2020 SH2 Promo H2 Computing 9569 Paper 1 Marking Scheme/Solutions

Q1 [**13]**

|  |  |
| --- | --- |
| **(a) [4]** | FUNCTION Transpose(A:ARRAY[1:M,1:N] OF INTEGER): RETURNS ARRAY[1:N,1:M] OF INTEGER  DECLARE Matrix AS ARRAY[1:N,1:M] OF INTEGER **\\[1]**  FOR i = 1 TO M DO **\\[1]**  FOR j = 1 TO N DO **\\[1]**  Matrix[j][i] 🡨 A[i][j] **\\[1]**  ENDFOR  ENDFOR  RETURN Matrix  ENDFUNCTION |
| **(b) [3]** | |  |  |  | | --- | --- | --- | | Category | Input | Expected Output | | Normal | [ [1,2,3], [4,5,6] ] | [ [1,4],[2,5],[3,6] ] | | Boundary | [ [1] ],[[]] | [[1] ],[[]] | | Invalid | [ [1,2], [1,2,3] ], [] | Indexing Error |   **1m – each category** |
| **+(c) [3]** | * Stores only those elements that have a value. **[1]** * Use a Dictionary data structure. A Dictionary data structure has a set of keys and each key has a single associated value. It can be represented as follows: { key1: value1, key2:value2 }. * Using the row and column to form a tuple and use it as a key to store element's value. * The sparse matrix shown in 1(c) can be represened as (assume index starts at 0)   { (1,0):5, (1,1):8, (2,2):3, (3,1):6 }   * Stores the original dimension as a tuple (4,4),so that the 0s can be retrieve indirectly.   **[2] Any form of data structure to map elements that have values from their indexes to their values.** |
| (**d) [3]** | **Sample**  // A DICTIONARY Data Structure, D stores data as  // D[key] = value  // A TUPLE is a data structure containing 1 or more values represented as (x,y), where x and y are integers  DECLARE SparseMatrix AS DICTIONARY  DECLARE Dimension as TUPLE  DECLARE A as ARRAY[1:M] of ARRAY[1:N] OF INTEGER  DECLARE Row, Col as INTEGER  FOR Row = 1 to M DO  FOR Col = 1 to N DO  IF A[Row][Col] <> 0 THEN  SparseMatrix[(Row,Col)] 🡨 A[Row][Col]  ENDIF  ENDFOR  ENDFOR  Dimension 🡨 (M,N) |

Q2[13]

|  |  |  |
| --- | --- | --- |
| **(a) [1]** | **Inplace, elements are swapped or exchanged within the array. No extra storage used.**  **Non-inplace , extra storage is used** |  |
| **(b)**  **[4]** | Bubble or Insertion sort  2 loop [**1]**  Compare using Noise.Level attribute **[1]**  Correct start end for outer loop [**1]**  Correct start, end for inner loop [**1]** |  |
| **(c) [5]** | Assume that the sorted array contains only integers.  A TUPLE is a data structure containing 1 or more values of different data types represented as (x,y), where x, y are values.  This Algorithm assumes that there are no duplicate values in the array. See the Python code for duplicate values  FUNCTION BinSearch(L:ARRAY[0:N] OF INTEGER, Target: INTEGER) RETURNS TUPLE  DECLARE LB, UB, Floor, Ceiling, Mid  LB 🡨 0  UB 🡨 N  Floor 🡨 None  Ceiling 🡨 None  WHILE LB <= UB DO  Mid 🡨 (LB + UB) DIV 2  IF L[Mid] > Target THEN  Ceiling 🡨 L[Mid]  UB 🡨 Mid -1  ELSE IF L[Mid] < Target  Floor 🡨 L[Mid]  LB 🡨 Mid + 1  ELSE // Assumes that there are no duplicates  IF Mid > 0 THEN  Floor 🡨 L[Mid - 1]  ELSE  Floor 🡨 NONE  ENDIF  IF Mid < N THEN  Ceiling 🡨 L[Mid + 1]  ELSE  Ceiling 🡨 NONE  ENDIF  RETURN (Floor, Ceiling)  ENDIF  ENDWHILE  RETURN (Floor, Ceiling)  ENDFUNCTION  #SAMPLE IN Python Code  def bsearch(L, target):  lb = 0  ub = len(L)-1  floor = None  ceiling = None  while lb <= ub:  mid = (lb + ub)//2  if L[mid] > target:  ceiling = L[mid]  ub = mid -1  elif L[mid] < target:  floor = L[mid]  lb = mid + 1  else:  f\_index = mid  while f\_index > -1 and L[f\_index] == target:  f\_index -=1  c\_index = mid  while c\_index < len(L) and L[c\_index] == target:  c\_index +=1  floor = L[f\_index] if f\_index >= 0 else None  ceiling = L[c\_index] if c\_index < len(L) else None  return (floor, ceiling)   * **Initialisation of pointers, lb, ub, floor, ceiling mid [1]** * **Outer loop with correct exit condition [1]** * **Condition for target < element, set ceiling to mid element [1]** * **Condition for target > element, set floor to mid element [1]** * **Condition for target = element, set floor, ceiling to elements before and after mid and return [1]** | |
| **(d) [3]** | - client socket program will get data from sensor and send Timestamp and noise level data to server program.**[1]**  - server socket program to listen for incoming client connection. It will run in a loop to accept data from client and writes to a file or data base. [**1]**  - for processing the data, a program can be written **[1]** |  |

Q3**[12]**

|  |  |  |
| --- | --- | --- |
| **(a) [3]** | **Loop to initialise the free nodes[1]**  **Update Node.Next to point to next element index in Array, except last Node, which will have a value of -1 [1]**  **Update Start, Free[1]** |  |
| **(b) [3]** | **Check for Full List return False if full [1]**  **Get next available Node from Free pointer, update Free pointer to point to next available Node [1]**  **Update Node with data, next to point at Start, Start to point at new Node return True [1]** |  |
| **(c) [4]** | **Loop with exit condition [1]**  **If L.Buffer[current].Data = data , update previous Node.next to point to current Node.next. update Deleted flag [1]**  **If L.Buffer[L.Start].Data == data, update L.Start [1]**  **Initialise/Update previous, current [1]** | |
| **(d) [2]** | **When the program is running on a platform that has limited memory [1]**  **example an embeded system like adruino, raspberry pie, [1]**  **-1m if expain O(1) access vs O(n) access** |  |

Q4[23]

|  |  |  |
| --- | --- | --- |
| **(a)**  **[2]** | * **All Cloud Services are Web Services, ie uses http, html, url, but not all web services are Cloud Services [1]** * **A cloud service has a business model, where services are delivered on the web. These services include, Software, Infrastructure and Platform services [1]** * **A web app solves a specific problem [1]** * **A cloud service offers a generic service [1]**   **Any 2** |  |
| **(b) [2]** | **IaaS, Customer provide the applications, operating system , database Service Provider provides the physical hardware, servers, storage devices, network devices [1]**  **PaaS, Customer builds their application on the database, operating system and hardware provided by the Service Provider [1]** |  |
| |  | | --- | | Customer | | -id  -services | | +getId()  +getServices() |   **(c) [10]**   |  | | --- | | Service | | -type | | +constructor()  +getType() |  |  | | --- | | ManagedInstance | | -TPS  -databaseSize  -pricePerHour  +monthlyRunTime | | +constructor()  +getTPS()  +getDatabaseSize()  +getPrice()  +setPrice()  +calMonthlyBill()  +run() |  |  | | --- | | VirtualDisk | | -size  -pricePerMonth | | +constructor()  +getSize()  +getPrice()  +setPrice() |  |  | | --- | | VirtualMachine | | -numberOfCPU  -RAM  -virtualDisks  -pricePerHour  +monthlyRunTime | | +constructor()  +getType()  +getNumberOfCPU()  +getRAM()  +getPrice()  +setPrice()  +calMonthlyBill()  +addDisk()  -removeDisk()  +run() |  |  | | --- | | PremiumMI | | -pricePerMonth | | +getPrice()  +setPrice()  +calMonthlyBill()  +backup()  +recover() |  |  | | --- | | PremiumVM | | -pricePerMonth | | +getPrice()  +setPrice()  +calMonthlyBill()  +backup()  +recover() |   Customer.getServices(): returns a list of the Services subscribed by customer  VirtualMachine/ManagedInstance.monthlyRunTime: number of hours used  VirtualMachine.calMonthlyBill: calculate bill based on monthlyRunTime \* pricePerHour + virtualDisk.priceperMonth  VirtualMachine.virtualDisks: List of VirtualDisk  PremiumVM/PremiumMI.calMonthlyBill: super().calMonthlyBill + self.pricePerMonth  PremiumVM/PremiumMI.backup: backup storage  PremiumVM/PremiumMI.recover: restore the storage from a previous backup.   * **VirtualMachine, VirtualDisk, ManagedInstance [2] -1m for missing 1** * **Base class, Service for VM,VD,MI [2] -1m for wrong inheritance relationship. Accept derived VM and MI classes instead of type** * **Derived classes PremiumVM, PremiumMI [2]-1m each** * **Correct attributes/access specifier for all requirements [2] -1m for each missing or wrong attribute** * **Correct methods for all requirements [2] -1m for each missing or wrong method** | | |
| **(d)**  **[4]** | **- inheritance [1]**   * **VM, VirtualDisk,MI are all cloud services with a common set of attributes, name and type.** * **PremiumVM/MI have are VM/MI with additiona functions**   **[1] Any 1**  **- polymorphism [1]**   * **calMonthlyBill is in both VM and MI but they have different functions and algorithm to calculate the monthly bill** * **recover and backup in premiumVM/MI but they have different functions and algorithm [1] Any one**   **-encapsulation[1]**   * **data hiding of attributes using a private accessor like size and type prevents the instances default values from being overridden manually** * **attributes and methods within the same object are bound together such that the method can only access the attributes that belongs to the same object, example calMonthlyBill can only access the monthlyRunTime in the same object to calculate the monthly bill Any one [1]**   **Any 2 of inheritance, polymorphism, encapsulation** | |
| **(e) [3]** | **Example solutions:**  **Data stored in service provider allows the provider to use the customer data without knowledge of customer [1]**  **When customers unsubscribe their services from the provider, how are their data being handle. [1]**  **How are the customers data secure by the service provider [1]**  **Any 2**  **Mitigation:**   * **Customer can encrypt the contents before storing them** * **Customer can request the provider to provide a service agreedment on how the data are handle** * **Customer can store only non-sensitive data on the cloud**   **Any 1 [1]** | |
| **(f) [2]** | **Data can be archived easily by exporting the collections**  **Data schema can be dynamicaly updated**  **The database can be scaled based on customer's workload**  **Any 2, [2]** | |

Q5[21]

|  |  |  |
| --- | --- | --- |
| **(a)**  **[6]** | **Owner(Optional),Vendor, Customer [1]**  **FoodItem, 1-M with Vendor [1]**  **Vendor-FoodType [1] , M-1 relationships with Vendor, FoodType[1]**  **Order[1], M-1 Relationships with FoodItem, Customer [1]**  **Alternative: Order-Fooditem** |  |
| **(b) [6]** | **Customer** ( Name : TEXT, Address:TEXT, Contact:TEXT) ID is accepted  **Owner**(Name:TEXT,Email:TEXT,Contact:TEXT) *[Accept this as attributes in Vendor table]*  **Vendor** ( StoreName: TEXT, StoreAddress:TEXT, RegNo: TEXT, OwnerContact\*:TEXT)  **FoodItem**( VendorRegNo\*: TEXT, ItemNo: INTEGER, ItemName: TEXT, Description: TEXT, Price: REAL)  **FoodType**( Id: INTEGER, Type: TEXT)  **Vendor-FoodType** (RegNo\*: TEXT, Id\*: INTEGER)  **Order** (Contact\*: TEXT, VendorRegNo\*: TEXT, ItemNo\*: INTEGER, Timestamp: DATETIME, Quantity: INTEGER)  \*The order must capture vendor + food item and timestamp  **1m- each, all attributes in web form must be captured, pri/foreign keys are correct. Accept Owner fields as attributes of Vendor** |  |
| **(c)**  **[3]** | User Experience –   * determines whether a user is able to achieve his objective or needs when using the web app. * determines whether a user will revisit the web app   Any one **[1]**   * UI/UX are **interrelated**, a **good UI will contribute to a good UX**, BUT a **good UI does not mean it will have a good UX** [**2]** | |
| **(d)**  **[4]** | All data in the database design are captured **[2]** -1m for 1 missing data  Demonstrates any 2 of the following: **[2]**  Visual Hierarachy, Affordance, Consistentncy, Responsive |  |
| **(e)**  **[2]** | Orders will be timestamped **[1]**  Orders older than 1 year for example will be deleted from the current database and inserted into a less expensive storage solutions like cloud storage database, or exported to be stored in dvd, tapes.[**1]** | |

Q6[14]

|  |  |  |
| --- | --- | --- |
| **(a) [2]** | **LAN – small geographical area, Private any 1**  **WAN – large geographical area, Public any 1** |  |
| **(b)**  **[2]** | **End-devices, use's laptops, mobile devices connect to LAN [1]**  **LAN connect to a ISP network using a WAN link provided by the service provider[1]** |  |
| **(c)**  **[4]** | **A set of rules that determine exchange of data between 2 nodes in the network [1]**  **TCP/IP consists of 4 layers of protocols [1]**  **Application layers deals with end application communicating with end application**  **Transport layer determines whether the communication is reliable or best-effort delivery**  **Network layers determines how packets are routed in the network**  **Physical layers determine the physical hardware involves in the communication**  **Any 2 of the layer correctly described [2]** | |
| **(d) [4]** | **Server connected to a LAN [1]**  **LAN connects to a ISP/Internet[1]**  **Student PC connects to the Internet [1]**  **Teacher PC connects to the Internet [1]** |  |
| **(f)**  **[2]** | **DNS name consists of a hostname and a domain name, it is unique in the public Internet [1]**  **Obtain/register a valid domain name from a service provider [1]** | |