Software Development Tools and Methods LAB4 REPORT

By

AMAKOR Augustina Chidinma and KARIMI Soufiane

November 8th

Contents

1 Syntax of SWIG's input file

1. The following are the differences between Part A and those in Part B:

   a) Part A which contains the `%{#include myheader.h %}` block simply copies the myheader.h file, verbatim into the resulting wrapper file created by SWIG. While the Part B contains the ANSI C variable and function declarations (the global variable, function) of the header file.

2 Elementary Types and Functions

1. We created a file tp4.i for swig that permits to access the functionalities of tp4 library declared in the first part of tp4.h file with the code below:

```
%module tp4

%{
#include "tp4.h"
%}
#define VERSION "1.0.3"

extern const double PI;

/* return PI value */
double pi();

/* return PI+v */
double add_pi(double v);

/* log function */
void set_log(int v);

/* Some stats */
void stats();
```

2. Explanation on the line-commands in the creation of the library `_tp4`.

   - `swig -python tp4.i` :- Builds a python module tp4.py

   - `gcc -O2 -fPIC -c tp4.c` :- Generates a shared C source file `tp4_wrap.c`

   - `gcc -O2 -fPIC -c tp4_wrap.c -I/usr/include/python3.8/` :- Links the shared `tp4_wrap.c` file to the `python3.8` library.

   - `gcc -shared tp4.o tp4_wrap.o -o _tp4.so` :- For linking the module `tp4` to the corresponding output file (object file) `_tp4.so`

3. Yes we were able to import the tp4 in python

4. Below is the content of our updated `CMakeLists.txt` file:

```
add_library(tp4 SHARED tp4.c)

set_target_properties( tp4  PROPERTIES
          VERSION 1.3.4
          SOVERSION 1
          CFLAGS -Wall
        )

  # This is a CMake example for Python
  FIND_PACKAGE(SWIG REQUIRED)
  INCLUDE(${SWIG_USE_FILE})

  FIND_PACKAGE(PythonLibs)
  INCLUDE_DIRECTORIES(${PYTHON_INCLUDE_PATH})

  INCLUDE_DIRECTORIES(${CMAKE_CURRENT_SOURCE_DIR})

  SET(CMAKE_SWIG_FLAGS "")

  SET_SOURCE_FILES_PROPERTIES(tp4.i PROPERTIES SWIG_FLAGS "-includeall")
  SWIG_ADD_MODULE(tp4 python tp4.i tp4.c)
  SWIG_LINK_LIBRARIES(tp4 ${PYTHON_LIBRARIES})
```

5. Python program `test1.py`

Listing 1: test1.py

```
import tp4

def print_part1():
        print("VERSION=%s\n", tp4.VERSION)
        print("PI=%lg\n", tp4.PI)
        print("pi()=%lg\n", tp4.pi())
        print("PI+5=%lg\n", tp4.add_pi(5))

def call():
        tp4.stats()
        print_part1()
        tp4.set_log(1)
        print_part1()
        tp4.set_log(0)
        tp4.stats()
call()
```

Listing 2: output

```
Number of created vectors : 0
Number of destroyed vectors : 0
VERSION=%s
 1.0.3
PI=%lg
 3.14
pi()=%lg
 3.14
PI+5=%lg

 8.14
VERSION=%s
 1.0.3
PI=%lg
 3.14
        LOG: Invocation of pi()
pi()=%lg
 3.14
        LOG: Invocation of add_pi(5)
        LOG: Invocation of pi()
PI+5=%lg
 8.14
        LOG: Invocation of set_log(0)
Number of created vectors : 0
Number of destroyed vectors : 0
```

3 Structures, pointers and objects

We created a python code which do the same as the tp.c code, below the following code :

Listing 3: Python example

```
# global variables
PI=3.14;
nb_created=0
nb_destroyed=0

import gc


# Struct of Vector
class Vector():
    def __init__(self):
        # Initialise
        self.valid = 100
        self.cs = [100,100,1000]


# Functions
def pi():
    print("Invocation of pi()\n")
    return PI

def add_pi(v):
    print("Invocation of add_pi(%lg)\n", v)
    return pi()+v

def stats():
    print("Number of created vectors : %i\n", nb_created)
    print("Number of destroyed vectors : %i\n", nb_destroyed)


def alloc():
    v = Vector()
    v.valid = 0xBEEF
    nb_created=+1
    return v

def desalloc(v):
    v.valid=0
    #nb_destroyed+=1
    del v
    gc.collect()

def check(v):
    assert(v.valid == 0xBEEF)

def Vector_create(a,b,c):
    print("Vector_create\n")
    v=alloc()
    check(v)
    v.cs[0]=a
    v.cs[1]=b
    v.cs[2]=c
    return v

def Vector_add(v1,v2):
    print("Vector_add\n");
    v=alloc()
    check(v)
    check(v1)
    check(v2)
    for i in range(3):
        v.cs[i]=v1.cs[i]+v2.cs[i]
    return v

def Vector_elem(v,coord):
    print("Vector_elem\n")
    check(v)
    return v.cs[coord]

def Vector_str(v):
    print("Vector_str\n")
    check(v)
    print(v.cs[0], v.cs[1], v.cs[2])

def Vector_destroy(v):
    print("Vector_destroy\n")
    check(v)
    desalloc(v)
```

4 GUI

We first created a window class that represents a Vector which will be described in two rows ie; first row is a text(a string of number that identifies the Vector) and, the Second rows contains three cells(columns), each cell is takes in an entry of the vector component.

Listing 4: Window Class Vector

```
import tkinter as tk
from tkinter import *
class Vector:
    def __init__(self, master,number):
        self.label = tk.Label(master,text=str(number))
        self.e1 = Entry(master)
        self.e2 = Entry(master)
        self.e3 = Entry(master)
    def grid(self,nb_row):
        self.label.grid(row=nb_row)
        self.e1.grid(row=nb_row+1, column=0)
        self.e2.grid(row=nb_row+1, column=1)
        self.e3.grid(row=nb_row+1, column=2)
    def destroy(self):
        self.label.destroy()
        self.e1.destroy()
        self.e2.destroy()
        self.e3.destroy()
    def get(self):
        return float(self.e1.get()),float(self.e2.get()),
        float(self.e3.get())
    def insert(self,a,b,c):
        self.e1.insert(0,a)
        self.e2.insert(0,b)
        self.e3.insert(0,c)
```

We then created functions that will manipulate the buttons actions : (Creating a Vector, Deleting a Vector, Showing the vector in the terminal, adding two vectors)

Listing 5: Python example

```
all_vectors = []
def createVector():
    print("Create Vector ")
    next_vector = len(all_vectors)
    # add entry in second row
    vec = Vector(frame_for_boxes,next_vector)
    vec.grid(2*next_vector+1)
    all_vectors.append(vec)
def deleteVector():
    print(’delete Vector’)
    all_vectors[-1].destroy()
    all_vectors.remove(all_vectors[-1])
def showVector():
    showing_vector = EntryShow.get()
    print(" Vector of index :  \n")
    print(showing_vector)
    print(all_vectors[int(showing_vector)].get())
def add_vectors():
    print(’Add two Vectors’)
    sum_vector1 = EntrySum1.get()
    sum_vector2 = EntrySum2.get()
    sum1_a,sum1_b,sum1_c = all_vectors[int(sum_vector1)].get()
    sum2_a,sum2_b,sum2_c = all_vectors[int(sum_vector2)].get()
    sum_a = sum1_a + sum2_a
    sum_b = sum1_b + sum2_b
    sum_c = sum1_c + sum2_c
    createVector()
    all_vectors[-1].insert(str(sum_a),str(sum_b),str(sum_c))
```

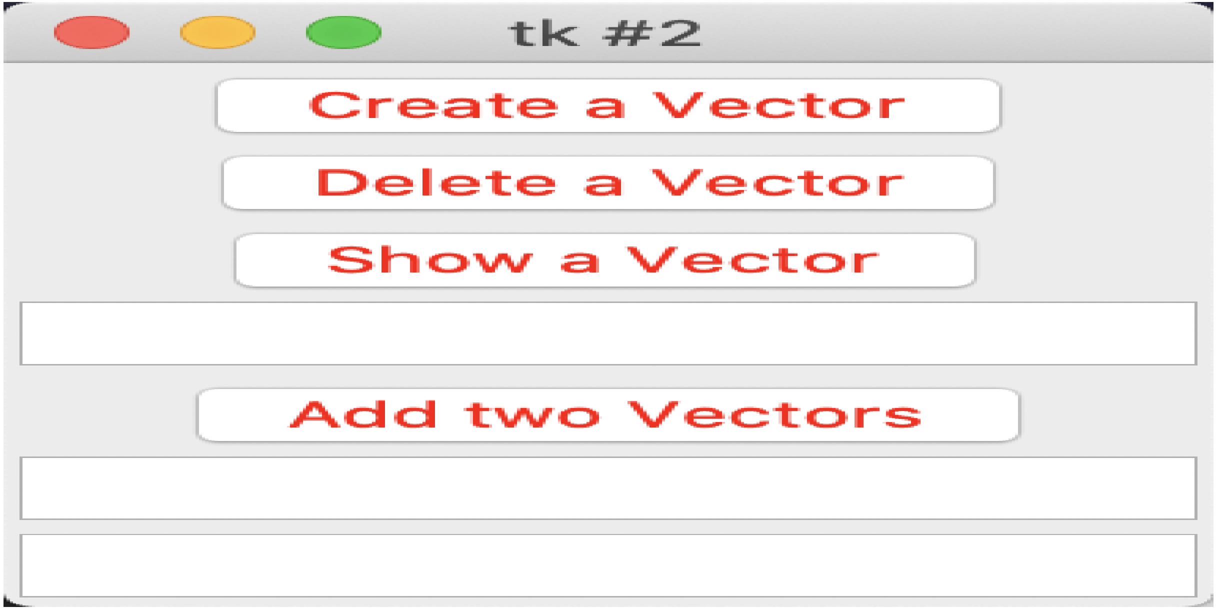
Finally we created the main buttons and entries : a button which creates a vector, a button which deletes the vector, a button which show the vector named "showing Entry", a button which creates a vector "Summing Entries" by summing two previously created vectors using their string number name as identifiers.

Listing 6: Python example

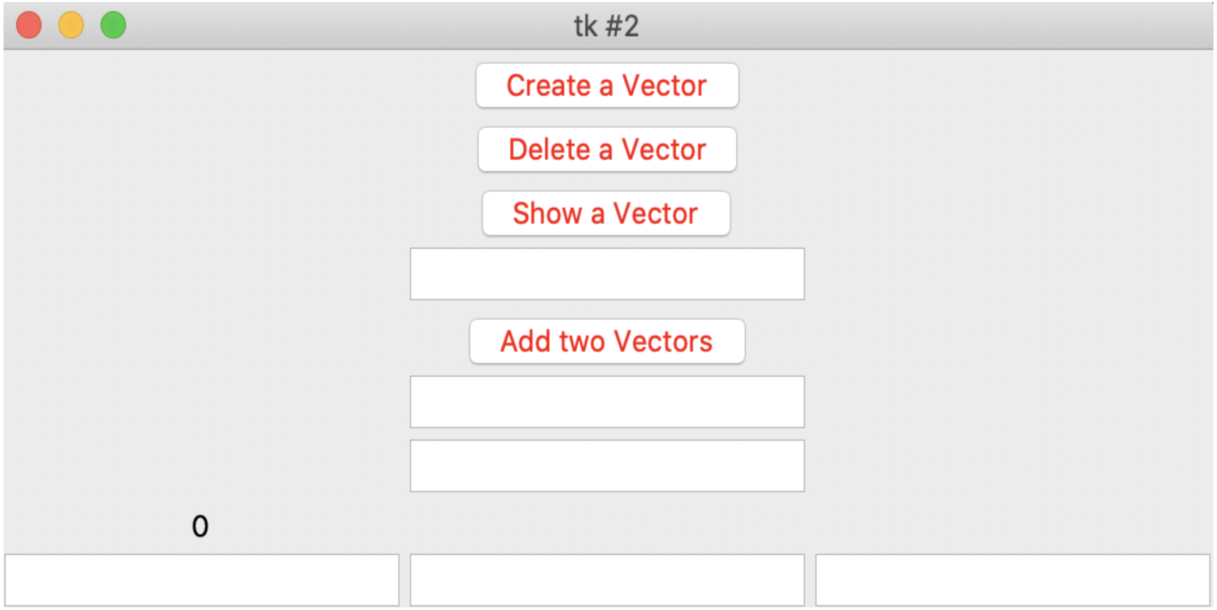
```
root = tk.Tk()
addboxButton = Button(root, text=’Create a Vector’,
fg="Red", command=createVector)
addboxButton.pack()
aboxButton = Button(root, text=’Delete a Vector’,
fg="Red", command=deleteVector)
aboxButton.pack()
ShowButton = Button(root, text=’Show a Vector’,
fg="Red", command=showVector)
ShowButton.pack()
EntryShow = Entry(root)
EntryShow.pack()
AddButton = Button(root, text=’Add two Vectors’,
fg="Red", command=add_vectors)
AddButton.pack()
EntrySum1 = Entry(root)
EntrySum1.pack()
EntrySum2 = Entry(root)
EntrySum2.pack()
frame_for_boxes = Frame(root)
frame_for_boxes.pack()

root.mainloop()
```

5 Pictorial outputs of our tkinter code

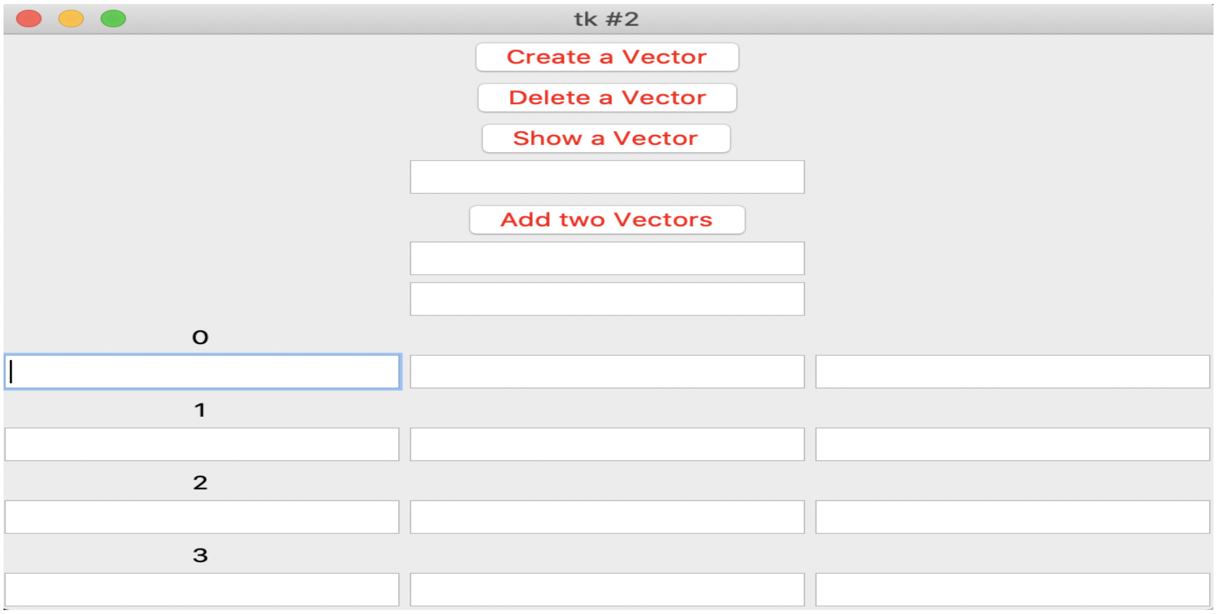
Below the main buttons and Entries :



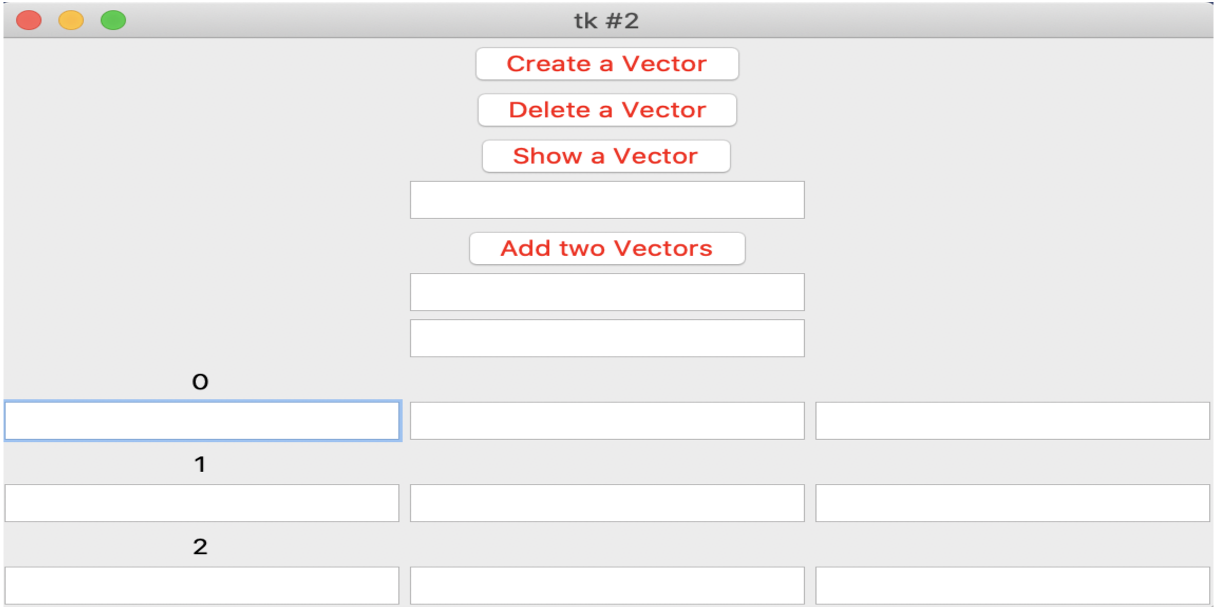
When clicking then on Create a Vector, We create a new vector represented by the number and the three entries as below



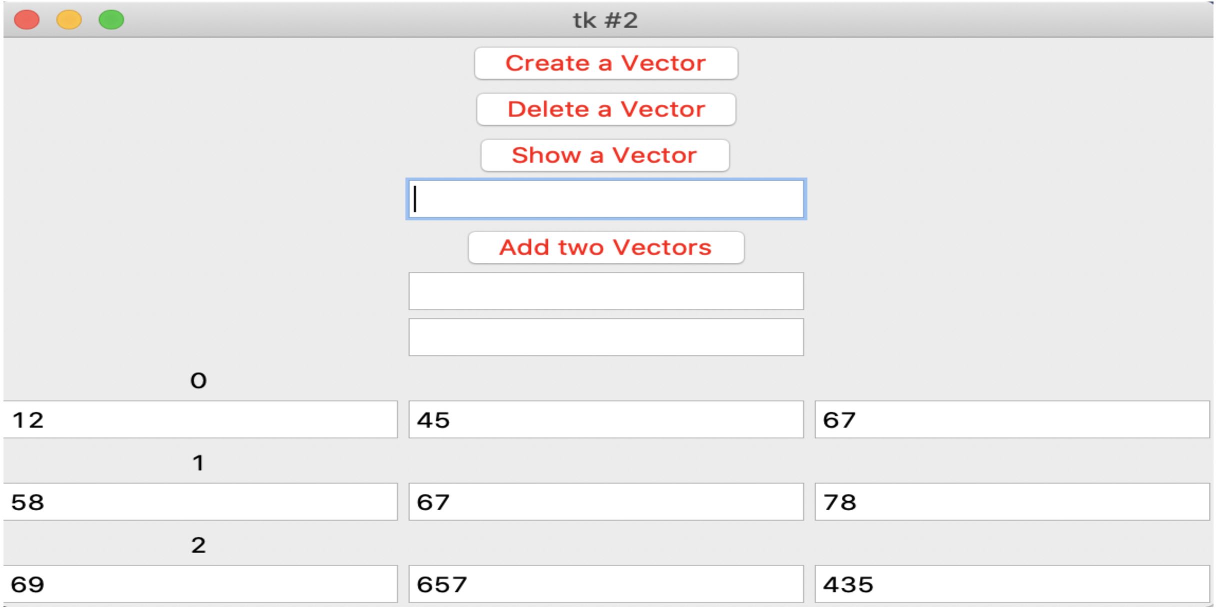
Here, For example we create Four Vectors as below



By clicking on Delete a vector, We destroy the last created vector as belox



We fill then our three Vectors as below



0

12 45 67

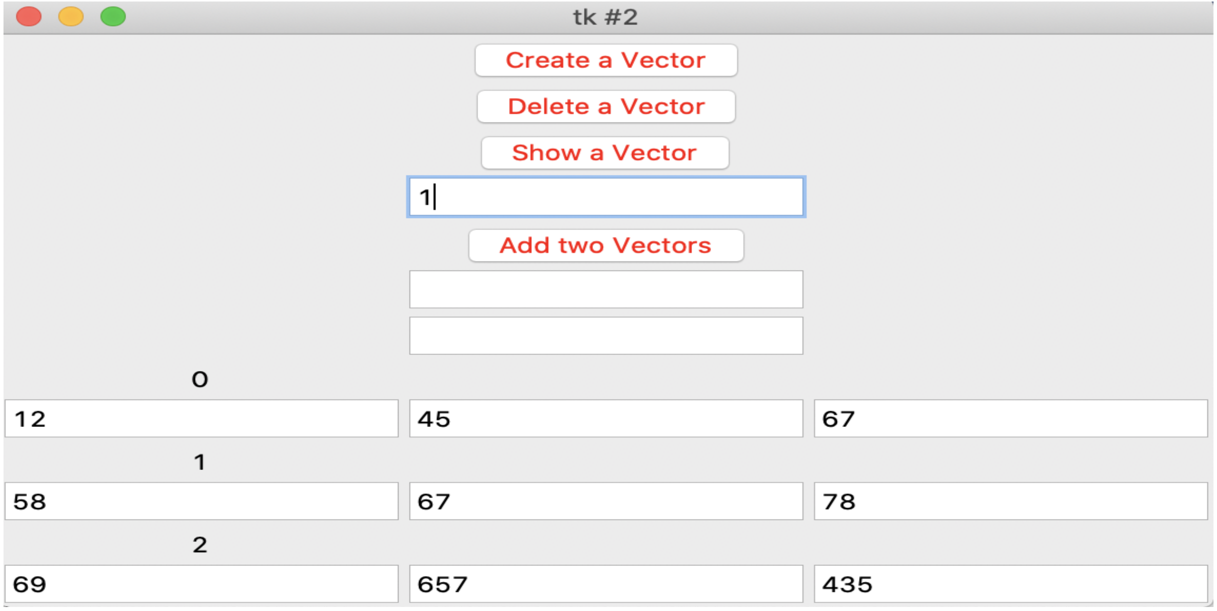
1

58 67 78

2

69 657 435

Then we select the Vector 1 as below



1

0

12 45 67

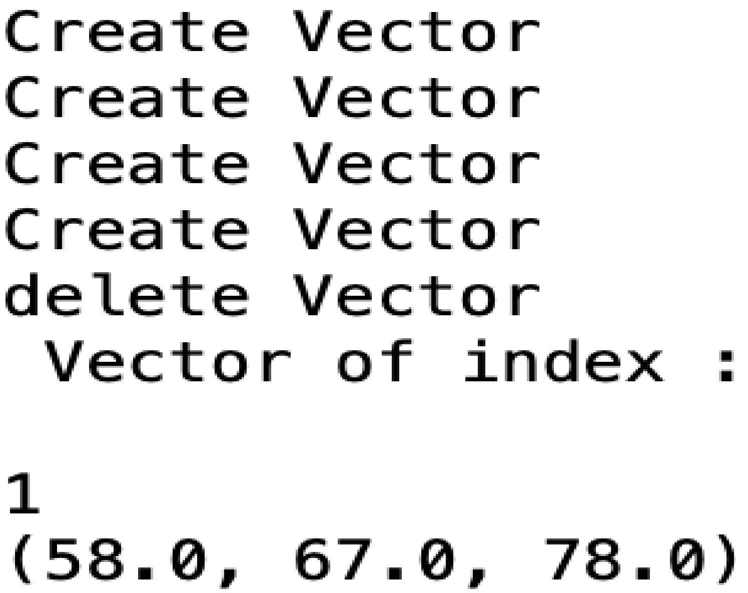
1

58 67 78

2

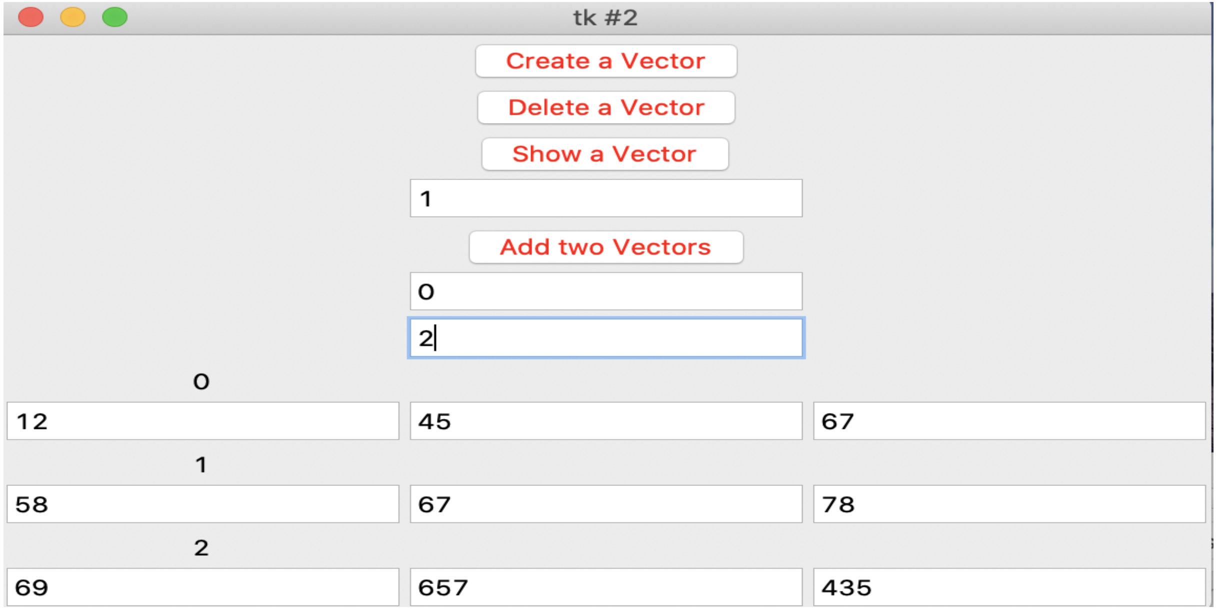
69 657 435

By clicking on show vector, we get the following output on the terminal



```
Create Vector
Create Vector
Create Vector
Create Vector
delete Vector
 Vector of index :

1
(58.0, 67.0, 78.0)
```

We Select the vectors 0 and 2 to sum them



1

0

2

0

12	45	67

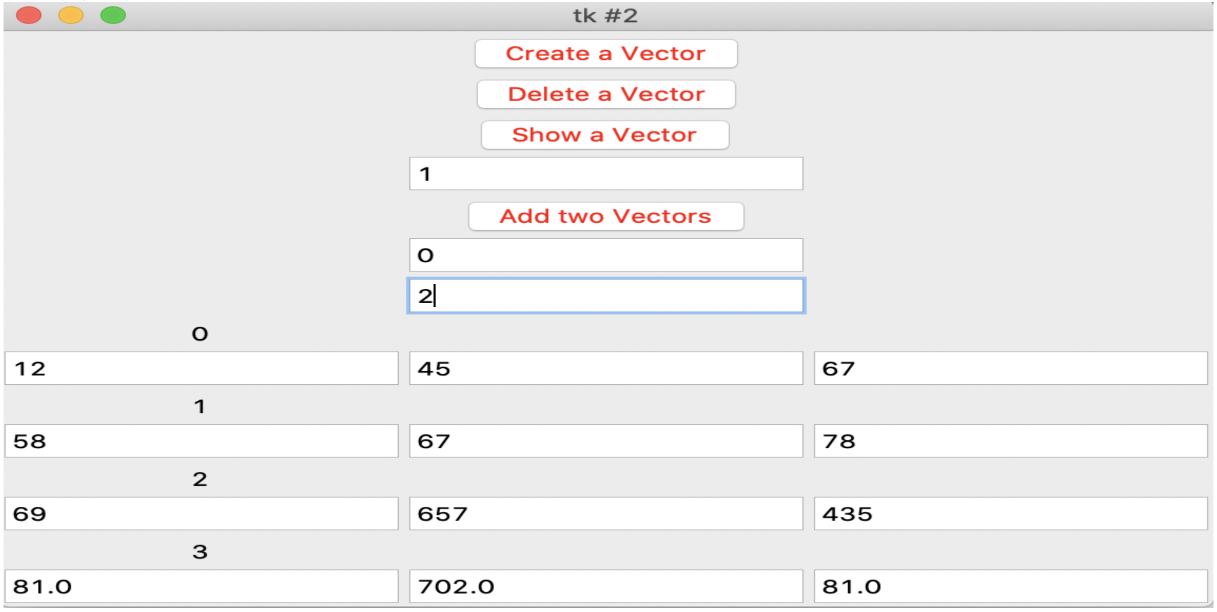
1

58	67	78

2

69	657	435

By clicking on Add two vectors, we create a new vector with entries filled by the sum as below



1

0

2

0	12	45	67
1	58	67	78
2	69	657	435
3	81.0	702.0	81.0

6 Reference

- Site http://www.swig.org/Doc1.3/Python.html
- Site https://www.tutorialspoint.com/python/python_gui_programming.htm