* **Author/Authors:**

This project was done by Saffa Gbondo and Marima Andrew Mambondiumwe

**Driver- Saffa Gbondo**

**Navigator- Marima Andrew Mambondiumwe**

* **Motivation**:

We both enjoyed doing the Animal guessing game, which was A17 and we thought it would be nice to modify it and make it more visual, Saffa and I worked together for our final project inCSC226 and we used the Tkinter library and thought it was fun, so we decided to implement The Animal guessing game using Tkinter. We however decided to use a stack instead of a tree just to make it different and explore something different.

* **Purpose**:

The program makes use of the stack data structure as well as implementing the GUI Tkinter module, which allows it to be interactive with the user. It shows a visual representation of how the stack data structure works. The animal guessing game was originally designed to use trees but it could be also used using stacks., so the purpose of this program is to show how stacks work whilst having fun using animals at the same time.

* **Audience**:

The program is interactive and has visuals so it is suitable for any audience, it helps the users gain an understanding of how the stack data structure works at the same time have fun seeing “pushing” and “popping” of items in the stack take place.

* **Instructions:** There is a text input button that the user can use to enter a name of an animal. The program gives the user a list of animals to choose from, if you select an animal it then appends the animal to the stack, as it does it, it shows the stack and an image of the selected animal. The user can add as many animals as they want but only from the specified list, if an animal that is not on the list is entered, the program will display a message saying there is not such an animal from the list. The user can also at any time “peek” at the list and see what is currently on the stack. The user can also “pop” or remove the animals from the stack, after every pop, a new updated stack is displayed and an image of the animal next in line to be removed is displayed. The order of removal of a stack is L.I.F.O(Last in First Out).
* **Design**: Using CRC cards, pseudocode, and/or a mixture, create a design plan which meets the computational requirements.

|  |
| --- |
| from Tkinter import \*  Class stack():"""this opens a window and allows the user to input in animals from given options and the animals will then be appended to a stack"""  def \_\_init\_\_(self, master):  """initializes the values to be used in the class and acts as a constructor"""    def append\_list(self):  """this method adds the inputted animals to the stack and also displays an image that corresponds to the animal chosen"""    def isEmpty(self):  """this method returns an empty list"""    def push(self, item):  """this allow the items to be stacked in order of which they are added"""    def pop(self):  """this allows items to be removed from the stack"""    def popping(self):  """items are removed from the stack, one after the other from the last one that was added"""  def peek(self):  """allows you to see the items that are already in the stack"""  def size(self):  """returns the length of the stack""" |
| root = Tk()  root.title("Animal Stack Game")  name = Stack(root)  root.mainloop() |

* **Enhancement**:

We incorporated the Tkinter library as well as a class called stack that takes user input and appends it to a stack, the stack is displayed and the items in the stack can be removed one after the other in the order LIFO.

* **Functionality**:
* The program displays brief instructions on how it works
* It has a text input button that allows the user to type in a selected animal from the list given
* Allows the user to then append the animal they chose to the stack by clicking the append button
* It allows the user to see what animals are currently in the stack by clicking the peek button.
* It allows the user to remove the items in the stack by clicking the remove/pop button.
* It also displays an image that corresponds to an animal that is selected.
* **Files**:
* Cow.gif, cat.gif, dog.gif, pig.gif, rat.gif, ant.gif, bat.gif, bee.gif, fox.gif.1.gif
* Gbondo-Mambondiumwe-FP.py
* Gbondo-Mambondiumwe-Fp.docx
* **Utilized Data Structures:**

The original implementation of this project contained trees but we decided to use the stack data structure, Stacks implemented using lists have a big O (1) for peek, insert and remove so this case stacks are more efficient than Binary trees. So, we decided to use stacks instead of trees, Iteration is faster also in stacks than in a binary tree. Binary trees have a Big O(logn) for insert, remove and find while implementing a list.

* **Big O Analysis:**

def append\_list(self): This method has a big(0) of 0(1), which is to insert an item to the stack

* **Resources**:

Tkinter module for The Graphical User Interface that made the program interactive.

New Boston YouTube Python GUI with Tkinter tutorial videos- <https://www.youtube.com/watch?v=RJB1Ek2Ko_Y&list=PL6gx4Cwl9DGBwibXFtPtflztSNPGuIB_d>.

Dr Jan Pearce for content on exploring the Tkinter module. <http://cs.berea.edu/courses/csc226/tasks/t18.gui-making.html>

* **Challenges**:

We had challenges trying to implement the Tkinter Library as it is not really something that was covered in class in detail, so we had to do extra research and watch you tube tutorials.

* **Testing**: A list in bulleted form of all input values or unit tests used for testing. Here you should be careful to select representative input cases, including both representative typical cases as well as extreme cases.

Tried using, Ant, ant and ANT, the program is case sensitive and will only display the image of an animal if it is typed in lower case, the ant was used as a test, however, if you type in ant using uppercase, it will still append what you typed in to a stack but will not display an image. The same goes if you type in an animal that is not on the list, the text, “there is no such animal will be displayed”, however the animal will still be appended to the stack.

* **Errors**:  No errors were observed or noted.
* **Measures and Assessment**:

The program appends items to a stack and allows items to be removed in the LIFO order, so the project was a success.

* **Video: https://youtu.be/M-yg3Yzckac**
* **Summary**:

We initially just wanted to use binary trees as the data structure of choice, as in the original implementation of the assignment A17. We wanted to incorporate the Tkinter library whilst using trees, however, we later decided to do something different so, we instead opted to use stacks. In that way, we would benefit from being able to compare the two data structures, since we already knew how to do the project using trees. So, learning how to do it using stacks would increase our knowledge of two data structures that can be implemented for the same project. We spent about 10 days working on this project, which was about 2 hours a day, which translates to about 20 hours for the whole project.

* **Comments**:

This was a great project and we both had fun implementing our own ideas to improve the Animal guessing game. It made us appreciate more how the stack data structure works. Great class, great project, it has been an amazing semester, we learnt a great deal.