**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **29/5/2020** | | **Name:** | **Abhishek Vasudev Mahendrakar** | | |
| **Course:** | **Logic Design** | | **USN:** | **4AL17EC003** | | |
| **Topic:** | **1. Analysis of clocked sequential circuits**  **2. Digital clock design** | | **Semester & Section:** | **6th-‘A’** | | |
| **Github Repository:** | **ECEAbhishekVMahendrakar** | | **E-mail:** | **abhi2244mahendrakar@gmail.com** | | |
| **FORENOON SESSION DETAILS** | | | | | |
| **Image of session** | | | | | |
| **Report – Report can be typed or hand written for up to two pages.**  **The main parts of the circuit are as follows:**   1. **Timer 555: Responsible for generating the clock pulses for the counters, the frequency of the output shoul be 1 hz which means 1 second for each pulse.** 2. **Counters: Responsible for generating the time in BCD (Binary Coded decimal).** 3. **Decoders : Takes the BCD of the counter as input and produces 7 segment output .** 4. **7 segments : Displays the time, of course!**  * **Seconds have 2 displays , 2 decoders and 2 counters. The same for minutes and hours.The circuit works as follows :** * **555 timer produces 1 seconde pulses to the clock input of the first counter which is responsible the first column of seconds, so its output will change every second.** * **The counter produces numbers from 0 to 9 in BCD form and automatically resets to 0 after that, so the output of the first counter will count from 0 to 9 every second and that's exactly what we want from it, so we are done here. let's move to the next one.**   **What do we want here?**  **First, we want the 2nd counter to start counting when the 1st one moves for 9 to 0 (that makes 10 seconds!)**  **How can this be done?**  **Let's check the output of the fist counter in BCD :**  **MSB---LSB**  **0: 0000**  **1: 0001**  **2: 0010**  **3: 0011**  **4: 0100**  **5: 0101**  **6: 0110**  **7: 0111**  **8: 1000**  **9: 1001**  **0: 0000**   * **Remember that 7490 decade counters respond only to the pulses that go from 1 to 0 and notice that this case only happens in the BCD code above when the output changes from 9 to 0 (the Most significant bit changes from 1 to 0). So, we'll just connect the clock input of the 2nd counter to the most significant bit of the output of the first counter.** * **Second, Since we have 60 seconds in the minute we want the 2nd counter to count only to 5, that makes 59 seconds maximum, when it take another pulse it doesn't count to 60, instead it resets itself to 0 and send a pulse to the first counter in minutes to tell it to count 1 minute**   **How can this be done?**   * **From the BCD code above (6: 0110) when the output is 6 the two middle bits are 1 (Q1,Q2),** * **So By ANDing these two bits the output will be 1, This output will be connected to the reset pin of the same counter (2nd one) and the clock input of the next one(3rd).** * **When the output is 6 the AND gate output (1) will reset the same counter and its outputs goes 0000 so the output of the and gate again goes to 0 (1---->0), that will clock the next counter. Beautiful!**   **##Notice that the output of the counters are named : Q0 , Q1 , Q2 , Q3 ##**   * **The 4th counter will be the same as the second one so we are clocking it using the Most Significant Bit of the output of the previous one.** * **Again, the 5th counter is the same as the 3rd one and takes its clock from the AND gate.** * **The 5th and the 6th counters are responsible for hours so they are limited to 23, and resets themselves to 00 when the 5th counter is 4 and the last one is 2 (24).** * **This is done using and gate with Q2 (3rd bit) of the 5th counter as one input and Q1 (second bit) of the last counter as the other input, and the output of this AND gate will be connected to both resets of the last 2 counters.** * **When the last counter is 0(0000) or 1(0001), Q1 which is one of the inputs to the AND gate will be 0 so the output of the AND gate will be zero. when it counts to 2 this bit will be 1 so the output of the and gate will depend on the the other input which is Q2 of the previous counter, and this bit will be zero until it reaches 4 (0100),So, the output of the and gate will be 1 (0--->1) resetting both counters to 00.** * **The output of these counters are converted to 7 segment output using 7447 decoders, then to the 7 segment, we won't get into the details of their datasheets.** * **In the next part we're going to explain how to modify this design to make it only 12 hours mode (the problem will be that when we reset the hours it has to go to 01 not 00) and how to design both modes in the same ciruit with transferring between them using a simple switch...** | | | | | |
| **Date:** | **29/5/2020** | **Name:** | | | **Abhishek Vasudev Mahendrakar** |
| **Course:** | **UDEMY-The Python Mega Course: Build 10 real world applications** | **USN:** | | | **4AL17EC003** |
| **Topic:** | 1. **Object Oriented Programming** | **Semester & Section:** | | | **6th-‘A’** |
| **AFTERNOON SESSION DETAILS** | | | | | |
| **Image of session** | | | | | |
| **Report – Report can be typed or hand written for up to two pages.**  **Object-oriented programming(OOP):**   * Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data, in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods). * A feature of objects is an object's procedures that can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). * In OOP, computer programs are designed by making them out of objects that interact with one another. * OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types. * Object oriented programming (OOP) is a programming structure where programs are organized around objects as opposed to action and logic. * This is essentially a design philosophy that uses a different set of programming languages such as C#. * Understanding OOP concepts can help make decisions about how you should design an application and what language to use.   **CODE:**  **Here are the frontend.py and backend.py scripts in OOP style. To execute this program you should execute the frontend.py file.**  #frontend.py  from tkinter import \*  from backend import Database    database=Database("books.db")    class Window(object):    def \_\_init\_\_(self,window):    self.window = window    self.window.wm\_title("BookStore")    l1=Label(window,text="Title")  l1.grid(row=0,column=0)    l2=Label(window,text="Author")  l2.grid(row=0,column=2)    l3=Label(window,text="Year")  l3.grid(row=1,column=0)    l4=Label(window,text="ISBN")  l4.grid(row=1,column=2)    self.title\_text=StringVar()  self.e1=Entry(window,textvariable=self.title\_text)  self.e1.grid(row=0,column=1)    self.author\_text=StringVar()  self.e2=Entry(window,textvariable=self.author\_text)  self.e2.grid(row=0,column=3)    self.year\_text=StringVar()  self.e3=Entry(window,textvariable=self.year\_text)  self.e3.grid(row=1,column=1)    self.isbn\_text=StringVar()  self.e4=Entry(window,textvariable=self.isbn\_text)  self.e4.grid(row=1,column=3)    self.list1=Listbox(window, height=6,width=35)  self.list1.grid(row=2,column=0,rowspan=6,columnspan=2)    sb1=Scrollbar(window)  sb1.grid(row=2,column=2,rowspan=6)    self.list1.configure(yscrollcommand=sb1.set)  sb1.configure(command=self.list1.yview)    self.list1.bind('<<ListboxSelect>>',self.get\_selected\_row)    b1=Button(window,text="View all", width=12,command=self.view\_command)  b1.grid(row=2,column=3)    b2=Button(window,text="Search entry", width=12,command=self.search\_command)  b2.grid(row=3,column=3)    b3=Button(window,text="Add entry", width=12,command=self.add\_command)  b3.grid(row=4,column=3)    b4=Button(window,text="Update selected", width=12,command=self.update\_command)  b4.grid(row=5,column=3)    b5=Button(window,text="Delete selected", width=12,command=self.delete\_command)  b5.grid(row=6,column=3)    b6=Button(window,text="Close", width=12,command=window.destroy)  b6.grid(row=7,column=3)    def get\_selected\_row(self,event):  index=self.list1.curselection()[0]  self.selected\_tuple=self.list1.get(index)  self.e1.delete(0,END)  self.e1.insert(END,self.selected\_tuple[1])  self.e2.delete(0,END)  self.e2.insert(END,self.selected\_tuple[2])  self.e3.delete(0,END)  self.e3.insert(END,self.selected\_tuple[3])  self.e4.delete(0,END)  self.e4.insert(END,self.selected\_tuple[4])    def view\_command(self):  self.list1.delete(0,END)  for row in database.view():  self.list1.insert(END,row)    def search\_command(self):  self.list1.delete(0,END)  for row in database.search(self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get()):  self.list1.insert(END,row)    def add\_command(self):  database.insert(self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get())  self.list1.delete(0,END)  self.list1.insert(END,(self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get()))    def delete\_command(self):  database.delete(self.selected\_tuple[0])    def update\_command(self):  database.update(self.selected\_tuple[0],self.title\_text.get(),self.author\_text.get(),self.year\_text.get(),self.isbn\_text.get())    window=Tk()  Window(window)  window.mainloop()  **Below you will also find the backend.py script in OOP:**  #backend.py  import sqlite3  class Database:  def \_\_init\_\_(self, db):  self.conn=sqlite3.connect(db)  self.cur=self.conn.cursor()  self.cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER PRIMARY KEY, title text, author text, year integer, isbn integer)")  self.conn.commit()  def insert(self,title,author,year,isbn):  self.cur.execute("INSERT INTO book VALUES (NULL,?,?,?,?)",(title,author,year,isbn))  self.conn.commit()  def view(self):  self.cur.execute("SELECT \* FROM book")  rows=self.cur.fetchall()  return rows  def search(self,title="",author="",year="",isbn=""):  self.cur.execute("SELECT \* FROM book WHERE title=? OR author=? OR year=? OR isbn=?", (title,author,year,isbn))  rows=self.cur.fetchall()  return rows  def delete(self,id):  self.cur.execute("DELETE FROM book WHERE id=?",(id,))  self.conn.commit()  def update(self,id,title,author,year,isbn):  self.cur.execute("UPDATE book SET title=?, author=?, year=?, isbn=? WHERE id=?",(title,author,year,isbn,id))  self.conn.commit()  def \_\_del\_\_(self):  self.conn.close()  **Bonous Section:** | | | | | |