QUEENSBOROUGH COMMUNITY COLLEGE
The City University of New York
Department of Engineering Technology

Programming Exercises – Decision Structure

1. A - O, determine the output displayed by the lines of code where *a, b, c = 2, 3, 0*. Save your code as *PE6_1.py*.

A	print(a ** b == b ** a)	B	print(a < b or b < a)
Output		Output	
C	print('dog' > 'cat' + 'mouse')	D	print('Car' < 'Train')
Output		Output	
E	print((a == b) and ((a * a < b * b) or (b < a) and (2 * a < b)))		
Output			
F	print((a <= b) or ((a * a < b * b) or (b < a) and (2 * a < b)))		
Output			
G	print(not ((a < b) and (a < (b + a))))		
Output			
H	print("small" > "large" and (not c))		
Output			
I	print(isinstance(c, int))	J	print(isinstance(3.14, float))
Output		Output	

K	if (a < b < c):
	    b = c + a
	else:
	    b = c * a
	print(b)
		L	if ('A' in 'apple'):
			    print("A as apple.")
			else:
			    print('Oops, not there.')

Output		Output	

M	x = 6
	if (x < 0):
	    print('negative')
	else:
	    if (x == 0):
	        print('zero')
	    else:
	        print('positive')
		N	n = 1
			if n <= 9:
			    print ("Less than ten.")
			elif n == 1:
			    print("Equal to one.")

Output		Output	

O	let = input("Enter A, B or C: \n")
	let = let.upper()
	
	if (let == 'A'):
	    print('\nA, my name is Alice.')
	elif (let == 'B'):
	    print('\nTo be, or not to be.')
	elif (let == 'C'):
	    print('\nOh, say, can you see.')
	else:
	    print('\nInvalid letter.')		

Output			

2. Write an *if-elif-else* chain that determines a person's stage of life.
   a) Set your age for the variable age.
   b) If the age is less than 0, print an error message, *invalid age*.
   c) If the age is less than 2 years old, print a message, *you're a baby*.
   d) If the age is at least 2 years old but less than 4, print a message, *you're a toddler*.
   e) If the age is at least 4 years old but less than 13, print a message, *you're a kid*.
   f) If the age is at least 13 years old but less than 20, print a message, *you're a teenager*.
   g) If the age is at least 20 years old but less than 65, print a message, *you're an adult*.
   h) If the age is 65 or older, print a message, *you're an elder*.

```
Example Output
Age = 20
You're an adult.
```

3. Implement the following to print a greeting to each user after they log in to a website.
   a) Make a list of **five** *usernnames*, including the name *"admin"*.
   b) Loop through the list and print a greeting to each user.
   c) If the username is "admin", print a special greeting, such as
   Hello Admin, would you like to see a status report?
   d) Otherwise, print a generic greeting, such as
   Hello Eric, thank you for logging in again!
   *e) Implement if the list is empty by printing the message, *We need to find some users.*

```
Example Output 1:
Hello Tom, thank you for logging in again!
Hello Jerry, thank you for logging in again!
Hello Bob, thank you for logging in again!
Hello Dora, thank you for logging in again!
Hello ADMIN, would you like to see a status report?


Example Output 2 (if the list is empty)
We need to find some users.
```

4. Implement the following to simulate how websites ensure that everyone has a unique username.
   a) Make a list of five or more usernames called *current_users*.
   b) Request an input of username.
   c) Print a message, *Sorry XXX, that name is taken* and also display the current user list if the input username has already been used. *XXX is the input user name.*
   d) Print a message, *Great, XXX is still available* and also display the updated user list if the username has not been used.
   e) Make sure your comparison is case insensitive. If 'John' has been used, 'JOHN' or 'john' should not be accepted.

```
Example Output 1
Enter your user name: TOM
Sorry TOM, that name is taken.
Current users: ['admin', 'tom', 'jerry', 'Dora', 'GEORGE']
```

```
Example Output 2
Enter your user name: curiousGeorge
Great, curiousGeorge is still available.
Updated users: ['admin', 'tom', 'jerry', 'Dora', 'GEORGE', 'curiousGeorge']
```

5. Implement the following to search a letter in a list.
   a) Create a list named vehicles: car, Truck, boat, PLANE.
   b) Request a user input for a search letter.
   c) Use the decision structure in a for loop to search all the items which contains the input letter (ignoring case) in the list.
   d) Print the item and its position in vehicles if it exists. Otherwise, print the statement indicating it does not contain the letter to search.
   e) Print the error message if more than one letter is entered.

```
Example Output 1
Vehicles = ['car', 'Truck', 'boat', 'PLANE']
Enter a search letter: a
car contains 'a' and it is in position 0.
Truck does not contain 'a'.
boat contains 'a' and it is in position 2.
PLANE contains 'a' and it is in position 3.
```

```
Example Output 2
Vehicles = ['car', 'Truck', 'boat', 'PLANE']
Enter a search letter: A
car contains 'A' and it is in position 0.
Truck does not contain 'A'.
boat contains 'A' and it is in position 2.
PLANE contains 'A' and it is in position 3.
```

```
Example Output 3
Vehicles = ['car', 'Truck', 'boat', 'PLANE']
Enter a search letter: z
car does not contain 'z'.
Truck does not contain 'z'.
boat does not contain 'z'.
PLANE does not contain 'z'.
```

```
Example Output 4
Vehicles = ['car', 'Truck', 'boat', 'PLANE']
Enter a search letter: abc
Invalid search letter.
```

6. A – D, identify the errors and **rewrite** the statement in the correct syntax. Save your code as *PE6_6.py*.

A	`n = eval(input("Enter a number: "))`
	`if n = 7:`
	`    print("The square is", n*2)`

Debug	

B	`n = 6`
	`if n > 5 and < 9:`
	`    print("Yes")`
	`else:`
	`    print("No")`

Debug	

C	`major = "Computer Science"`
	`if major == "Engineering Technology" Or "Computer Technology")`
	`    print("Yes")`

Debug	

D	`a, b = 1, 1.0`
	`if a = b:`
	`    print("same")`

Debug	

Reference: Relational Operators

==	!=	<	<=	>	>=	in	not in
equal to	not equal to	less than	less than or equal to	greater than	greater than or equal to	substring of	not a substring of

Reference: Logical operators and, or, not Table

A	**B**	**A and B**
True	True	**True**
False	True	**False**
True	False	**True**
False	False	**False**

A	**B**	**A or B**
True	True	**True**
False	True	**True**
True	False	**True**
False	False	**False**

A	**not**
False	**True**
True	**False**

Reference: Standard ASCII Table & Extended ASCII