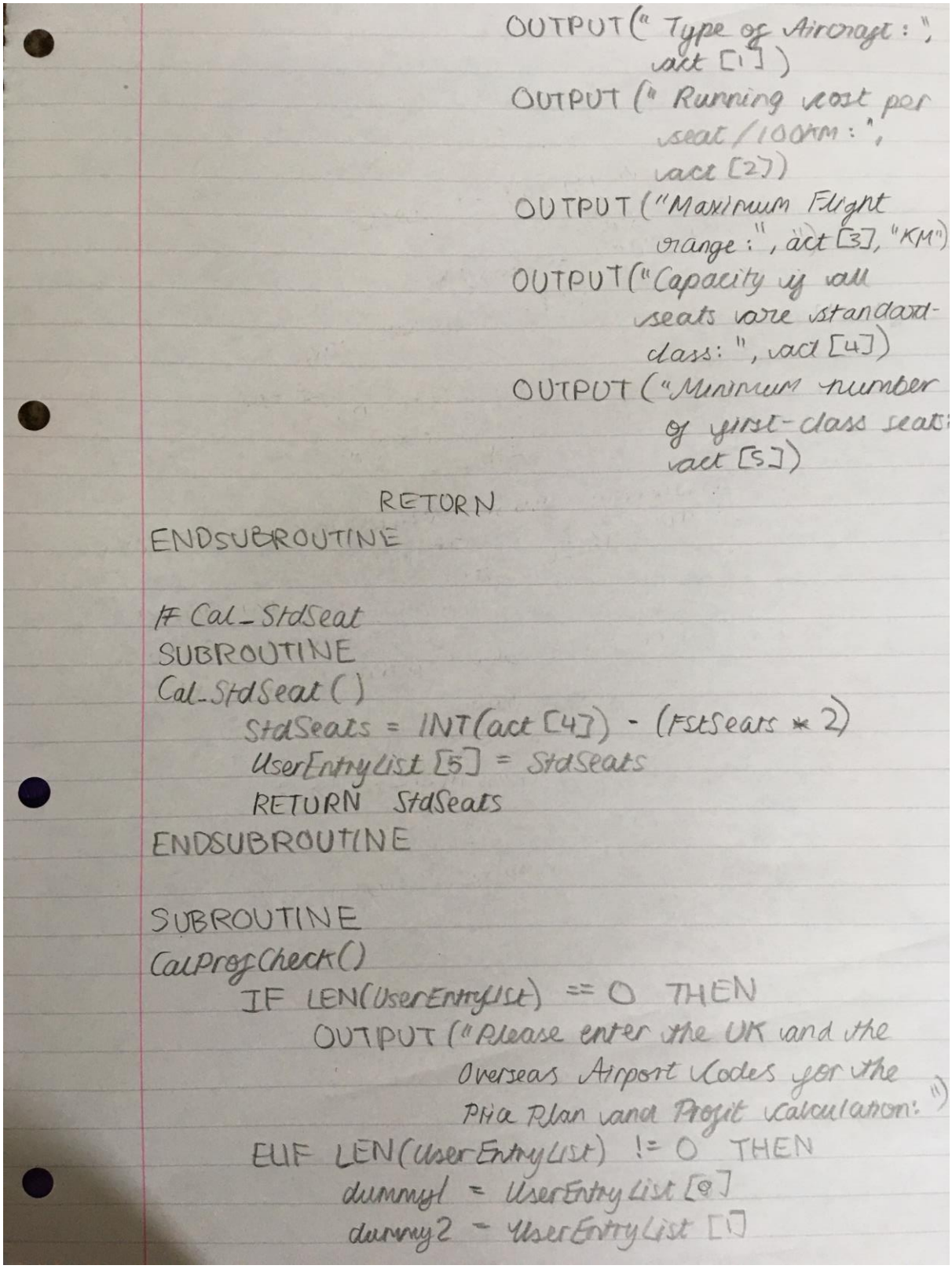


```
        data=[]
        FOR row in rows
            data.append(row)
    FOR item in data
        str1 = item[0]
        str2 = OVAP
        IF str1 == str2 THEN
            Found = 1
            OUTPUT("Valid code")
            FullOVAP = item[1]
            UserEntryList[1] = OVAP + FullOVAP
            OUTPUT(item)

            IF UKAP == "LJL" THEN
                Dist2AP = item[2]
                FullUKAP = "Liverpool John Lenno
                UserEntryList[0] = UKAP. + FullU
                UserEntryList[2] = Dist2AP
            ELIF UKAP == "BMI" THEN
                Dist2AP = item[2]
                FullUKAP = "Bournemouth
                           International"
                UserEntryList[0] = UKAP + Full
                UserEntryList[2] = Dist2AP
ENDSUBROUTINE

# to save data entered by user
SUBROUTINE
Savedata(somestr)
    EntryDB.update(somestr)
    f = open("myfile.txt", "wt")
    f.write(str(EntryDB))
    f.close()
ENDSUBROUTINE
```

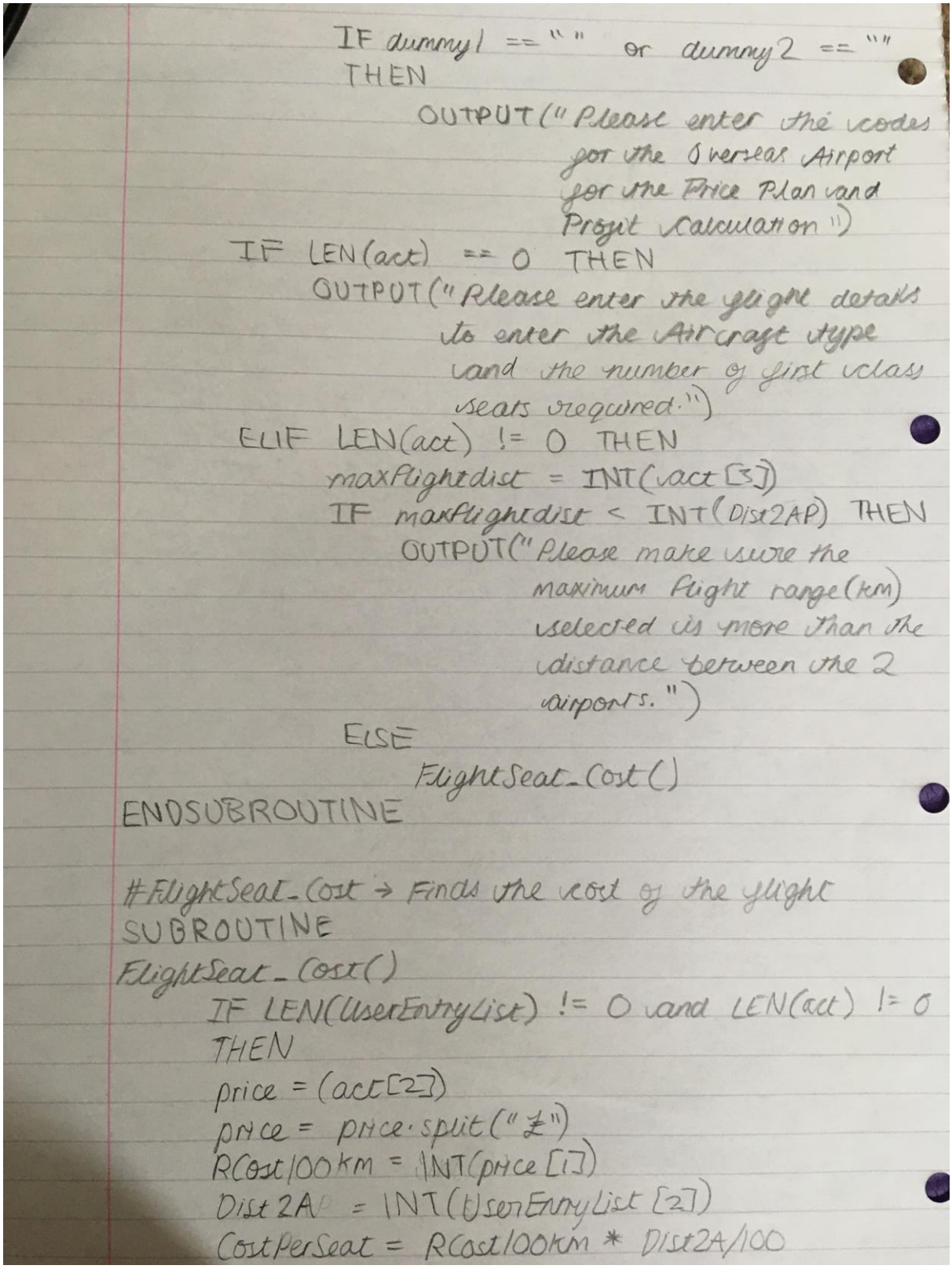
```
# Validates the user's choice in the main menu
SUBROUTINE
inputOption(message)
    WHILE TRUE:
        TRY
            Opt_Value = INT(USERINPUT(message))
        EXCEPT ValueError:
            OUTPUT("Not an integer!")
            CONTINUE
        ELSE
            RETURN Opt_Value
            BREAK
ENDSUBROUTINE

SUBROUTINE
FlightDetails()
    RFiles("AirCraftType.txt")
    ACtype = ["MN","LN","MW"]
    OUTPUT("Enter the type of aircraft. Please refer
        below.")
    OUTPUT("MN - Medium Narrow Body")
    OUTPUT("LN - Large Narrow Body")
    OUTPUT("MW - Medium Wide Body")
    OUTPUT(ACtype)
    AirCft → USERINPUT
    IF AirCft in ACtype THEN
        OUTPUT("Chosen Aircraft is:", AirCft)
    ELSE
        OUTPUT("Invalid choice.")
    DispACT("AirCraftType.txt")

    FstSeats = inputOption("Please enter the
        number of first class seats: ")
```



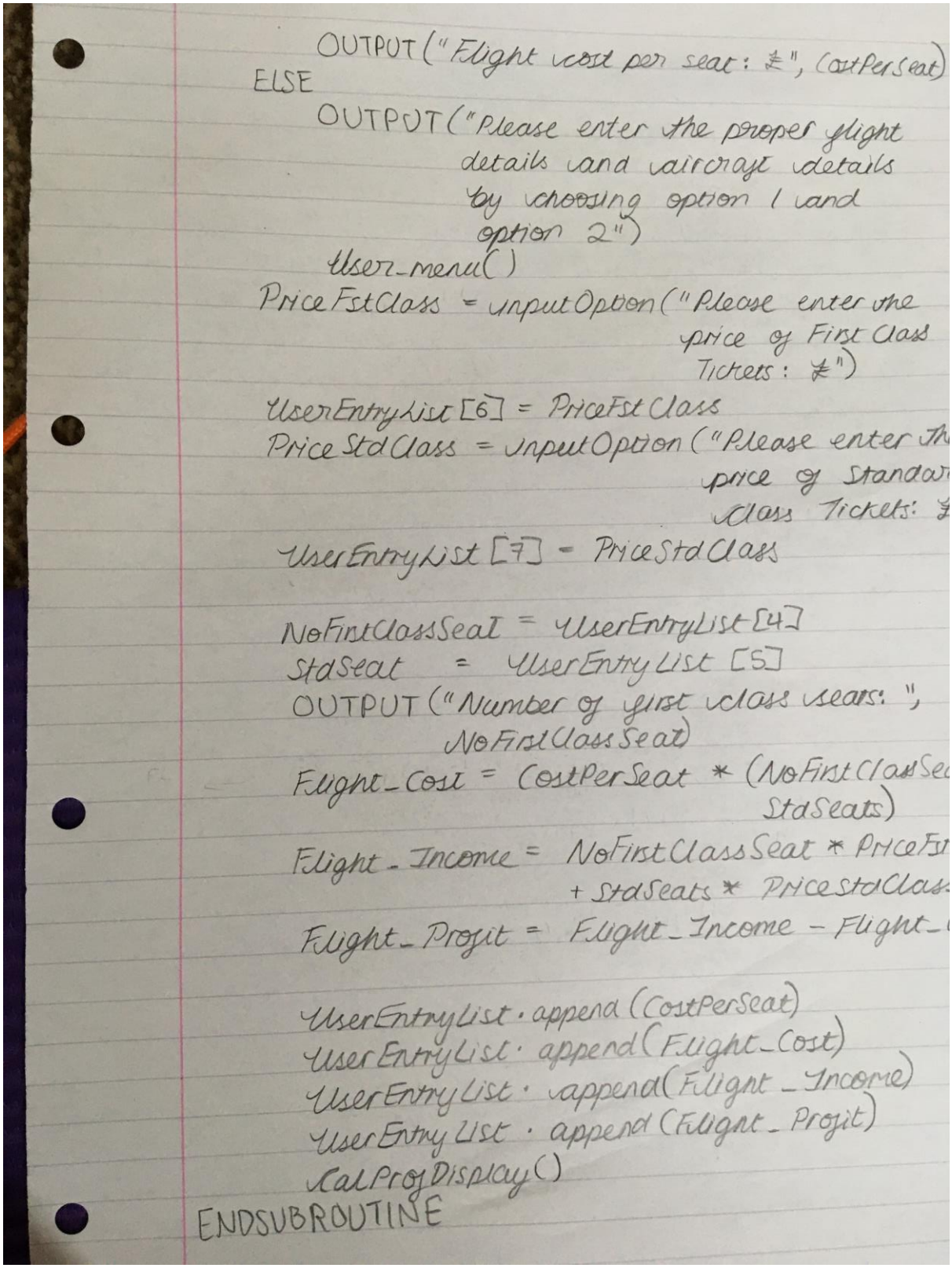
```
        IF FstSeats != 0 THEN
            cmp = int(act[5])
            IF FstSeats < cmp THEN
                OUTPUT("You have entered less
                than the minimum number of
                first class seats.")
            EESE
                str2 = {'AirCraftType': AirCft}
                str3 = {'No_First Class_Seat': FstSeats}
                UserEntryList[4] = FstSeats
                Savedata(str2)
                Savedata(str3)
                StdSeats = Cal_StdSeat()
        ELSE
            OUTPUT("Invalid choice")
            main()
ENDSUBROUTINE

# DispACT - checks the validity for the data of
  the chosen aircraft
SUBROUTINE
DispACT(filename)
    with open(filename, newline = '') as anyFile
        rows = csv.reader(anyFile, delimiter=',')
        act = []
        data1 = []
        FOR row in rows
            data1.append(row)
        FOR item in data1
            IF item[0] == AirCft THEN
                act = item
                OUTPUT("Details of the chosen
                        aircraft: ")
```



```
        OUTPUT("Type of Aircraft: ", act[1])
        OUTPUT("Running cost per seat/100KM: ", act[2])
        OUTPUT("Maximum Flight range: ", act[3], "KM")
        OUTPUT("Capacity if all seats are standard-class: ", act[4])
        OUTPUT("Minimum number of first-class seats: ", act[5])

    RETURN
ENDSUBROUTINE

# Cal_StdSeat
SUBROUTINE
Cal_StdSeat()
    StdSeats = INT(act[4]) - (FstSeats * 2)
    UserEntryList[5] = StdSeats
    RETURN StdSeats
ENDSUBROUTINE

SUBROUTINE
CalProgCheck()
    IF LEN(UserEntryList) == 0 THEN
        OUTPUT("Please enter the UK and the Overseas Airport Codes for the Flight Plan and Profit calculation.")
    ELIF LEN(UserEntryList) != 0 THEN
        dummy1 = UserEntryList[0]
        dummy2 = UserEntryList[1]
```



```
        IF dummy1 == "" or dummy2 == ""
        THEN
            OUTPUT("Please enter the codes
                    for the Oversea Airport
                    for the Price Plan and
                    Profit Calculation")
    IF LEN(act) == 0 THEN
        OUTPUT("Please enter the flight details
                to enter the Aircraft type
                and the number of first class
                seats required.")
    ELIF LEN(act) != 0 THEN
        maxflightdist = INT(act[3])
        IF maxflightdist < INT(Dist2AP) THEN
            OUTPUT("Please make sure the
                    maximum flight range(km)
                    selected is more than the
                    distance between the 2
                    airports.")
        ELSE
            FlightSeat_Cost()
ENDSUBROUTINE

#FlightSeat_Cost → Finds the cost of the flight
SUBROUTINE
FlightSeat_Cost()
    IF LEN(UserEntryList) != 0 and LEN(act) != 0
    THEN
    price = (act[2])
    price = price.split("£")
    RCost/100km = INT(price[1])
    Dist2AP = INT(UserEntryList[2])
    CostPerSeat = RCost/100km * Dist2AP/100
```



```
        OUTPUT("Flight cost per seat: £", CostPerSeat)
ELSE
        OUTPUT("Please enter the proper flight
                details and aircraft details
                by choosing option 1 and
                option 2")
        User_menu()
PriceFstClass = InputOption("Please enter the
                price of First Class
                Tickets: £")
UserEntryList[6] = PriceFstClass
PriceStdClass = InputOption("Please enter th
                price of Standar
                Class Tickets: £
UserEntryList[7] = PriceStdClass

NoFirstClassSeat = UserEntryList[4]
StdSeat = UserEntryList[5]
OUTPUT("Number of first class seats: ",
        NoFirstClassSeat)
Flight_Cost = CostPerSeat * (NoFirstClassSe
                StdSeats)
Flight_Income = NoFirstClassSeat * PriceFi
                + StdSeats * PriceStdClas
Flight_Profit = Flight_Income - Flight_

UserEntryList.append(CostPerSeat)
UserEntryList.append(Flight_Cost)
UserEntryList.append(Flight_Income)
UserEntryList.append(Flight_Profit)
CalProfDisplay()
ENDSUBROUTINE
```

```
# CalProgDisplay - Displays all values
SUBROUTINE
CalProgDisplay()
        OUTPUT(UserEntryList)
        OUTPUT("UK Airport: ", UserEntryList[0])
        OUTPUT("OVERSEAS AIRPORT: ", UserEntryList[1])
        OUTPUT("DISTANCE: ", UserEntryList[2], "KM")
        OUTPUT("FLIGHT COST PER SEAT", UserEntryList[8])
        OUTPUT("FLIGHT COST: ", UserEntryList[9])
        OUTPUT("FLIGHT INCOME: ", UserEntryList[10])
        OUTPUT("FLIGHT PROFIT: ", UserEntryList[11])
ENDSUBROUTINE

SUBROUTINE
ClearData()
    OUTPUT("You have chosen to clear data.")
    AirCraftTypeEntered = 0
    NoFirstClassSeats = 0
    UKAP = ""
    OVAP = ""
    AirCft = ""
    StdSeats = 0
    FstSeats = 0
    UKAPEntered = 0
    OVAPEntered = 0
    UserEntryList[]
    act clear()
    os.remove("myfile.txt")
ENDSUBROUTINE
```

Test Plan

Test	What am I testing?	What data will I use?	Normal?	Boundary?	Erroneous?	Expected Result
1	**Menu functionality** **a) Option 1** - Airport Details **b) Option 2** - Flight Details **c) Option 3** - Profit Calculation **d) Option 4** – Clear Data **e) Option 5** – Exit	I will use the data from the example in **Figure 4**, which is on the Specification sheet for the project.	An integer only should be entered for the main menu. If anything, else is entered a suitable message is displayed and the user is given another chance. ***For the rest of the options:*** -Where required integers should be entered (e.g., price of standard class seat in **Option 3**) -Similarly, required letters or words should be entered by the user (e.g., the aircraft type which asks for the initials of the chosen aircraft in **Option 2**)	When the user is asked to enter a word or initials (e.g., **Option 2** when it asks for the chosen Aircraft Type), I have allowed the use of both uppercase and lowercase letters to prevent an error from displaying in case if the user types either.	I have observed that whilst testing my code, several errors would occur due to typos of variables or other syntax errors which then displayed unrelated error messages which could confuse the user. So, after correcting the errors I decided to include some error handling and wrote a try-block code so that in case if anything goes wrong with the code, a suitable message will be displayed to the user as though there is no error and the program will not be disrupted or stopped.	

My program code:

```
#The Code

#LIBRARIES
import csv
import os
from idlecolors import *

#Welcome header
print("************************************************")
for i in range(0,2):
    print("**                                            **")
print("   WELCOME TO THE FILGHT PLANNING PROGRAM     ")
for i in range(0,2):
    print("**                                            **")
print("************************************************")

#Files, Arrays and Global variables
filename = "Airports.txt"
EntryDB = {} #library for storing the user's data

StdSeats = 0
FstSeats = 0
Dist2AP = 0
User_Opt = 0
act = []
CostPerSeat = 0
FullOVAP = ""
FullUKAP = ""
found = 0
OVAP = ""
UKAP = ""

UserEntryList = [0,0,0,0,0,0,0,0] #This list will be updated and will store the user's information

########


#SUBROUTINE TO OPEN AND READ A FILE
def RFiles(filename): #the value that called this subroutine is put in the parameter 'filename'
    with open(filename, newline='') as myfile: #opens and reads the file + stores it with the filename 'myfile'
        rows = csv.reader(myfile, delimiter=',') #passes my csv file into the 'csv reader method' which by default/already built in
                                                 #the reader method already knows the format of a csv file(seperated by commas...etc)
                                                 #the data is then stored in a 2D array called rows
        for row in rows: #prints out each array in the 2D array
            print("  ",row) #each array will be printed after a space

########

#SUBROUTINE TO CHECK DATA OF AIRCRAFT ENTERED
def DispACT(filename): #Displays the aircraft
    with open(filename, newline='') as anyfile:
        rows = csv.reader(anyfile, delimiter=',') #creates 2D array called 'rows'
        global act
        act = [] #creates an array called act
        datal = [] #array called datal
        for row in rows: #for each array in 'rows'
            datal.append(row) #it is added in the 2D array called datal
        global UserEntryList
        for item in datal: #checks if the first value in every array in 'datal' is equal to the initials(uppercase) of the Aircraft
            if item[0] == AirCft.upper():
                act = item #stores the array containing only the data about the users chosen aircraft in the variable, 'act'
                #print("Details of the aircraft type entered:",act) #displays details of the entered aircraft
                printc(purple("Details of the aircraft chosen:\n"))
                print("     TYPE OF AIRCRAFT:",act[1])
                print("     RUNNING COST PER SEAT / 100KM:",act[2])
                print("     MAXIMUM FLIGHT RANGE:",act[3],"KM")
                print("     CAPACITY IF ALL SEATS ARE STANDARD-CLASS:",act[4])
                print("     MINIMUM NUMBER OF FIRST-CLASS SEATS:",act[5])
        return

########
```

```
#SUBROUTINE TO CHECK THE VALIDITY OF THE OVERSEAS AIRPORT CODE
def CheckOAP(ffile): #Checking the Overseas Airport
    with open(ffile, newline='') as myfile: #reads, opens the OAP file and renames it 'ffile'
        rows = csv.reader(myfile, delimiter=',')
        data = []
        for row in rows:
            data.append(row) #appends each row from the 2D array 'rows' to the 2D array 'data'
    global found
    global Dist2AP
    global FullOVAP
    global FullUKAP
    global UserEntryList
    global act
    global OVAP
    global UKAP

    for item in data: #for each array stored in the 2D array 'data'
        str1 = item[0] #the variable str1 stores the 1st value of that array which is the initials of the overseas airport(which is in uppercase)
        str2 = OVAP.upper() #str2 stores the uppercase(the same as str1)

        if str1==str2: #if they are both equal then the following code runs
            found=1 #dummy variable
            print(" ")
            print("Valid code")
            print("The Overseas Airport chosen is:",item[1]) #prints the full name of the Overseas Airport

            FullOVAP=item[1] #stores the full name of the Overseas Airport in the variable FullOVAP
            UserEntryList[1] = OVAP.upper() + FullOVAP #the initials and full name of the OVAP is stored as the second value UserEntryList array
            print(item)


              #finds the distance between both the overseas and UK airport + stores it in the UserEntryList
              if UKAP.upper() == "LJL":
                  Dist2AP = item[2] #finds the Distance between 2 airports
                  FullUKAP = "Liverpool John Lennon" #stores the full name of the UK airport as FullUKAP
                  UserEntryList[0] = UKAP.upper() + FullUKAP #adds both the names to the list
                  UserEntryList[2] = Dist2AP #adds the distance to the list
                  #print("SS",Dist2AP)
              elif UKAP.upper() == "BMI": #same as above but for BMI
                  Dist2AP = item[2]
                  FullUKAP = "Bournemouth International"
                  UserEntryList[0] = UKAP.upper() + FullUKAP
                  UserEntryList[2] = Dist2AP
              #print("SS",Dist2AP)

#######

#SUBROUTINE TO CALCULATE THE AMOUNT OF STANDARD SEATS
def Cal_StdSeat(): #Calculating number of Standard Class Seats
    #print("inside cal_std",act[4])
    global StdSeats
    global UserEntryList
    global FstSeats
    StdSeats = int(act[4]) - (FstSeats*2) #capacity if all seats are standard class - number of first class seats x 2
    UserEntryList[5] = StdSeats #stores the amount of standard class seats in the UserEntryList
    #print("inside cal_std2,",StdSeats)
    #print(UserEntryList)
    return StdSeats

#######

#SUBROUTINE TO SAVE THE OUTPUT AS THE DETAILS ARE BEING ENTERED EVERY QUESTION
def Savedata(somestr):
    EntryDB.update(somestr) #updates the values entered into the library EntryDB
    #print(EntryDB)
    f = open("myfile.txt","w+") #writes the values to the library
    f.write(str(EntryDB)) #converts all data to be strings
    f.close()

#######
```

```
#SUBROUTINE TO PRINT AN ERROR MESSAGE
def Error_Message():
    print("INVALID AIRPORT CODE!")

#######

def inputOption(message):
    while True:
        try:
            Opt_Value = int(input(message))
        except ValueError:
            print("Not an integer! Try again.")
            continue
        else:
            return Opt_Value
            break

#######

#4 - Choice 1: Airport Details
def AirportDetails():
    print(" ")
    printc(green("You have chosen to enter Airport Details.\n"))

    #UK Airport
    RFiles("UKAirports.txt") #calls the subroutine RFiles to open, read and display the csv file: Airports.txt
    global UKAP
    global OVAP
    global Dist2AP
    global found
    global FullUKAP
    global FullOVAP

    try:
        print(" ")
        UKAP = input("Please enter the three letter UK Airport Code (Refer Above): ")
    except:
        print("Please refer above for the 3 letter UK Airport Code: ") #if any error happens with the above code then this line will be printed



    #Is the UK Airport code valid?
    if UKAP.upper() == "LJL" or UKAP.upper() == "BMI": #accepts both uppercase and lowercase for answers
        val = "FALSE"
    else:
        val = "TRUE"
        while val == "TRUE":
            print("Invalid UK Airport Code. Please re-enter the code by choosing option 1-Airport details")
            main()
            if UKAP.upper() == "LJB" or UKAP.upper() == "BMI":
                val = "FALSE"

    print("You have chosen:",UKAP.upper()," as your UK Airport.")


    #Asks the User for the Overseas Airport Code
    print(" ")
    RFiles("OverseasAirports.txt") #again reads and opens the file for the overseas airport

    try: #error handling
        print(" ")
        OVAP = input("Please enter the three-letter Airport Code for the Overseas Airport (Refer Above): ") #this is what should be displayed to the user
    except: #in case if user input causes an error, then the following code should run
        print("Please refer above for the 3 letter Overseas Airport Code: ") #this line is displayed instead in case of an error


    #Is the Overseas Airport code valid?
    CheckOAP("OverseasAirports.txt") #sends it to check the Overseas Airport code
    while found == 0:
        #print(OVAP.upper(),"inside while")
        RFiles("OverseasAirports.txt")
        OVAP = input("Invalid Overseas Airport Code. Please refer above for the three letter codes: ")
        CheckOAP("OverseasAirports.txt")

    #prints out the entered information to the user
    print("\nYou have chosen:",OVAP.upper(),"-", FullOVAP, "as your Overseas Airport")
    print("You have chosen:",UKAP.upper(),"-", FullUKAP, "as your UK Airport")
    print("The distance between the two is",Dist2AP,"KM")
```

```
    strl={'UK Airport':UKAP.upper(),'Overseas Airport':OVAP.upper(),'Dist2AP':Dist2AP} #stores it in the library strl
    Savedata(strl) #sends strl to the subroutine to save the data
    print(" ")

#########

#5 - Choice 2: Flight Details
def FlightDetails():
    print(" ")
    printc(purple("You have chosen to enter Flight Details.\n"))
    RFiles("AirCraftType.txt") #reads, opens and displays the file which stores info on Aircraft types for the user
    global FstSeats #first class seats
    global AirCft #aircraft
    global StdSeats #standard class seats


    FstSeats = 0
    AType = ["MN", "LN", "MW"] #array stores initials of the types of aircraft

    #error handling
    try:
        print(" ")
        print("Enter the type of Aircraft (refer below) ")
        AirCft = input("Please type   'MN' for a 'Medium Narrow Body Aircraft'\n            'LN' for a 'Large Narrow Body Aircraft'\n            'MW' for a 'Medium
    except:
        print("Please type Aircraft information and refer above.")

    #Is it valid?
    AType = ["MN", "LN", "MW"]

    if AirCft.upper() in AType:
        print("Chosen Aircraft is",AirCft.upper())
        print(" ")
    else:
        print("Invalid type. Try again!")

    DispACT("AirCraftType.txt") #sends to subroutine to display the aircraft information for the user's choice
```

```
    #Entry for First Class Seats and Validating the data
    print(" ")
    FstSeats = inputOption("Please enter the number of First Class Seats: ")
    if FstSeats != 0:
        #print(act)
        cmp = int(act[5]) #retreiving the number of first class seats from the array 'act' whic contains data specific to the users chosen aircraft

        #code checks whether the user has entered a value which is less than the maximum number of FstSeats
        if FstSeats < cmp:
            print("You have entered less than the minimum number of first class seats for the chosen Aircraft!\n Please select Option 2-Flight details and re-enter.")

        else:
            global UserEntryList
            str2 = {'AirCraftType':AirCft}
            str3 = {'No_FirsTClas_Seat':FstSeats}
            UserEntryList[4] = FstSeats #saves it to the UserEntryList

            #saves the data entered
            Savedata(str2)
            Savedata(str3)
            StdSeats = Cal_StdSeat()
            #print("hiii",StdSeats)

    else:
        print("Invalid choice. Please select Option 2-Flight details and re-enter.")
    #print("NUMBER OF STANDARD CLASS SEATS:",StdSeats)

########

#SUBROUTINE TO CALCULATE THE FLIGHT COST PER SEAT
    #THE RUNNING COST PER SEAT PER 100KM
def FlightSeat_Cost():
    global UserEntryList
    global act
    global CostPerSeat
    global Flight_Cost

    #print(UserEntryList)


      if len(UserEntryList) != 0 and len(act) != 0:
          #print(act)
          #print(UserEntryList)
          #print(act[2])
          #print(UserEntryList[2])
          price = (act[2])
          #print(price)
          price = price.split("£")
          #print(price)
          #print(price[1])
          RCost100km = int(price[1])
          Dist2A = int(UserEntryList[2])

          CostPerSeat = RCost100km * Dist2A/100
          print("Flight cost per seat: £",CostPerSeat)

      else:
          print("Please enter the proper flight details and aircraft details by choosing option 1 and option 2")
          user_menu()

      PriceFstClass = inputOption("Please enter the price of First Class Tickets: £")
      UserEntryList[6] = PriceFstClass

      PriceStdClass = inputOption("Please enter the price of Standard Class Tickets: £")
      UserEntryList[7] = PriceStdClass

      #Flight Cost
      NoFirstClassSeat = int(UserEntryList[4])
      StdSeat = int(UserEntryList[5])
      print("Number of First Class Seats: ",NoFirstClassSeat)
      Flight_Cost = CostPerSeat * (NoFirstClassSeat + StdSeats)
      #print("Flight cost: £",Flight_Cost)

      #Flight Income
      Flight_Income = NoFirstClassSeat * PriceFstClass + StdSeats * PriceStdClass
      #print("Flight Income: £",Flight_Income)
```

```
    #Flight Profit
    Flight_Profit = Flight_Income - Flight_Cost
    #print("Flight Profit: £",Flight_Profit)

    #Appending all the calculated data to the UserEntryList
    UserEntryList.append(CostPerSeat)
    UserEntryList.append(Flight_Cost)
    UserEntryList.append(Flight_Income)
    UserEntryList.append(Flight_Profit)
    #print(UserEntryList)
    CalProfDisplay()

#########

#6 - Option 3
#this code ensures that all values have been entered to proceed with the calculations for the total flight cost
def CalProfCheck():
    global UserEntryList
    global Dist2AP
    global act

    printc(orange("You have chosen to enter the Price Plan and Calculate profit.\n"))
    #RFiles("myfile.txt")
    if len(UserEntryList) == 0:
        print("Pleae enter the UK and the Overseas Airport Codes for the Price Plan and Profit Calculation.")
        print("Please select Option 1 in the main menu to do this.")
    elif len (UserEntryList) != 0:
        dummy1 = UserEntryList[0]
        dummy2 = UserEntryList[1]
        if dummy1 == "" or dummy2 == "":
            print("Please enter the codes for the Overseas Airport for the Price Plan and Profit Calculation")
            print("Please select Option 1 in the main menu to do this.")

    if len(act) == 0:
        print("Please enter the flight details to enter the Aircraft type and the number of first class seats required. This information in needed to calculate the pro
        print("To do this please select Option 2 in the main menu.")
```

```
#2 - User's Menu
def user_menu():
    global User_Opt
    print(" ")
    print("--- ")
    print("MENU")
    print("1: Enter airport details")
    print("2: Enter flight details")
    print("3: Enter price plan and calculate profit")
    print("4: Clear data")
    print("5: Quit")
    print("--- ")
    print(" ")
    User_Opt = inputOption("Please enter your choice (1-5): ")

#3 - Asking the user's choice
def main():
    global User_Opt
    user_menu()
    if User_Opt == 0:
        print("You have chosen an invalid choice! The choice can't be zero. \nTry again!\n ")
        user_menu()
        User_Opt = inputOption("Please enter your choice (1-5): ")

    while User_Opt != 0:

        if User_Opt == 1:
            AirportDetails()

        elif User_Opt == 2:
            FlightDetails()

        elif User_Opt == 3:
            CalProfCheck()

#Q7 - Option 4: Clear data
def ClearData():
    global UKAPEntered
    global OVAPEntered
    global AirCft
    global FstSeats
    global StdSeats
    global AirCraftTypeEntered
    global NoFirstClassSeats
    global UserEntryList

    #ADD ALL VARIABLES AND ARRAYS

    print(" ")
    printc(red("You have chosen to Clear Data."))
    AirCraftTypeEntered = 0
    NoFirstClassSeats = 0
    UKAPEntered = 0
    OVAPEntered = 0
    FstSeats = 0
    StdSeats = 0
    UKAP = ""
    OVAP = ""
    AirCft = ""
    UserEntryList = [0*8]
    try:
        act.clear() #The clear() methof only empties the given list. It doesn't return any values.
        os.remove("myfile.txt") #To delete a file, you must import the OS module, and run its os.remove()
    except:
        print("...")

    print("Cleared all data. You can start over or exit using Option 5 in the main menu.")

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

```
#2 - User's Menu
def user_menu():
    global User_Opt
    print(" ")
    print("--- ")
    print("MENU")
    print("1: Enter airport details")
    print("2: Enter flight details")
    print("3: Enter price plan and calculate profit")
    print("4: Clear data")
    print("5: Quit")
    print("--- ")
    print(" ")
    User_Opt = inputOption("Please enter your choice (1-5): ")

#3 - Asking the user's choice
def main():
    global User_Opt
    user_menu()
    if User_Opt == 0:
        print("You have chosen an invalid choice! The choice can't be zero. \nTry again!\n ")
        user_menu()
        User_Opt = inputOption("Please enter your choice (1-5): ")

    while User_Opt != 0:

        if User_Opt == 1:
            AirportDetails()

        elif User_Opt == 2:
            FlightDetails()

        elif User_Opt == 3:
            CalProfCheck()


          elif User_Opt == 4:
              ClearData()

          elif User_Opt == 5:
              print("You have chosen to quit \nBye!")
              break

          else:
              print("You have chosen an Invalid Option! \nTry again!")

          user_menu()

main()

print("Thank you for using this program!")
```

Program Output:

```
**************************************************
**                                              **
**                                              **
   WELCOME TO THE FILGHT PLANNING PROGRAM
**                                              **
**                                              **
**************************************************

---
MENU
1: Enter airport details
2: Enter flight details
3: Enter price plan and calculate profit
4: Clear data
5: Quit
---

Please enter your choice (1-5): 1

You have chosen to enter Airport Details.

   ['LJL', 'Liverpool John Lennon']
   ['BMI', 'Bournemouth International']

Please enter the three letter UK Airport Code (Refer Above): ljl
You have chosen: LJL  as your UK Airport.

   ['JFK', 'John F Kennedy International', '5326', '5486']
   ['ORY', 'Paris-Only', '629', '379']
   ['MAD', 'Adolfo Suarez Madrid-Barajas', '1428', '1151']
   ['AMS', 'Amserdam Schipol', '526', '489']
   ['CAI', ' Cairo International', '3779', '3584']
```

```
Please enter the three-letter Airport Code for the Overseas Airport (Refer Above): jkk
   ['JFK', 'John F Kennedy International', '5326', '5486']
   ['ORY', 'Paris-Only', '629', '379']
   ['MAD', 'Adolfo Suarez Madrid-Barajas', '1428', '1151']
   ['AMS', 'Amserdam Schipol', '526', '489']
   ['CAI', ' Cairo International', '3779', '3584']
Invalid Overseas Airport Code. Please refer above for the three letter codes: jfk

Valid code
The Overseas Airport chosen is: John F Kennedy International
['JFK', 'John F Kennedy International', '5326', '5486']

You have chosen: JFK - John F Kennedy International as your Overseas Airport
You have chosen: LJL - Liverpool John Lennon as your UK Airport
The distance between the two is 5326 KM


---
MENU
1: Enter airport details
2: Enter flight details
3: Enter price plan and calculate profit
4: Clear data
5: Quit
---

Please enter your choice (1-5): 2

You have chosen to enter Flight Details.

   ['MN', 'Medium Narrow Body', 'Â£8', '2650', '180', '8']
   ['LN', 'Large Narrow Body', 'Â£7', '5600', '220', '10']
   ['MW', 'Medium Wide Body', 'Â£5', '4050', '406', '14']
```

```
Enter the type of Aircraft (refer below)
Please type    'MN' for a 'Medium Narrow Body Aircraft'
               'LN' for a 'Large Narrow Body Aircraft'
               'MW' for a 'Medium Wide Body Aircraft' : ln
Chosen Aircraft is LN

Details of the aircraft chosen
     TYPE OF AIRCRAFT: Large Narrow Body
     RUNNING COST PER SEAT / 100KM: Â£7
     MAXIMUM FLIGHT RANGE: 5600 KM
     CAPACITY IF ALL SEATS ARE STANDARD-CLASS: 220
     MINIMUM NUMBER OF FIRST-CLASS SEATS: 10

Please enter the number of First Class Seats: 8
You have entered less than the minimum number of first class seats for the chosen Aircraft!
 Please select Option 2-Flight details and re-enter.
NUMBER OF STANDARD CLASS SEATS: 0

---
MENU
1: Enter airport details
2: Enter flight details
3: Enter price plan and calculate profit
4: Clear data
5: Quit
---

Please enter your choice (1-5): 2

You have chosen to enter Flight Details.

   ['MN', 'Medium Narrow Body', 'Â£8', '2650', '180', '8']
   ['LN', 'Large Narrow Body', 'Â£7', '5600', '220', '10']
   ['MW', 'Medium Wide Body', 'Â£5', '4050', '406', '14']

Enter the type of Aircraft (refer below)
Please type    'MN' for a 'Medium Narrow Body Aircraft'
               'LN' for a 'Large Narrow Body Aircraft'
               'MW' for a 'Medium Wide Body Aircraft' : ln
Chosen Aircraft is LN
```

```
Details of the aircraft chosen
     TYPE OF AIRCRAFT: Large Narrow Body
     RUNNING COST PER SEAT / 100KM: Â£7
     MAXIMUM FLIGHT RANGE: 5600 KM
     CAPACITY IF ALL SEATS ARE STANDARD-CLASS: 220
     MINIMUM NUMBER OF FIRST-CLASS SEATS: 10

Please enter the number of First Class Seats: 40
NUMBER OF STANDARD CLASS SEATS: 140

---
MENU
1: Enter airport details
2: Enter flight details
3: Enter price plan and calculate profit
4: Clear data
5: Quit
---

Please enter your choice (1-5): 3
You have chosen to enter the Price Plan and Calculate profit.

Flight cost per seat: £ 372.82
Please enter the price of First Class Tickets: £1200
Please enter the price of Standard Class Tickets: £400
Number of First Class Seats:  40

['LJLLiverpool John Lennon', 'JFKJohn F Kennedy International', '5326', 0, 40, 140, 1200, 400, 372.82, 67107.6, 104000, 36892.399999999994]

         UK AIRPORT:  LJLLiverpool John Lennon
         OVERSEAS AIRPORT:  JFKJohn F Kennedy International
         DISTANCE:  5326 KM
         FLIGHT COST PER SEAT 372.82
         FLIGHT COST:  67107.6

         FLIGHT INCOME:  104000
         FLIGHT PROFIT:  36892.4

---
MENU
1: Enter airport details
2: Enter flight details
3: Enter price plan and calculate profit
4: Clear data
5: Quit
---

Please enter your choice (1-5): 4

You have chosen to Clear Data.
Cleared all data. You can start over or exit using Option 5 in the main menu.

---
MENU
1: Enter airport details
2: Enter flight details
3: Enter price plan and calculate profit
4: Clear data
5: Quit
---

Please enter your choice (1-5): 5
You have chosen to quit
Bye!
Thank you for using this program!
>>>
```

Testing

<table>
<tr><th>Test</th><th>What am I testing?</th><th>Expected result</th><th>Pass/Fail</th><th>Do I need to change my program? If so, how?</th></tr>
<tr><td>1</td><td>Airport Details</td><td>To display and check the entered airport details entered by the user.</td><td>Fail</td><td>Sometimes I entered the UK airport in lowercase (for LJL I entered ljl) and this displayed an error on the screen.
So, I used the .upper and .lower functions in my code to allow both upper and lower case. This gives the user more freedom as to how they want to enter their data. (refer function below)
<pre>#Is the UK Airport code valid?
if UKAP.upper() == "LJL" or UKAP.upper() == "BMI":</pre>I did the same thing for the Overseas Airport in the program.</td></tr>
<tr><td>2</td><td>Flight Details</td><td>To display and check the entered flight details entered by the use and sensibly calculate the number of standard class seats on the chosen aircraft.</td><td>Fail</td><td>When testing my program, I realised that the number of standard class seats was not being calculated correctly. I tried to add a print statement for the variable that the number of standard class seats were stored in but that was printing a 0.
So, I went back to the subroutine which I had specifically made for calculating the number of standard class seats: (inside the subroutine below)
<pre>global StdSeats
global UserEntryList
global FstSeats
StdSeats = int(act[4]) - (fSeat*2)
UserEntryList[5] = StdSeats #store
return StdSeats</pre>Then I realised that my variable for first class seats is different in the code (FstSeats and fSeat) – syntax error. So, I changed them both to the same variable.
<pre>global StdSeats
global UserEntryList
global FstSeats
StdSeats = int(act[4]) - (FstSeats*2)
UserEntryList[5] = StdSeats #stores t
#print("inside cal_std2,",StdSeats)
#print(UserEntryList)
return StdSeats</pre>And now it works! After this I checked my whole code (piece by piece to ensure I have used all the correct variables and to prevent any syntax errors from occurring again)</td></tr>
<tr><td>3</td><td>Enter Price Plan and Calculate Profit</td><td>Flight cost per seat, Flight cost, Flight income, Flight profit</td><td>Pass</td><td>A bit of formatting.</td></tr>
<tr><td>4</td><td>Clear Data</td><td>Clears all variables, arrays</td><td>Pass</td><td>No problems whilst testing.</td></tr>
<tr><td>5</td><td>Exit</td><td>Exits the program</td><td>Pass</td><td>No problems whilst testing.</td></tr>
</table>

Evaluation

How successful was my program?

My program was quite successful as it ended up working correctly.

Any enhancements and refinements to the solution

I refined my code a bit by formatting it so that it is easier to read. I also commented on the important parts of my code so that I am aware of what each section of the code is actually doing.

What new skills have I developed?

Firstly, I had never used a library before, so now I am aware of how to use it and its difference to an array. Secondly, I had used error handling(try block) code for the first time and am now aware of its purpose and what it does.

Any initiative of your own?

I have also used the imported the colour module to change the colour of some of my outputs to make it more appealing for the user. Additionally, I learnt how to use a USE-CASE DIAGRAM and have used it for the design and planning stage for my code.

Softwares used

Python3…

Libraries

Microsoft word

For Flow charts: https://www.lucidchart.com/

For Use Case diagram: Visual Paradigm Online (visual-paradigm.com)