ASSIGNMENT-7[ADITYA ANAND\_21108048]

**#1**

from tkinter import \*

def findGst() :

org\_cost= int(org\_priceField.get())

N\_price = int(net\_priceField.get())

gst\_rate = ((N\_price - org\_cost) \* 100) / org\_cost;

gst\_rateField.insert(10, str(gst\_rate) + " % ")

def clearAll():

org\_priceField.delete(0, END)

net\_priceField.delete(0, END)

gst\_rateField.delete(0, END)

if \_name\_ == "\_main\_" :

gui = Tk()

gui.configure(background = "light green")

gui.title("GST Rate Finder")

gui.geometry("300x300")

org\_price = Label(gui, text = "Original Price",

bg = "blue")

net\_price = Label(gui, text = "Net Price",

bg = "blue")

find = Button(gui, text = "Find", fg = "Black",

bg = "Red",

command = findGst)

gst\_rate = Label(gui, text = "Gst Rate", bg = "blue")

clear = Button(gui, text = "Clear", fg = "Black",

bg = "Red",

command = clearAll)

org\_price.grid(row = 1, column = 1,padx = 10,pady = 10)

net\_price.grid(row = 2, column = 1, padx = 10, pady = 10)

find.grid(row = 3, column = 2,padx = 10,pady = 10)

gst\_rate.grid(row = 4, column = 1,padx = 10, pady = 10)

clear.grid(row = 5, column = 2, padx = 10, pady = 10)

org\_priceField = Entry(gui)

net\_priceField = Entry(gui)

gst\_rateField = Entry(gui)

org\_priceField.grid(row = 1, column = 2 ,padx = 10,pady = 10)

net\_priceField.grid(row = 2, column = 2, padx = 10,pady = 10)

gst\_rateField.grid(row = 4, column = 2, padx = 10,pady = 10)

gui.mainloop()

**#2**

def partition(l, r, nums):

pivot, ptr = nums[r], l

for i in range(l, r):

if nums[i] <= pivot:

nums[i], nums[ptr] = nums[ptr], nums[i]

ptr += 1

nums[ptr], nums[r] = nums[r], nums[ptr]

return ptr

def quicksort(l, r, nums):

if len(nums) == 1:

return nums

if l < r:

pi = partition(l, r, nums)

quicksort(l, pi-1, nums)

quicksort(pi+1, r, nums)

return nums

example = [4, 5, 1, 2, 3]

result = [1, 2, 3, 4, 5]

print(quicksort(0, len(example)-1, example))

example = [2, 5, 6, 1, 4, 6, 2, 4, 7, 8]

result = [1, 2, 2, 4, 4, 5, 6, 6, 7, 8]

print(quicksort(0, len(example)-1, example))

**#3**

from tkinter import \*

import calendar

def showCal() :

new\_gui = Tk()

new\_gui.config(background = "white")

new\_gui.title("CALENDAR")

new\_gui.geometry("550x600")

fetch\_year = int(year\_field.get())

cal\_content = calendar.calendar(fetch\_year)

cal\_year = Label(new\_gui, text = cal\_content, font = "Consolas 10 bold")

cal\_year.grid(row = 5, column = 1, padx = 20)

new\_gui.mainloop()

if \_name\_ == "\_main\_" :

gui = Tk()

gui.config(background = "white")

gui.title("CALENDAR")

gui.geometry("250x140")

cal = Label(gui, text = "CALENDAR", bg = "dark gray",

font = ("times", 28, 'bold'))

year = Label(gui, text = "Enter Year", bg = "light green")

year\_field = Entry(gui)

Show = Button(gui, text = "Show Calendar", fg = "Black",

bg = "Red", command = showCal)

Exit = Button(gui, text = "Exit", fg = "Black", bg = "Red", command = exit)

cal.grid(row = 1, column = 1)

year.grid(row = 2, column = 1)

year\_field.grid(row = 3, column = 1)

Show.grid(row = 4, column = 1)

Exit.grid(row = 6, column = 1)

gui.mainloop()

**#4**

from tkinter import \*

expression = ""

def press(num):

global expression

expression = expression + str(num)

equation.set(expression)

def equalpress():

try:

global expression

total = str(eval(expression))

equation.set(total)

expression = ""

except:

equation.set(" error ")

expression = ""

def clear():

global expression

expression = ""

equation.set("")

if \_name\_ == "\_main\_":

gui = Tk()

gui.configure(background="light green")

gui.title("Simple Calculator")

gui.geometry("270x150")

equation = StringVar()

expression\_field = Entry(gui, textvariable=equation)

expression\_field.grid(columnspan=4, ipadx=70)

button1 = Button(gui, text=' 1 ', fg='black', bg='red',

command=lambda: press(1), height=1, width=7)

button1.grid(row=2, column=0)

button2 = Button(gui, text=' 2 ', fg='black', bg='red',

command=lambda: press(2), height=1, width=7)

button2.grid(row=2, column=1)

button3 = Button(gui, text=' 3 ', fg='black', bg='red',

command=lambda: press(3), height=1, width=7)

button3.grid(row=2, column=2)

button4 = Button(gui, text=' 4 ', fg='black', bg='red',

command=lambda: press(4), height=1, width=7)

button4.grid(row=3, column=0)

button5 = Button(gui, text=' 5 ', fg='black', bg='red',

command=lambda: press(5), height=1, width=7)

button5.grid(row=3, column=1)

button6 = Button(gui, text=' 6 ', fg='black', bg='red',

command=lambda: press(6), height=1, width=7)

button6.grid(row=3, column=2)

button7 = Button(gui, text=' 7 ', fg='black', bg='red',

command=lambda: press(7), height=1, width=7)

button7.grid(row=4, column=0)

button8 = Button(gui, text=' 8 ', fg='black', bg='red',

command=lambda: press(8), height=1, width=7)

button8.grid(row=4, column=1)

button9 = Button(gui, text=' 9 ', fg='black', bg='red',

command=lambda: press(9), height=1, width=7)

button9.grid(row=4, column=2)

button0 = Button(gui, text=' 0 ', fg='black', bg='red',

command=lambda: press(0), height=1, width=7)

button0.grid(row=5, column=0)

plus = Button(gui, text=' + ', fg='black', bg='red',

command=lambda: press("+"), height=1, width=7)

plus.grid(row=2, column=3)

minus = Button(gui, text=' - ', fg='black', bg='red',

command=lambda: press("-"), height=1, width=7)

minus.grid(row=3, column=3)

multiply = Button(gui, text=' \* ', fg='black', bg='red',

command=lambda: press("\*"), height=1, width=7)

multiply.grid(row=4, column=3)

divide = Button(gui, text=' / ', fg='black', bg='red',

command=lambda: press("/"), height=1, width=7)

divide.grid(row=5, column=3)

equal = Button(gui, text=' = ', fg='black', bg='red',

command=equalpress, height=1, width=7)

equal.grid(row=5, column=2)

clear = Button(gui, text='Clear', fg='black', bg='red',

command=clear, height=1, width=7)

clear.grid(row=5, column='1')

Decimal= Button(gui, text='.', fg='black', bg='red',

command=lambda: press('.'), height=1, width=7)

Decimal.grid(row=6, column=0)

# start the GUI

gui.mainloop()

**#5**

def heapify(nums, heap\_size, root\_index):

largest = root\_index

left\_child = (2 \* root\_index) + 1

right\_child = (2 \* root\_index) + 2

if left\_child < heap\_size and nums[left\_child] > nums[largest]:

largest = left\_child

if right\_child < heap\_size and nums[right\_child] > nums[largest]:

largest = right\_child

if largest != root\_index:

nums[root\_index], nums[largest] = nums[largest], nums[root\_index]

heapify(nums, heap\_size, largest)

def heap\_sort(nums):

n = len(nums)

for i in range(n, -1, -1):

heapify(nums, n, i)

for i in range(n - 1, 0, -1):

nums[i], nums[0] = nums[0], nums[i]

heapify(nums, i, 0)

random\_list\_of\_nums = [35, 12, 43, 8, 51]

heap\_sort(random\_list\_of\_nums)

print(random\_list\_of\_nums)

**#6**

def Remove(duplicate):

final\_list = []

for num in duplicate:

if num not in final\_list:

final\_list.append(num)

return final\_list

duplicate = [2, 4, 10, 20, 5, 2, 20, 4]

print(Remove(duplicate))