**Pseudocode**

Import csv

table1 : ARRAY

table2 : ARRAY

table3 : ARRAY

CLASS **etf**: #create own function

FUNCTION removeDuplicated (array: ARRAY ):

Duplist: ARRAY

FOR num 🡨 array do:

IF num not in Duplist:

Duplist.append( num )

NEXT num

RETURN Duplist

FUNCTION sorting (array: ARRAY):

Sortlist: ARRAY

WHILE array do:

Smallest = min( array )

Sortlist.append (smallest)

array.pop (array.index(smallest) )

RETURN Sortlist

FUNCTION datechecker (dd: int, mm:int, yyyy:int):

IF mm 🡨 ["1", "3", "5", "7", "8", "10", "12"] THEN:

IF dd 🡨 1 to31 THEN:

RETURN dd/mm/yyyy

ELIF mm 🡨["4", "6", "7", "9", "11"] THEN:

IF dd 🡨 1 to 30 THEN:

RETURN dd/mm/yyyy

ELIF mm is 2 THEN:

IF yyyy % 4 == 0 and (yyyy % 100 != 0 or yyyy % 400 == 0) THEN:

IF dd 🡨 1 to 29 THEN:

RETURN dd/mm/yyyy

ELIF dd 🡨 1 to 28 THEN:

RETURN dd/mm/yyyy

ELSE: RETURN FALSEs

FUNCTION **loadTestingdata**():

Define global table1

Define global table2

Define global table3

Using csv.reader to read excel files

Each table[i] add data line by line

Only table3 make it be the same colomn with table1 (easier to read and search)

Table3.insert ‘col2’ and ‘col4’ and ‘col5’

FUNCTION **Pet&Wild():** # make the big table include pet and animal

RETURN table1+table3

FUNCTION **ID\_checker(signal: String,ID: String):**

Set signal to ‘13L’: IF ID same with the ID of Pet&Wild () THEN RETURN i

Set signal to ‘13B’ : IF ID same with the ID of Pet&Wild () THEN RETURN True

set signal to ‘1L’ ‘2L’ ‘3L’: IF ID same with the ID of table1/2/3 THEN RETURN i

Set signal = ‘1B’ ‘2B’ ‘3B’: IF ID same with the ID of table1/2/3 THEN RETURN True

FUNCTION List\_element\_match ( string: String, column1: int, column2: int):

Temparray: Array

FOR i 🡨 1 to length of Pet&Wild table DO:

IF string in Pet&Wild() [i] [column1] THEN:

FOR j 🡨 1 to length of table2 DO:

IF ID of table2 same with ID of Pet&Wild() THEN:

Temparray will be added some elements from table2[j][colum2]

OUTPUT ( removeDuplicated ( Sorting ( Temparray) ) )

FUNCTION List\_animal\_adopt (string: String):

Temparray: Array

FOR i 🡨 1 to length Pet&Wild DO:

IF Pet&Wild() [i][1] == string THEN :

IF booleen(merge\_animal()[i][3]) and bool(merge\_animal()[i][4] and bool(merge\_animal()[i][5])) == True THEN:

temp\_array.append(merge\_animal()[i][0])

temp = ETF.Sorting(temp\_array)

for i in range(len(temp)):

temp\_Bo = mBoAni(ID\_checker('13L',temp[i]))

return temp\_Bo

FUNCTION List\_animal\_return\_funct (temp\_array: ARRAY):

Temp\_after: ARRAY

FOR i 🡨 1 to length Pet&Wild table:

FOR j 🡨 0 to length temp\_array :

IF temp\_array[j] in Pet&Wild column [9]:

Temp\_after add Pet&Wild ID

THEN do Sorting

FUNCTION edit\_details( which\_table: ARRAY,col : int ):

rowID 🡨 INPUT

enter 🡨 INPUT

IF ID\_checker(which\_table,rowID) is TRUE:

THEN set line is ID\_checker(which\_table,rowID)

ovwrt\_element\_details(which\_table,line,col,enter)

Function list\_data\_ID(ID: String):

IF ID\_checker('13B',ID) is True:

Set line is ID\_checker('13L',ID)

RETURN OUTPUT board

FUNCTION list\_ID()

ASK : String 🡨 INPUT

IF INPUT is 'Yes':

ID: String 🡨 INPUT

list\_data\_ID(ID)

FUNCTION list\_ppl\_abused():

List\_element\_match ("Abused",6,6)

FUNCTION list\_ppl\_abandoned():

List\_element\_match ("Abandoned ",6,7)

FUNCTION list\_cat\_adopt():

List\_animal\_adopt("Cat")

FUNCTION list\_dog\_adopt():

List\_animal\_adopt("Dog")

FUNCTION list\_ani\_return():

temp\_array = ["return","zoo"]

List\_animal\_return\_funct(temp\_array)

FUNCTION add\_new\_line():

Input\_table 🡨 INPUT

write\_entry(input\_table)

FUNCTION add\_new\_pet():

write\_entry(1)

FUNCTION add\_new\_wild():

write\_entry(2)

FUNCTION edit\_ID ():

edit\_details( 1, 0 ):

FUNCTION edit\_type ():

edit\_details( 1, 1 ):

FUNCTION edit\_breed ():

edit\_details( 1, 2 ):

FUNCTION edit\_vaccinated ():

edit\_details( 1, 3 ):

FUNCTION edit\_neutered ():

edit\_details( 1, 4 ):

FUNCTION edit\_mircochip ():

edit\_details( 1, 5 ):

FUNCTION edit\_reason\_admiss ():

edit\_details( 1, 6 ):

FUNCTION edit\_arrival ():

edit\_details( 1, 7 ):

FUNCTION edit\_departure ():

edit\_details( 1, 8 ):

FUNCTION edit\_destination ():

edit\_details( 1, 9 ):

FUNCTION edit\_address ():

edit\_details( 1, 10 ):

FUNCTION edit\_surgery ():

edit\_details( 2, 1 ):

FUNCTION edit\_surgery\_date ():

edit\_details( 2, 2 ):

FUNCTION edit\_medication ():

edit\_details( 2, 3 ):

FUNCTION edit\_ medication\_start():

edit\_details( 2, 4 ):

FUNCTION edit\_ medication\_finish():

edit\_details( 2, 5 ):

FUNCTION edit\_ppl\_abused():

edit\_details( 2, 6 ):

FUNCTION edit\_ppl\_abandoned ():

edit\_details( 2, 7):

CLASS main:

FUNCTION mainMenu():

C: int

C 🡨 INPUT

IF C == 1:

List\_ID()

mainMenu()

IF C ==2:

addMenu()

IF C ==3:

editMenu()

IF C==4:

produceMenu()

IF C==0:

Exit()

FUNCTION addMenu():

C: int

C 🡨 INPUT

IF C == 1:

Add\_new\_line()

IF C == 2:

Add\_new\_pet()

IF C == 3:

Add\_new\_wild()

FUNCTION produceMenu():

C: int

C 🡨 INPUT

IF C == 1:

list\_ppl\_abused ()

IF C == 2:

list\_ppl\_abandoned ()

IF C == 3:

list\_cat\_adopt ()

IF C == 4:

list\_dog\_adopt ()

IF C == 5:

list\_ani\_return ()

FUNCTION editMenu():

C: int

C 🡨 INPUT

IF C == 1:

edit\_ID ()

IF C == 2:

edit\_type ()

IF C == 3:

edit\_breed()

IF C == 4:

edit\_vaccinated()

IF C == 5:

edit\_neutered()

IF C == 6:

edit\_microchip()

IF C == 7:

edit\_reason\_admiss()

IF C == 8:

edit\_arrival()

IF C == 9:

edit\_departure()

IF C == 10:

edit\_destination()

IF C == 11:

edit\_address()

IF C == 12:

edit\_surgery()

IF C == 13:

edit\_surgery\_date()

IF C == 14:

edit\_medication()

IF C == 15:

edit\_medication\_start()

IF C == 16:

edit\_medication\_finish()

IF C == 17:

edit\_ppl\_abused()

IF C == 18:

edit\_ppl\_abandoned()