C2 PRELIMINARY EXAMINATION 2020

COMPUTING HIGHER 2 PAPER 1 (9569/01)

SUGGESTED SOLUTION (WITH MARK SCHEME)

|  |  |  |
| --- | --- | --- |
|  | Solution | Marks |
| 1. | 1. One or more computers **acts as server;** Other computers **request service** from server 2. **Advantage**: server can control the access rights of resources. If one client is down, the server and other clients are not affected. Resources can be updated faster, and it is easier to perform backup   **Disadvantage**: If server is down, the whole network is down. Centralized server is more expensive to build up, and requires professional to maintain.   1. DNS provides the directory service to **translation** from hostnames to IP addresses. Answers should provide an overview of the **hierarchical** decentralized naming system. 2. \*allow mistake in the words ‘Discover, Offer, Request, Ack’   Discover: **client broadcast** to look for a DHCP server  Offer: DHCP **server** **offers** an address  Request: **client** **requests to release** the address  Ack: DHCP **server** **sends the address** to the client for acknowledgement | 2M  1M  1M  2M  4M |
| 2. | 1. Any suitable answer 2. Integrity: act with complete discretion when entrusted with confidential information. Unethical behaviour of leaking data or personal information   Responsibility: carry out full responsibility in a professional manner, adhere to guidelines. Unethical behaviour of not carrying out duties  Competence: innovate new technology or learn from other countries. Unethical behaviour of claiming programming proficiency in a language that he/she has never used. | 2M  4M |
| 3. | 1. **Data validation** ensures the input data conforms with the data requirements.  * Range check: 1<= month <= 12 * Format check: date in the format dd/mm/yyyy * Length check: length of password * Presence check: username cannot be left blank * Check digit: last digit of NRIC   **Data verification** ensures the input data matches the original resource  For example, double entry of password, proofread forms before submission     1. Data is **separated** into packets and each packet **independently** find **the best route** to the receiver.   Advantage:   * Different packets may travel in **different** route and thus **more efficient**, **saves bandwidth** and **avoids congestion**. * It is also more **secured** since it became much harder to attack **all the routes** instead of one routes in circuit switching network.  1. Reason for layering: Enables programmers to specialize in a particular layer of the model. Allows for standardized interfaces to be produced by networking vendors  |  |  | | --- | --- | | Layer | Function | | Application Layer | interacts with **end user**, create/format the data package OR  Consists of **applications**/programs and processes that use the network | | Transport Layer | **establish connection** between **applications** of the sender and the receiver OR  provide **end-to-end** data delivery services | | Internet/Network Layer | find the best **route** to deliver the data package OR  defines the datagram and handles the **routing** of data | | Physical/Link/  Network Access Layer | Transform data package to **electrical signals or radio waves** and transmit to the **physical devices** OR  Consists of routines for accessing **physical** **medias** | | 4M  4M  1M  1M-layers in correct order (either way is fine)  4M -1 mark for each layer |
| 4. | 1. (i) 011 0100   (ii) 3416     1. 0000 0100 1011 0001 2. A check digit is a redundant digit or letter calculated from digits of a code number. It is then added to the code number that permits the accuracy of other digits in the code to be checked.   (ii)   |  | | --- | | sum 🡨 0  weight 🡨 7  FOR i 🡨1 TO 5  sum 🡨 sum + INTEGER of account[i] \* weight  weight 🡨 weight – 1  ENDFOR    checkDigit 🡨 ASCII of (account[6])-ASCII of(‘C’) + 1  totalWeightedSum 🡨 sum + checkDigit  IF totalWeightedSum MOD 11 = 0  OUTPUT ‘valid’  ELSE  OUTPUT ‘valid’ ENDIF |  1. Not valid. The total weighted sum (including the cheek digit) is 160, which is not divisible by 11. Check digit should be E and not K. | 1M  1M  1M-correct conversion  1M-16 bits  1M  1M |
| 5  6 | |  | | --- | | # Assume the string index starts at 1  FUNCTION Encode(dataStr : STRING) RETURNS STRING  result 🡨 ‘’  prevChar 🡨 dataStr[1]  prevCount 🡨 1  FOR i 🡨 2 TO LENGTH(dataStr)  currChar 🡨 dataStr[i]    IF currChar = prevChar  prevCount 🡨 prevCount + 1  ELSE  result 🡨 result + prevChar + str(prevCount)  prevChar 🡨 currChar  prevCount 🡨 1  ENDIF  ENDFOR  result 🡨 result + prevChar + str(prevCount)  RETURN result  ENDFUNCTION |   (b)   |  | | --- | | FUNCTION SearchQinP(P:STRING, Q:STRING) RETURNS INTEGER  pSize 🡨 LENGTH(P)  qSize 🡨 LENGTH(Q)    match 🡨 FALSE  index 🡨 1  WHILE(NOT match)AND(index <= pSize – qSize + 1)  extract 🡨 ‘’  FOR i 🡨 index TO (index + qSize - 1)  extract 🡨 extract + P[i]  ENDFOR  IF extract = Q  match 🡨 TRUE  ELSE  index 🡨 index + 1  ENDIF  ENDWHILE  IF match  RETURN index  ELSE  RETURN 0  ENDIF  ENDFUNCTION |   (a)   1. A function which contains a call to itself. 2. Should include at least one terminal case – a case that contains no further calls to the recursive subprogram so that it will not continue indefinitely. 3. Used when the original task can be reduced to a simpler version of itself     (b)   |  |  | | --- | --- | |  | **OUTPUT** | | Hanoi(3, 1, 3)  Hanoi(2, 1, 2)  Hanoi(1, 1, 3)  Hanoi(1, 3, 2)  Hanoi(2, 2, 3)  Hanoi(1, 2, 1)  Hanoi(1, 1, 3) | Move disc from peg 1 to peg 3  Move disc from peg 1 to peg 2  Move disc from peg 3 to peg 2  Move disc from peg 1 to peg 3  Move disc from peg 2 to peg 1  Move disc from peg 2 to peg 3  Move disc from peg 1 to peg 3 |  |  | | --- | | Stack contents: n, i, j  1st call : 3, 1, 3 🡨 SP  2nd call: 2, 1, 2 🡨SP  3, 1, 3  3rd call: 1, 1, 3 🡨 SP  2, 1, 2  3, 1, 3  4th call: 1, 3, 2 🡨 SP  2, 1, 2  3, 1, 3  5th call: 2, 2, 3 🡨 SP  3, 1, 3 |  * variables take different values  at  each recursion   or   * these values must all be preserved for later recall. | 1M |
| 7 | (a)    Node Class Structure :   |  | | --- | | FUNCTION Delete(x, p)  IF LENGTH(x)=0  OUTPUT “List is EMPTY”  ENDIF  IF p=1 #case 1: delete 1st item  currPtr 🡨 Start  Start 🡨 currPtr.next  ELSE #case 2: delete in middle  prevPtr 🡨 null  currPtr 🡨 Start  # find deletion point  FOR n 🡨 1 to p-1  prevPtr 🡨 currPtr  currPtr 🡨 currPtr.next  ENDFOR  prevPtr.next 🡨 currPtr.next  ENDIF  ENDFUNCTION | | FUNCTION Insert(x, item, p)  prevPtr 🡨 null  currPtr 🡨 Start  new 🡨 Node(item, null)    IF p=1 #insert as 1st item  new.next 🡨 currPtr  Start 🡨 new  ELSE #insert anywhere  FOR n 🡨 1 to p-1  prevPtr 🡨 currPtr  currPtr 🡨 currPtr.next  ENDFOR  new.next 🡨 currPtr  prevPtr.next 🡨 new  ENDIF  ENDFUNCTION |   (b)   |  | | --- | | 1. FUNCTION Create()   Create(S)  ENDFUNCTION | | 1. FUNCTION Push(item)   Insert(S, item, 1)  ENDFUNCTION | | 1. FUNCTION Pop() RETURNS STRING   IF IsEmptyList(S)  THEN  RETURN ‘ ‘  ELSE  Temp 🡨 Read(S,1)  DELETE(S,1)  RETURN Temp  ENDIF  ENDFUNCTION |    |  | | --- | | Ii)  FUNCTION Undo() RETURNS STRING  content = ’’ IF undoStack.IsEmpty() = False  content 🡨 undoStack.Pop()  redoStack.Push(content)  ENDIF  RETURN content  ENDFUNCTION | | (ii)  FUNCTION Redo() RETURNS STRING  content =’’  IF redoStack.IsEmpty()= False  content 🡨 redoStack.Pop()  undoStack.Push(content)  ENDIF  RETURN content  ENDFUNCTION | | 1M Length=0 Empty List  1M – del 1st Item  1M – del in middle  1M – find deletion point using for loop  1M – Create Node with item  1M – insert at 1st position  1M – For loop to shift currPtr and prevPtr to the position for insertion  1M – insert at other position |
| 8 | (a)  (i)    3M: 3 tables with correct relationship and connectivities as shown above.  (ii)  Member (Member\_ID, Name, Contact\_No, Price)  Outlet(Outlet\_ID, Outlet\_Address)  ServiceRecord(Service\_ID, Member\_ID, Outlet\_ID, Service\_Date).  1M - Member + Member\_ID as PK  1M – Outlet + Outlet\_ID as PK  1M – ServiceRecord attributes as shown above  1M – ServiceRecord with Service\_ID as PK  1M – ServiceRecord with Member\_ID and Outlet\_ID as FK   1. The field “Service\_Amount” should be added to the ServiceRecord table to capture the amount paid by the customer for each haircut service.   ServiceRecord(Service\_ID, Member\_ID, Outlet\_ID, Service\_Date, Service\_Amount)  Member (Member\_ID, Name, Contact\_No)  1M – Add Service\_Amount in ServiceRecord and remove Price from Member. | |
|  | 1. (i)         1M – Arrow from subclass to superclass in diagram to indicate “member” class as superclass and “Premium Member” as subclass  1M – Polymorphism. Eg. showInfo()  2M – Member with private attributes and public methods  2M – Premium Member with private attributes and public methods  (ii) Encapsulation refers to the combining of data and functions into a single object. The Member class’s attributes (Name, Contact) and methods (getName(), setName(), getContact(), setContact()) that operate on the attributes are bundled together in a class.  (iii)Polymorphism refers to an object’s ability to take different forms. It allows subclasses to have methods with the same name as methods in their superclass. It gives the ability for a program to call the correct method depending on the type of object that is used to call it.  The method showInfo in the subclasses overrides the superclass showInfo. If the subclass object (PremiumMember) is used to call showInfo, then the subclass’s version of the method is invoked. If the superclass object (Member) is used to call showInfo, then the superclass method will be invoked.   |  |  | | --- | --- | | **Web Applications** | **Native Applications/Desktop Application** | | Deployment and maintenance (updates) for a web-based application require deployment on a single set of server machines. | Deployment and any maintenance/patch are done on individual client machines separately. | | Web applications can be accessed from anywhere (most locations), so there is no location constraint. | As desktop are confined to a standalone machine, so they can be only accessed from the machines they are deployed in. | | Web applications are platform-independent, they can work in different types of platforms with the only requirement of a web browser. | Desktop applications need to be developed separately for different platform machines. (Windows, Linux, Unix, Mac etc) | | Web applications are at higher security risks as they are inherently designed to increase accessibility. | Desktop applications, on the other hand, have better authorization and administrators have better control, hence more secure. | | Web applications rely heavily on internet connectivity, for their operation. | Desktop applications don’t require the internet for their operations. Some applications just require internet connectivity at the time of update. |  1. Any 2 differences | |
|  | 1. PDPA- any 2 Obligations   1. Consent Obligation - Only collect, use or disclose personal data when an individual has given his/her consent.  2.Purpose Limitation Obligation - An organisation may collect, use or disclose personal data about an individual for the purposes that a reasonable person would consider appropriate in the circumstances and for which the individual has given consent.  3.Notification Obligation - Notify individuals of the purposes for which your organisation is intending to collect, use or disclose their personal data on or before such collection, use or disclosure of personal data.  4. Access and Correction Obligation - Upon request, the personal data of an individual and information about the ways in which his or her personal data may have been used or disclosed in the past year should be provided. Organisations are also required to correct any error or omission in an individual’s personal data upon his or her request.  5. Accuracy Obligation - Make reasonable effort to ensure that personal data collected by or on behalf of your organisation is accurate and complete, if it is likely to be used to make a decision that affects the individual, or if it is likely to be disclosed to another organisation.  6. Protection Obligation - Make security arrangements to protect the personal data that your organisation possesses or controls to prevent unauthorised access, collection, use, disclosure or similar risks.  7. Retention Limitation Obligation - Cease retention of personal data or remove the means by which the personal data can be associated with particular individuals when it is no longer necessary for any business or legal purpose.  8. Transfer Limitation Obligation -Transfer personal data to another country only according to the requirements prescribed under the regulations, to ensure that the standard of protection provided to the personal data so transferred will be comparable to the protection under the PDPA.  9. Accountability Obligation -Make information about your data protection policies, practices and complaints process available on request. | |