**HCI Computing 2019 Prelim Paper 2 Solution Guide**

|  |  |
| --- | --- |
| 1 | (a)   * Modules can be kept in a library and re-used in other solutions * Many programmers can work on same problem as each can be given difference modules to solve * Easier to debug as modules are small * Easier to maintain and modify as modules can be removed/added easily   (b)    (c)  Critical Task: A, B, D, F, I  (d)  Slack Time for C: 1  Slack Time for G: 2  (e)  Although it would cause Task E to start 1 day later because Task E is dependent on Task C, it has no significant impact on the completion date of the project because Task C is not on the critical path, and has 1 day of slack it could use.  (f)   * Put more people to work on it * Work more hours in a day * Increase the efficiency of the work eg automating a manual process   (g)  Reducing Tasks D and F each by 1 day would only shorten the project by 1 day since after a 2 day reduction, tasks D and F are no longer on the critical path. To further shorten the project time, the project manager would have to shorten tasks on the new critical path A,B,C,E,I.  (h)    (i)  Gantt chart shows more clearly, visually, the total time scheduled for a project and its relative scale as it contains a fixed grid of evenly spaced chronological timed cells |

|  |  |
| --- | --- |
| 2 | (a)  Processes Involved:  **Process Enquiry/Availability**   * Input from customer entity (bookTtitles, qty) * Input from Stock data store (bookID, bookTitle, price, qtyOnHand) * Output to customer entity (availability)   **Process Order**   * Input from Process Enquiry/Availability (availability) * Input from customer entity (bookTtitles, qty, name, address, credit card details) * Output to customer entity (total payable) * Output to Stock data store (*update* qtyOnHand) * Output to Cust data store(custID, custName, address, credit card details) * Output to Order data store (orderID, custID, bookID, qty)   **Generate Packing List**   * Input from Cust data store (custName, address, credit card details) * Input from Order data store (orderId, custID, bookID, qty) * Input from Stock data store (bookId, bookTitle) * Output to Warehouse personnel (packing list)   (b)   * Tables in the database can be based on the files/data stores in the data flow diagram. * Processes identified in the DFD can be implemented by use of queries or macros. The processes might involve the use of forms. * Letters output from the system will be reports in the database solution * Data identified on the data flows will be some of the fields required to be defined when the database tables are created.   (c)   * Although fields may be identified, there is nothing on the DFD about the type of data or its possible values or validation that may be carried out on it; all of which are necessary for defining fields when creating database tables. * There is also no help as to the layout of forms and reports. |

|  |  |
| --- | --- |
| 3 | (a)  Asynchronous transmission sends only one byte of data at a time. Synchronous transmission combines data into longer frames that may contain multiple bytes. Asynchronous transmission uses start bit and stop bit where synchronous does not.    (b)  Simplex mode is one direction. Full-duplex mode is two directions at the same time. Half-duplex mode only allows two directions but only one direction at a time.  (c)  Packet switching provides a faster and less congestion data transformation. |

|  |  |
| --- | --- |
| 4 | (a)  data validation:  format check/range check for the dates of return tickets  length check for passport number, handphone number  data verification:  summary page of details which requires customer to check and confirm  (b)  Client scripting is used to make the webpages more interactive after they have been sent to the client’s web browser. Example: JavaScript, ActionScript, VBScript  Server scripting is used to create HTML pages on the web server that are sent to the browser. Example: JavaScript, PHP, ASP.NET, Perl, Ruby, Python  (c)  DOS (Denial of Service). The attacker may send large amounts of request to the server which makes it unable to attend to the legitimate requests. Strategies include intrusion detection system, increase bandwidth.  Virus attack. Virus can be implanted in the computer and thus alter or delete the data stored. Strategies include anti-virus software, firewall.  (d)  Advantage: easy to grow or shrink based on demand. Instant accessibility for staff regardless of locations.  Disadvantage: costly and possible leak of data  (e)  Staff must keep themselves updated with new technologies to protect the digital security of the company’s resources. Staff must protect the customers’ data and not to use their access rights to modify any data. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | (a)   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Is Book Overdue > 5 days | N | N | N | N | Y | Y | Y | Y | | Has Previous Warning Letter | N | N | Y | Y | N | N | Y | Y | | No. of books overdue > 4 | N | Y | N | Y | N | Y | N | Y | | Send reminder letter | X | X | X | X | X |  |  |  | | Send warning letter to student only |  |  |  |  |  | X |  |  | | Send warning letter to student and parents |  |  |  |  |  |  | X | X |   (b)   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Is Book Overdue > 5 days | N | Y | Y | Y | | Has Warning Letter | - | N | N | Y | | No. of books overdue > 4 | - | N | Y | - | | Send reminder letter | X | X |  |  | | Send warning letter |  |  | X |  | | Send warning letter to student and parents |  |  |  | X | |

|  |  |
| --- | --- |
| 6 | (a)  A flat file   * all the records are held in one table or unrelated tables   A relational database   * two or more tables are linked/related to each other   (b)  State the following problems:   * Repeated fields in multiple records * As a result of the repetition, there is a possibility of inconsistency * Customer without current subscription, will not have the record of the customer * If no subscriptions for magazine, there will not be records on the magazine   (c)  The possible relations are:   * Customer //To keep track of all customers who have ever make a subscription with the company. * Magazine //To keep track of all the magazines the company has to offer for subscriptions. * Subscription //To track all subscriptions * Category // To record all the different category of magazines   (d)  State the following improvements:   * All magazine will be recorded regardless if there is any subscriptions currently * All customers will be recorded even if they do not have any subscriptions currently * No repetition of customer info over multiple records * No repetition of magazine info over multiple records   (e)  Solutions should include all the tables mentioned in part c. Beside the ER diagram, a brief description is needed for each set of relationships.   * Each customer subscribes to zero or many magazines * Each magazine is subscribe by zero or more customers * Each customer has zero or many subscriptions * Each subscription is own by a customer * Each Magazine has none or many subscriptions * Each subscription is for 1 magazine.   C:\Users\tanck\Desktop\ER.jpg |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | (a)    X 2 A \* B + =    A \* B + =  2 2\*A 2\*A+B  X X X X=2\*A+B   * Scan the expression form left to right * When an operand is encountered, push it onto the stack * When an operator is encountered, pop the last two operands and perform the operation, * the result is then pushed onto the stack   (b)   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Tree[9] |  |  |  |  | | Tree[8] | A | -1 | -1 |  | | Tree[7] | \* | 5 | 8 |  | | Tree[6] |  |  |  |  | | Tree[5] | 2 | -1 | -1 |  | | Tree[4] | + | 7 | 1 |  | | Tree[3] | X | -1 | -1 |  | | Tree[2] |  |  |  |  | | Tree[1] | B | -1 | -1 |  | | Tree[0] | = | 3 | 4 |  |   (c)  **=**  **Y --**  **\* ^**  **2 + A 2**  **A B**    (d)  Y 2 A B + \* A 2 ^ -- = |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | A[0] | A[1] | A[2] | A[3] | A[4] | A[5] |  | | 435 | 646 | 344 | 54 | 23 | 98 | Original list | | **344** |  | **435** |  |  |  | low = 0, high = 5, pivot = A[2] =344 | |  | **98** |  |  |  | **646** | left = 1, right = 5, swap 646, 98 | |  |  | **23** |  | **435** |  | left = 2, right = 4, swap 435,23 | |  |  |  |  |  |  | left = 4, right = 3, pos = 3 | | 54 | 98 | 23 | 344 | 435 | 646 | Swap 344, 54. Partition Done. | | 54 | 98 | 23 |  |  |  | **Left sublist** | | **98** | **54** |  |  |  |  | low = 0, high = 2, pivot = A[1] = 98 | |  |  |  |  |  |  | left = 3, right = 2, pos = 2 | | 23 | 54 | 98 |  |  |  | Swap 98, 23. Partition Done. | | 23 | 54 |  |  |  |  | **Left sublist** | |  |  |  |  |  |  | 2-element sublist requires no partition. Compare 23 & 54, no swap required | |  |  |  |  | 435 | 646 | **Right sublist** | |  |  |  |  |  |  | 2-element sublist requires no partition. Compare 435 & 646, no swap required | | 23 | 54 | 98 | 344 | 435 | 646 | Sorting is done. | |