CA341 Assignment 1 - Analytical Comparison of a Procedural and Object-Oriented Programming language

Adrian Irwin - 20415624
Afolabi Fatogun - 20409054

Intro

In this assignment we were tasked to implement a phonebook program which stores data using a binary tree in a procedural and object-oriented programming language.

We chose to use C and Python 3.10 for the procedural and object-oriented implementation respectively. Python is a general purpose programming language which emphasises code readability and indentation. The reason we chose Python was because of the easy to understand syntax and the code can be read and interpreted using Classes. C is also a general purpose programming language which was designed to be compiled to provide low level access to memory. It remains the backbone of many commonly used languages today. We chose to use C for our Procedural implementation because using Structs, we can read and analyse the code procedurally(in sequence of instructions to be executed).

Writing the Code

In the planning phase of the project we decided to split the code from both languages into functions/ objects. We tried as much as possible to write the functions/objects on a one to one basis to further aid in our comparison.

Below is a table with the descriptions of each function in both implementations.
(Note: The functions are equivalent to one another)

Python(Object-Oriented)	C(Procedural)
__init__(self, name, number, address) - The initializer is a constructor for the phonebookEntry class. It pre-defines a phonebook entry.	createEntry(char * name, char * address, int number) - This function allocates memory for a new entry then copies values into it.
insert(self, name, number, address, sortType) - This function accepts 4 arguments to input into the phonebook. The entries in this function are sorted by either its name or its number. This is defined by the sortType argument.	insertEntry(phonebookEntry * root, char * name, char * address, int number, char * sortType) - This function uses the root value to place entries into the phonebook. It also accepts 4 arguments to input into the phonebook. The entries in this function are sorted by either its name or its number. This is defined by the sortType argument.
search(self, searchFor, sortType) - This function searches for an entry in the phonebook based either on the name or number. This is defined by the sortType argument.	searchEntry(phonebookEntry * root, char * name, int number) - This function searches for an entry in the phonebook based either on the name or number. This is determined by the values of name and number.

delete(self, deleteItem, sortType) - This function removes an entry from the phonebook. This is based on either the name or number and is defined by the sortType argument.	deleteEntry(phonebookEntry * root, char * name, int number) - This function removes an entry from the phonebook. This is based on either the name or number and is determined by the values of name and number.
validateNumber(number) - This is a utility function used to ensure that the number entered by the user is a valid one.	getNumberInput() - This function both gets the user's number input and ensures that it is a valid number.
printPhonebook(self) - This function returns all the contents of the phonebook in alphabetical order.	printPhonebook(phonebookEntry * root) - This function returns all the contents of the phonebook in alphabetical order
userPrompt() - This function uses basic try and except error checking. The function would take user input from the getUserChoice() function and go through number's one through five and depending on the users input, perform one of the following functions; - Add a new contact - Delete a contact - Search for a contact - Print all contacts - Exit the Program	userPrompt(phonebookEntry * root, phonebookEntry * numberRoot) - This function would take user input from the getUserChoice() function and go through number's one through five and depending on the users input, perform one of the following functions; - Add a new contact - Delete a contact - Search for a contact - Print all contacts - Exit the Program
getUserChoice() - This function provides directions on how to use the program by displaying all the functionality the program has at once. It also takes the user's input which is sent to userPrompt().	getUserChoice() - This function provides directions on how to use the program by displaying all the functionality of the program. It also takes the user input by calling getNumberInput() which is sent to userPrompt().
findMin(self) - this function finds the smallest value in the phonebook. It is a utility function for the delete function.	The implementation of this function is incorporated into the delete function.

Differences

After we completed the code it was more than clear to us that both programming languages/methods were vastly different. It was clear that both methods posed advantages and disadvantages.

The first key difference we noticed in our implementation was that incorporation of new functions and data was a lot easier in the object-oriented implementation. It seemed to be more intuitive/ reasonable for larger scale programs than procedural programming because of the reusability of the classes in python [1].

The next thing we noticed in our implementation was the issue of memory allocation.

```
// Allocate memory for the new entry and then copy the values into it
phonebookEntry * newEntry = malloc(sizeof(phonebookEntry));
```

In the above snippet, memory is being allocated for the newEntry manually using malloc. While this seems to be a uniquely C experience, Python had no need for memory allocation thereby reducing the learning curve.

Another difference we noticed was the need for exception handling/handlers [2]. In our Python(Objected Oriented) implementation it was necessary to include exception handling as opposed to the C.

```
try:
    while userChoice != 6:
        if userChoice == 1:
            # add a new contact
            name = input("Enter name: ")
            number = validateNumber(input("Enter number: "))
            address = input("Enter address: ")

            root.insert(name, number, address)
            numberRoot.insertNumber(name, number, address)

            userChoice = int(getUserChoice())
```

```
except ValueError:
    print("Invalid input")
    userPrompt()
```

In the above snippets, the error handling was done using a try and except to handle a value error.

Due to the nature of using classes in Python(object-oriented), deleting the root node would destroy the structure of the tree and its children, so to combat this, we opted to have the user input their details on starting the program. The user's details would then serve as the root node which cannot be deleted by the user. On the other hand the procedural implementation(C) did not have this issue so users are allowed to freely delete the root node

[3] Another difference we noticed is the interaction of the programming languages with compilers/interpreters.
To run the program in C we implemented the make file as seen below.

```
all:
    @gcc -o compare compare.c
    @./compare

build:
    @gcc -o compare compare.c

run:
    @./compare

clean:
    @rm -f compare

.PHONY:
    run build all clean

```

Python on the other hand had no need for compiling and ran by running the below snippet in the terminal.

```
afolabi@Afolabi:/mnt/g/My Drive/third year$ python3 compare.py
```

Advantages/Disadvantages

During our implementations we were able to derive advantages and disadvantages of both methods.

Python(Object-Oriented)

Advantages	Disadvantages
Code is easily reusable using classes	Steep learning curve due
Easier to implement new functions into working code	Not suitable for smaller projects
Python is a level language which is easier to understand and navigate	This is more difficult to use with compilers and interpreters

C(Procedural)

Advantages	Disadvantages
The flow of the program can be tracked easily	Not suitable for large scale programs
This form of programming is suitable for small scale programs	More difficult to implement new functions into working code
Compilers and interpreters are easy to pair with this for of programming	C is more complex due to its age

Constraints

- For both programming languages, numbers with leading zeros are not accepted

References

[1] R. Half, “4 Advantages of Object-Oriented Programming,” Roberthalf.com, Jul. 23, 2018. https://www.roberthalf.com/blog/salaries-and-skills/4-advantages-of-object-oriented-programming

[2] Programiz, “Python Exception Handling Using try, except and finally statement,” www.programiz.com. https://www.programiz.com/python-programming/exception-handling

[3] GeeksforGeeks, “Compiling a C program:- Behind the Scenes,” GeeksforGeeks, Nov. 12, 2015. https://www.geeksforgeeks.org/compiling-a-c-program-behind-the-scenes/