**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **28th may 2020** | **Name:** | **Rashmitha** |
| **Course:** | **Logic design** | **USN:** | **4AL17EC077** |
| **Topic:** | **Boolean equation for digital circuits, MUX and decoders to logic gates,7 segment decoder** | **Semester & Section:** | **6th sem ‘B’ sec** |
| **Github Repository:** | **Rashmitha** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session**  C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (173).png  C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (174).png  **BOOLEAN EQUATIONS FOR DIGITAL CIRCUITS :**    Boolean Algebra is an algebra, which deals with binary numbers & binary variables. Hence, it is also called as Binary Algebra A mathematician, named George Boole had developed this algebra in 1854. The variables used in this variables.  The range of voltages corresponding to Logic ‘High’ is represented with ‘1’ and the range of voltages corresponding to logic with ‘0’.  Postulates and Basic Laws of Boolean Algebra In this section, let us discuss about the Boolean postulates and basic laws that are used in Boolean algebra. These are useful in m Boolean functions.  Boolean Postulates Consider the binary numbers 0 and 1, Boolean variable xx and its complement x′x′. Either the Boolean variable or complement of it is known as literal. The four possible logical OR operations among these literals and binary numbers are shown below.  x + 0 = x  x + 1 = 1  x + x = x  x + x’ = 1  Similarly, the four possible logical AND operations among those literals and binary numbers are shown below.  x.1 = x  x.0 = 0  x.x = x  x.x’ = 0  These are the simple Boolean postulates. We can verify these postulates easily, by substituting the Boolean variable with ‘0’  Note− The complement of complement of any Boolean variable is equal to the variable itself. i.e., x′x′’=x.  Basic Laws of Boolean Algebra Following are the three basic laws of Boolean Algebra.  Commutative law Associative law Distributive law Commutative Law If any logical operation of two Boolean variables give the same result irrespective of the order of those two variables, then that logical o  is said to be Commutative. The logical OR & logical AND operations of two Boolean variables x & y are shown below  x + y = y + x  x.y = y.x  The symbol ‘+’ indicates logical OR operation. Similarly, the symbol ‘.’ indicates logical AND operation and it is optional t Commutative law obeys for logical OR & logical AND operations.  Associative Law If a logical operation of any two Boolean variables is performed first and then the same operation is performed with the remaining variable give the same result, then that logical operation is said to be Associative. The logical OR & logical AND operations of three Bool z are shown below.  x + y+zy+z = x+yx+y + z  x.y.zy.z = x.yx.y.z  Associative law obeys for logical OR & logical AND operations.  Distributive Law If any logical operation can be distributed to all the terms present in the Boolean function, then that logical operation is said to be Distributive. The distribution of logical OR & logical AND operations of three Boolean variables x, y & z are shown below.  x.y+zy+z = x.y + x.z  x + y.zy.z = x+yx+y.x+zx+z Distributive law obeys for logical OR and logical AND operations.  These are the Basic laws of Boolean algebra. We can verify these laws easily, by substituting the Boolean variables with ‘0’.  **DESIGN OF 7 SEGMENT DECODER WITH COMMON ANODE DISPLAY:**  BCD to 7-Segment Display Decoders A binary coded decimal (BCD) to 7-segment display decoder such as the TTL 74LS47 or  74LS48, have 4 BCD inputs and 7 output lines, one for each LED segment. This allows a smaller 4-bit binary number (half a byte) to be used to display all the denary numbers from 0 to 9 and by adding two displays together, a full range of numbers from 00 to 99 can be displayed with just a single byte of eight data bits.  The use of packed BCD allows two BCD digits to be stored within a single byte (8-bits) of data, allowing a single data byte to hold a BCD number in the range of 00 to 99. An example of the 4-bit BCD input ( 0100 ) representing the number “4” is given below.  **Display Decoder Example :**  In practice current limiting resistors of about 150Ω to 220Ω would be connected in series between the decoder/driver chip and each LED display segment to limit the maximum current flow. There are different display decoders and drivers available for the different types of available displays, either LED or LCD. For example, the 74LS48 for commoncathode LED types, the 74LS47 for common-anode LED types, or the CMOS CD4543 for liquid crystal display (LCD) types.Liquid crystal displays (LCD´s) have one major advantage over similar LED types in that they consume much less power and nowadays, both LCD and LED displays are combined together to form larger Dot-Matrix Alphanumeric type displays which can show letters and characters as well as numbers in standard Red or Tri-colour outputs.  It can be seen that to display any single digit number from 0 to 9 in binary or letters from A to F in hexadecimal, we would require seven separate segment connections plus one additional connection for the LED’s “common” connection. Also as the segments are basically a standard light emitting diode, the driving circuit would need to produce up to 20mA of current to illuminate each individual segment and to display the number “8”, all seven segments would need to b lit resulting a total current of nearly 140mA, (8 x 20mA).  Obviously, the use of so many connections and power consumption is impractical for some electronic or microprocessor based circuits and so in order to reduce the number of signal lines required to drive just one single display, display decoders such as the BCD to 7-Segment Display Decoder and Driver IC’s are used instead. |
|  |

**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **28th may 2020** | **Name:** | **Rashmitha** |
| **Course:** | **Python** | **USN:** | **4AL17EC077** |
| **Topic:** | **Build a desktop database application** | **Semester & Section:** | **6th sem ‘B’ sec** |
| **Github Repository:** | **Rashmitha** |  |  |

|  |
| --- |
| **AFTERNOON SESSION DETAILS** |
| **Image of session**  C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (175).png  C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (176).png  **CREATE DATABASE :**    A Database is defined as a structured set of data. So, in SQL the very first step to store the data in a well structured manner is to create a database. The CREATE DATABASE statement is used to create a new database in SQL.  Syntax:  CREATE DATABASE database\_name;  database\_name: name of the database.  Example Query:  This query will create a new database in SQL and name the database as my\_database.  CREATE DATABASE my\_database;  CREATE TABLE  We have learned above about creating databases. Now to store the data we need a table to do that. The CREATE TABLE statement is used to create a table in SQL. We know that a table comprises of rows and columns. So while creating tables we have to provide all the information to SQL about the names of the columns, type of data to be stored in columns, size of the data etc. Let us now dive into details on how to use CREATE TABLE statement to create tables in SQL.  Syntax:  CREATE TABLE table\_name  (  column1 data\_type(size),  column2 data\_type(size),  column3 data\_type(size),  ....  );  table\_name: name of the table.  column1 name of the first column.  data\_type: Type of data we want to store in the particular column.  For example,int for integer data.  size: Size of the data we can store in a particular column.  For example if for a column we specify the data\_type as int and size as 10 the this column can store an integer  number of maximum 10 digits.  Example Query:  This query will create a table named Students with three columns, ROLL\_NO, NAME and SUBJECT.  CREATE TABLE Students  (  ROLL\_NO int(3),  NAME varchar(20),  SUBJECT varchar(20),  );  This query will create a table named Students. The ROLL\_NO field is of type int and can store an integer number of size 3. The next two columns NAME and SUBJECT are of type varchar and can store characters and the size 20 specifies that these two fields can hold maximum of 20 characters. |
|  |