1. The following program creates a graphical user interface allowing the user to enter a sequence of numbers and calculate either the largest value or the sum. Assume that the functions `maxList` and `sumList` each take a string as a parameter and return the appropriate value. For example, `maxList("2 5 1 4")` returns `5`.

```
from Tkinter import *                #1

root = Tk()                                                  #2

f = Frame( root )                                            #3
f.pack()                                                     #4

def calculate():                                             #5
    s = e.get()                      #6
    r = v.get()                      #7
    if r==1:                         #8
        result = maxList(s)          #9
    else:                            #10
        result = sumList(s)          #11
    l.configure( text = str(result) )                #12
    l.update()                                               #13

e = Entry( f )                                               #14
e.pack()                                                     #15

l = Label( f, text="" )                                      #16
l.pack()                                                     #17

v = IntVar()                         #18

r1 = Radiobutton( f, text = "Max", variable = v,    #19
                  value = 1, command = calculate)   #20
r1.pack()                            #21

r2 = Radiobutton( f, text = "Sum", variable = v,    #22
                  value = 2, command = calculate)   #23
r2.pack()                            #24

q = Button( f, text = "Quit",                                #25
            command = root.destroy )#26
q.pack()                                                     #27

root.mainloop()                      #28
```

(a) Explain the purpose of each line or group of lines in the program, and describe exactly how the user should interact with it. Ignore any errors that may occur due to inappropriate input.

[20]

(b) The question does not say what `maxList` and `sumList` do if their input is inappropriate. What would you like them to do, so that the user can see an error message in this case? You do not need to define `maxList` and `sumList`, but if your solution also requires modifications to `calculate` then you should define the new version of `calculate`.

[5]

1a

Line 1 makes the definitions in the Tkinter module available for use; it provides GUI functionality.
Lines 2 – 4 set up a window and prepare it for other GUI elements to be added.
Lines 14 – 15 create a text entry field and add it to the main window.
Lines 16 – 17 create a text label (an area in which messages can be displayed) and add it to the main window.
Line 18 creates an integer variable object, required for use with the radio buttons.
Lines 19 – 20 create a radio button with label "Max", which when selected will store the value 1 in the variable v and call the function `calculate`, and adds it to the main window.
Lines 22 – 24 similarly create a second radio button with label "Sum", which when selected will store the value 2 in the variable v and call the function `calculate`.
Lines 25 – 27 create a button with label "Quit", which when pressed will call the `destroy` method of the `root` window, terminating the user interface.
Line 28 activates the user interface by calling the `mainloop` method of the `root` window, so that mouse and keyboard activity will be monitored in relation to the user interface elements.
Lines 5 – 13 define the function `calculate` which is called when either of the radio buttons is selected. This function obtains a string from the text entry field (line 6), and the value of the integer variable indicating which radio button was pressed (line 7). It uses either `maxList` or `sumList`, as appropriate, to calculate `result` (lines 8 – 11). Finally it converts result to a string and displays it in the label field of the user interface (lines 12 – 13).

The program creates a window with a text entry field, a label on which text can be displayed, and two radio buttons labelled "Max" and "Sum". The user can type a sequence of numbers into the text entry field and press either "Max" or "Sum". The result of the selected calculation is displayed on the label.

1b

The simplest solution is that the functions should return the string `"Error"` if their input is unsuitable, because this string will be displayed directly in the window by the existing version of `calculate`. Other solutions, for example using exceptions, are possible if `calculate` is also modified appropriately.

2. Consider the following Tkinter program:

```
from string import *;  from Tkinter import *
root = Tk(); f = Frame( root ); f.pack()

c = Canvas( f, height = 100, width = 200,
            background = "white" )
c.grid( columnspan=2 )

x = 100;  y = 50
i = 3
p = c.create_oval( x-i,y-i,x+i,y+i )
c.update()

def nextAction():
    global x, y
    s = e.get()
    r = v.get()
    bits = split( s )
    dx = int( bits[ 0 ] );  dy = int( bits[ 1 ] )
    nx = x+dx;  ny = y+dy
    if r==1:
        c.create_line( x,y,nx,ny )
        c.update()
    l.configure( text = str( nx ) + " " + str( ny ) )
    l.update()
    x = nx
    y = ny
    c.move( p,dx,dy )
    c.update()

e = Entry( f )
e.grid()

l = Label( f, text= str( x ) + " " + str( y ) )
l.grid( row=1, column=1 )

v = IntVar()

r1 = Radiobutton( f, text = "This", variable = v, value = 1)
r1.select()
r1.grid( row=2 )

r2 = Radiobutton( f, text = "That", variable = v, value = 2)
r2.grid( row=2, column=1 )

d = Button( f, text = "Do Something", command = nextAction )
d.grid( row=3)

q = Button( f, text = "Quit", command = root.destroy )
q.grid( row=3, column=1)

root.mainloop()
```

(a) Draw the graphical user interface created by the program. You do not need to be precise on the sizes of the items, but you should ensure that the positioning of items relative to one another is correct. [6]

(b) Explain, ***in only a few sentences***, the operation of the program e.g. what is displayed as the user interacts with the various components of the interface. ***DO NOT*** explain exactly what each separate line does. [6]

(c) What will happen if the user types in inappropriate input? How would you adjust the code to improve the user interface in this respect? (You do not need to write the precise code for this.) [3]

2a

The window contains a canvas drawing error across the top, twice as wide as it is high, with a very small circle right in the centre. Below this are two columns of widgets, in three rows. The first row contains a text entry area and a label area. The entry box is initially blank, the label area contains *100 50*. The second row contains a pair of radio buttons, labeled *This* and *That*; the latter is selected. The third row contains two buttons, labeled *Do Something* and *Quit*. All items are centred in their columns.

Only the general configuration need be shown. Absolutely precise positioning is not required.

2b

The program allows line drawings consisting of straight lines to be created on the canvas area. Initially, the drawing point is in the middle of the canvas area, shown with a small circle – and this position is displayed in the label to the right of the entry area. The user types in a pair of integers into the entry area, separated by a space. When the user clicks on *Do Something*, if *This* is selected, then a line is drawn from the current drawing position to a second position offset from the first by the two values in the entry box; if *That* is selected, the drawing position is moved to the second position, but no line is drawn. The offsets can be negative. The new position is displayed in the label field, irrespective of which radio button is selected, and the small circle is moved to show the position on the canvas.

2c

An error is raised when the *Do Something* button is pressed, but there is no report to the user via the graphical interface. Wrap the code in nextAction, from where the integer conversion is attempted to the end of the function, in a try…except… In the except case, an error message could be written to either the entry field or the label field. Below is code for the entry field (but they need not provide this):

```
def nextAction():
    global x, y
    s = e.get()
    r = v.get()
    bits = split( s )
    try:
        dx = int( bits[ 0 ] )
        dy = int( bits[ 1 ] )
        nx = x+dx
        ny = y+dy
        if r==1:
            c.create_line( x,y,nx,ny )
            c.update()
        l.configure( text = str( nx ) + " " + str( ny ) )
        l.update()
        x = nx
        y = ny
        c.move( p,dx,dy )
        c.update()
    except:
        e.delete( 0,END )
        e.insert( 0,"bad input" )
        e.update()
```

3. Consider the following Tkinter program:

```
from Tkinter import *

root = Tk() ; f = Frame( root ); f.pack()

c = Canvas( f, height=100, width=200, background = "white" )
c.grid( rowspan=2,columnspan=3 )

p = c.create_oval( 97,47,103,53 ); c.update()

def nextAction():
    s = int( e.get() )
    r = v.get()
    dx = 0; dy = 0
    if r == 1:
        dy = -s
    elif r == 2:
        dx = -s
    elif r == 3:
        dx = s
    else:
        dy = s
    c.move( p,dx,dy )
    c.update()

e = Entry( f, width=2 ); e.grid(row=1,column=3)
e.insert( 0,"1" )

l = Label( f, text="Step" ); l.grid( row=0, column=3 )

v = IntVar()

r1 = Radiobutton( f, text = "A", variable = v, value = 1)
r1.grid( row=2, column=1 ); r1.select()

r2 = Radiobutton( f, text = "B", variable = v, value = 2)
r2.grid( row=3, column=0 )

r3 = Radiobutton( f, text = "C", variable = v, value = 3)
r3.grid( row=3, column=2 )

r4 = Radiobutton( f, text = "D", variable = v, value = 4)
r4.grid( row=4, column=1 )

d = Button( f, text = "A1", command = nextAction )
d.grid( row=3, column=3)

q = Button( f, text = "A2", command = root.destroy )
q.grid( row=4, column=3)

root.mainloop()
```

(a) Draw the graphical user interface created by the program. You do not need to be precise on the sizes of the items, but you should ensure that the positioning of items relative to one another is correct. [5]

(b) Explain, ***in only a few sentences***, the operation of the program e.g. what is displayed as the user interacts with the various components of the interface. ***DO NOT*** explain exactly what each separate line does. [5]

(c) Is it possible to crash the program? If so, explain how, and explain accurately how you would go about fixing the problem, writing out the adjusted code if necessary.

3a

The window contains Tkinter objects on a 4 across by 5 down grid. It contains a canvas drawing area across the top, twice as wide as it is high taking 3 by 2 of the grid in the top left. The canvas contains a very small circle right in the centre. In the final column adjacent to the canvas on the right are a label "Step", and a text entry widget, initially containing "1", one above the other. Below this are four columns of widgets, in three rows. The left most 3 by 3 cells contain radio buttons labeled A, B, C, and D, at the North, West, East and South positions respectively. The other cells in the 3 by 3 area are empty. The North radio button should be selected. The lowest two cells in the fourth column contain two buttons labeled A1, A2. ALL items are centred in their columns.

Only the general configuration need be shown. Absolutely precise positioning is not required.

3b

The program allows the circle in the middle of the canvas to be moved, as follows. Each time button A1 is pressed, the circle is moved in the direction currently indicated by the radio buttons – if A is selected, the circle moves up, B, to the left, C, to the right, and D, downwards. The circle is moved by the number of pixels indicated in the entry widget under the label 'Step'. The A2 button quits the program.

3c

An error is raised when the *A1* button is pressed and there is not a well-formed integer in the entry box. No report to the user via the graphical interface. Wrap the code in nextAction, from where the integer conversion is attempted to the end of the function, in a try…except… In the except case, an error message could be written to either the entry field or the label field. Below is code for the entry field (but they need not provide this):

```
def nextAction():
    try:
        s = int( e.get() )
        r = v.get()
        dx = 0; dy = 0
        if r == 1:
            dy = -s
        elif r == 2:
            dx = -s
        elif r == 3:
            dx = s
        else:
            dy = s
        c.move( p,dx,dy )
        c.update()
    except:
        e.delete( 0,END )
        e.insert( 0,"Err" )
        e.update()
```

4. Consider the following Tkinter program:

```
from Tkinter import *

root = Tk()
f = Frame( root )
f.pack()

c = Canvas( f, height = 100, width = 200,
            background = "white",
            bd = 5, relief = RIDGE )
c.grid( rowspan = 2 )

p1 = 100
p2 = 50

def action( e ):
    global p1, p2
    if e.num == 1:
        if v.get() == 1:
            r1.deselect()
            r2.select()
        else:
            c.create_line( p1, p2, e.x, e.y )
        p1 = e.x
        p2 = e.y
    elif e.num == 2:
        s = d.get()                                          # 1
        c.create_text( e.x, e.y, text = s )

    c.update()

c.bind( "<Button>",action )                                  # 2

d = Entry( f, width=20 )
d.grid(row=2)
d.insert( 0,"message" )

v = IntVar()

r1 = Radiobutton( f, text = "Move", variable = v, value = 1)
r1.grid( row=0, column=1);

r2 = Radiobutton( f, text = "Join", variable = v, value = 0)
r2.grid( row=1, column=1); r2.select()

x = Button( f, text = "A2", command = root.destroy )
x.grid( row=2, column=1)                                     # 3

root.mainloop()
```

(a) Explain precisely the action of the lines identified with comments 1, 2, and 3. [5]

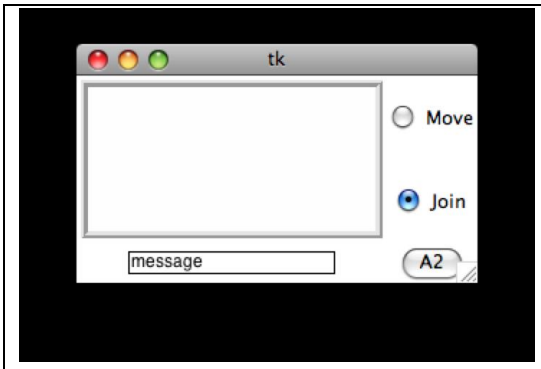
(b) Draw the graphical user interface created by the program. You do not need to be precise on the sizes of the items, but you will lose marks if you do not ensure that the positioning of items relative to one another is correct – look at the layout commands carefully. [6]

(c) Explain, ***in only a few sentences***, the operation of the program e.g. what is displayed as the user interacts with the various components of the interface. ***DO NOT*** explain exactly what each separate line does. [5]

4a

1: Gets the text that is currently in the text entry box 'd', assigning it to variable 's'.
2: This sets an action handler for the Canvas 'c'. The action that activates this handler is any button being pressed down on the mouse. The activity that is executed as a result of the action is the 'action' function.
3: This is a layout instruction, directing the button widget 'x' to be placed in the row 2 (ie third row) and column 1 (ie second column) of the grid inside Frame 'f'.

4b



The window contains Tkinter objects on a 2 across by 3 down grid. It contains a canvas drawing area across the top, twice as wide as it is high taking 2 down by 1 across of the grid in the top left. The canvas is initially empty. In the final column adjacent to the canvas on the right are two radio buttons, one above the other, labeled Move and Join respectively, in row positions 0 and 1 respectively. The Join button should be selected. Below the canvas in the lower left position (0,2), is a text entry box, initially holding the text "message". In the lower right position (1,2), below the radio buttons, there is a button with the text A2. All items are centred in their columns.

Only the general configuration need be shown. Absolutely precise positioning is not required.

3c

The program allows the user to draw lines and text in the canvas. The program remembers the last position clicked on in the canvas with mouse button 1 – initially this is set to 100, 50. With Join selected, each time mouse button 1 is clicked, a line is drawn from the last position clicked to the current position. If Move is selected, the position is moved without drawing a line, next time button 1 is clicked – this automatically resets the radio buttons to Join. If mouse button 2 is clicked, then the text in the text box is drawn onto the canvas, centred at the position clicked. The A2 button quits the program.