```
#question1
# import all functions/classes from the tkinter
from tkinter import *


# Function for finding GST rate
def findGst() :


        # take a value from the respective entry boxes
        # get method returns current text as string
        org_cost= int(org_priceField.get())


        N_price = int(net_priceField.get())


        # calculate GST rate
        gst_rate = ((N_price - org_cost) * 100) / org_cost;


        # insert method inserting the
        # value in the text entry box.
        gst_rateField.insert(10, str(gst_rate) + " % ")




# Function for clearing the
# contents of all text entry boxes
def clearAll():
        # deleting the content from the entry box
        org_priceField.delete(0, END)

        net_priceField.delete(0, END)

        gst_rateField.delete(0, END)


# Driver Code
if __name__ == "__main__" :

        # Create a GUI window
        gui = Tk()

        # Set the background colour of GUI window
        gui.configure(background = "light green")

        # set the name of tkinter GUI window
        gui.title("GST Rate Finder")

        # Set the configuration of GUI window
        gui.geometry("300x300")

        # Create a Original Price: label
        org_price = Label(gui, text = "Original Price",
                                    bg = "blue")
```

```
# Create a Net Price : label
net_price = Label(gui, text = "Net Price",
                                  bg = "blue")

# Create a Find Button and attached to
# findGst function
find = Button(gui, text = "Find", fg = "Black",
                          bg = "Red",
                          command = findGst)

# Create a Gst Rate : label
gst_rate = Label(gui, text = "Gst Rate", bg = "blue")

# Create a Clear Button and attached to
# clearAll function
clear = Button(gui, text = "Clear", fg = "Black",
                          bg = "Red",
                          command = clearAll)

# grid method is used for placing
# the widgets at respective positions
# in table like structure .

# padx attributed provide x-axis margin
# from the root window to the widget.
# pady attributed provide y-axis
# margin from the widget.
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)

# Create a text entry box for filling or typing the information.
org_priceField = Entry(gui)

net_priceField = Entry(gui)

gst_rateField = Entry(gui)

# grid method is used for placing
# the widgets at respective positions
# in table like structure .
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)


        # Start the GUI

        gui.mainloop()










        #Question2

        # import all methods and classes from the tkinter

from tkinter import *


# import calendar module

import calendar


# Function for showing the calendar of the given year

def showCal() :


        # Create a GUI window

        new_gui = Tk()


        # Set the background colour of GUI window

        new_gui.config(background = "white")
```

```
# set the name of tkinter GUI window
new_gui.title("CALENDAR")

# Set the configuration of GUI window
new_gui.geometry("550x600")

# get method returns current text as string
fetch_year = int(year_field.get())

# calendar method of calendar module return
# the calendar of the given year .
cal_content = calendar.calendar(fetch_year)

# Create a label for showing the content of the calendar
cal_year = Label(new_gui, text = cal_content, font = "Consolas 10 bold")

# grid method is used for placing
# the widgets at respective positions
# in table like structure.
cal_year.grid(row = 5, column = 1, padx = 20)

# start the GUI
new_gui.mainloop()
```

```
# Driver Code

if __name__ == "__main__" :


        # Create a GUI window

        gui = Tk()


        # Set the background colour of GUI window

        gui.config(background = "white")


        # set the name of tkinter GUI window

        gui.title("CALENDAR")


        # Set the configuration of GUI window

        gui.geometry("250x140")


        # Create a CALENDAR : label with specified font and size

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

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


        # Create a Enter Year : label

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


        # Create a text entry box for filling or typing the information.

        year_field = Entry(gui)


        # Create a Show Calendar Button and attached to showCal function
```

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

                                                bg = "Red", command = showCal)


# Create a Exit Button and attached to exit function

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


# grid method is used for placing

# the widgets at respective positions

# in table like structure.

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)


# start the GUI

gui.mainloop()



#Question3


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()
```

```
    #Question4

    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))

#Question 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)

# Question6


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))
```