## 2022 SH1 Promo Sample Solutions

**Q1 [8]**

(a) [2]

* Base case, line 5 to line 9
* Recursive case, line 10 to line 19

(b) [1]

* Returns all the different permutations of "0" and "1" of string with length n
* OR base 2 integers of length n

(d) [3]

| **Call Frame** | **result** | **ret** | **RETURN** |
| --- | --- | --- | --- |
| DoIt(3) | **["00","01",**  **"10","11"]** | **["000","001","010","011","100","101","110","111"**  **]** | ["000","001","010","011","100","101","110","111"  ] |
| DoIt(2) | ["0","1"] | **["00","01","10","11"]** | ["00","01","10","11"] |
| DoIt(1) |  |  | ["0","1"] |

**(e) [**2]

* O(2n )
* There are n recursive calls.

Base case returns 2 strings,

At each recursive case, for loop iterates 2 times over the results of the recursive call therefore the total number of times append is executed is:

2 + 2\*(2) + 2\*(2\*2) + 2\*(2\*2\*2) + .. 2n. Largest term is 2n. therefore the run time complexity is O(2n )

This is actually a convergence geometric series with first term a = 2 and common ratio r=2,

is O(2n )

**Q2 [23]**

1. [4]

* Code repetitions for get\_name(), get\_age(), get\_contact\_num()
* No setters methods, which means that the objects after instantiation are immutable. This is not practical if any of the attributes need to be modified.
* The attribute age is not practical to be implemented as an attribute as it is relative to the current date, instead DateOfBirth should be use as an attribute and age should be a method that is calculated base on the current date.

1. [2]

| Person |
| --- |
| -name  -age  -contact\_num |
| +constructor(name,age,contact)  +get\_name()  +get\_age()  +get\_contact\_num()  +set\_name()  +set\_age()  +set\_contact\_num() |



| Staff |  | Student |
| --- | --- | --- |
| -designation |  | -form\_class |
| +constructor(name,age,contact, designation)  +get\_designation()  +set\_designation() |  | +constructor(name,age,contact,form\_class)  +get\_form\_class()  +set\_form\_class() |

1. **[7]**

FUNCTION BBSort(All:List) RETURNS None // List index starts at 0

FOR i = 0 TO LEN(All)-1 // LEN returns size of List

FOR j = 0 TO LEN(All)-1-i:

IF Type(All[j], Student) AND Type(All[j+1], Staff) THEN

// Type checks the data type of an object instance

Swap(All, j, j+1) //swap positions in List

ELSEIF Type(All[j], Student) AND Type(All[j+1], Student) THEN

IF All[j+1].age < All[j].age THEN

Swap(All, j, j+1) //swap positions in List

ENDIF

ENDIF

ENDFOR

ENDFOR

ENDFUNCTION

**1m Outer loop**

**1m Inner loop**

**2m IF to check Student/Staff adjacency and swap**

**3m IF to check Student/Student adjacency and swap if age is greater**

**(d)** Any 2 [2]

* Inline
* Stable
* O(n2)

(e)[2]

* A very large array which is already partially sorted.

(f)[3]

FUNCTION merge(A:List,B:List) RETURNS List:

ret🡨[] //Empty List

WHILE LEN(A) and LEN(B) DO

IF Type(A[0],Staff) AND Type(B[0],Student) THEN

tmp 🡨 Pop(A,0) // remove first element from List

Append(ret, tmp ) // append tmp to List

ELSE IF isinstance(A[0],Student) AND Type (B[0],Student) THEN

IF A[0].age < B[0].age THEN

tmp 🡨 Pop(A,0)

Append(ret, tmp )

ELSE

tmp 🡨 Pop(B,0)

Append(ret, tmp )

ENDIF

ELSE

tmp 🡨 Pop(B,0)

Append(ret, tmp )

ENDIF

ENDWHILE

ret 🡨 Concatenate (ret,A,B) #[1]

return ret

**1m for while loop**

**1m for adding the smaller of the 2 elements in A, B list**

**1m for concatenating with left overs from A, B**

g)

* linear search O(n). [1]
* the list is not sorted by name, search is by name, binary search can only be used if they are first sorted by name then perform binary search, which will be O(nlogn) + O(log n ) = O(nlogn) which is worse than linear search. [2]

**Q3[9]**

1. O(n2) [1]
2. Line 5 [2]

while j > 0 and A[j-1] > item

OR

Line 6

A[j], A[j-1] = A[j-1], A[j]

(c) O(nlogn) [1]

(d) [2]

Using either the median or a random element in the array. Add the following lines after line 4:

mid = len(A)//2

A[0], A[mid] = A[mid], A[0]

(e) [3]

Run time complexity is described using the big O notation which is used to classify algorithms according to how their run time requirements grow as the input size grows. [1]

When you use timeit to measure the execution time, the execution time is dependent not only on the input, but the compiler/interpreter of the programming language as well as the hardware capabilities of the machine in which the code runs on. [1]

OR

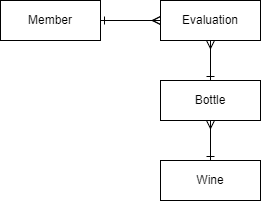
The timeit module measures how long it takes a program to run. [1]

Time complexity(in Big O notaion) is a description of the asymptotic behavior of the execution time as input size tends to very large number.[1]

The input list which is used in the example, is too small for the execution time of the two algorithms to correlate with the run time complexity of O(n2 ) of insertion sort and O(nlogn) of quick sort. [1]

Q4[21]

(a)[5]



Member, Wine [1]

Bottle , Evaluation [1]

Member 1 – N Evaluation [1]

Bottle 1 – N Evaluation [1]

Wine 1 – N Bottle [1]

(b)[5]

Member (ID, Name, Email,Contact\_ Address ) [1]

Bottle ( Name\*, Vintage\*, Number, Consumed ) [1]

Wine ( Name, Vintage, Country, Alcohol ) [1]

Evaluation (ID\*, Name\*, Vintage\*, BottleNumber\*, Score, Date ) [2]

\*Foreign key

(c)[2]

SELECT DISTINCT Wine.Country FROM Wine

OR

SELECT Wine.Country FROM Wine GROUP BY Country

(d)[4]

SELECT Member.Name, COUNT(\*) "Number Times" FROM Evaluation

INNER JOIN Member ON Evaluation.ID = Member.ID

GROUP BY Evaluation.ID

HAVING COUNT(\*) > 1

-JOIN [1]

-AGGREGATE COUNT()[1]

-GROUP BY [1]

-HAVING [1]

(e)[1]

DELETE FROM Bottle WHERE Bottle.Consumed = 1

(f)[2]

REFERENTIAL INTEGRITY ERROR,

There is a Evaluation record that has a FK referencing a record in Bottle.

(g)[2]

Archive the old evaluation records by removing and storing them on an external or cloud storage.

INSERT INTO TableFromExternalStorage WHERE Evaluation.Date > date 5 years ago

DELETE FROM Evaluation WHERE Evaluation.Date > date 5 years ago

Q5 [4]

* Use a dictionary [1] or an ADT
* Key is a tuple (row, column) and value is the value of the element in array [1]
* If key is not found, value in matrix is 0 [1]
* Dimension of the matrix is stored in the dictionary or as an attribute in the ADT[1]

#Assume Matrix as defined in Paper 2

A = Matrix(10,2)

A.set\_row(1,[0,2])

A.set\_row(2,[3,0])

for i in range(3,10):

A.set\_row(i,[0,0])

A.set\_row(10,[9,10])

## Create the sparse matrix from a matrix

sparse\_matrix = {}

sparse\_matrix["dimension"] = A.get\_dimension()

row\_count, col\_count = A.get\_dimension()

for r in range(1, row\_count+1):

for c in range(1, col\_count+1):

if A.get\_value(r,c) != 0:

sparse\_matrix[r,c] = A.get\_value(r,c)

## access the sparse matrix for row 1, col 1

## if zero value, a None will be returned

sparse\_matrix.get((1,2))

## Recreate the matrix from sparse matrix

row\_count, col\_count = sparse\_matrix["dimension"]

sparse\_matrix.pop("dimension")

B = Matrix(row\_count, col\_count)

B.init(0)

for k in sparse\_matrix:

B.set\_value(k[0],k[1], sparse\_matrix[k])