Data Structures and Algorithm Design

Assignment 1: Part 1

Contact book

Nowadays, a contact book in the 21st century is easily accessible and well organized, thanks to software developments. People searched through thick books to find a contact in the past; however, it did not take long to find the right contact. There are techniques to search for a contact quickly. People open the book in the middle since a contact book is sorted in alphabetic order. If the initial letter is on the left side, they apply the same method on the left side again until they narrow it so far down to find the contact finally. Many algorithms have a relationship with real-life, like the mentioned example. (Fitrian et al., 2019: 2)

The contact book application should include a name, contact (phone) number, email address, and birthday for each contact. Also, the contact book should be sorted alphabetically like an actual contact book. In addition, the application should have the following features:

1. Add: Creation of a new contact
2. Show: Reading a contact
3. Edit: Update a created contact for any modification
4. Remove: Deleting a created contact

A list is the best suitable data structure for building a contact book since a list is a linear data structure and includes functions such as traversal, insertion, deletion, searching and

sorting. When it comes to linear structure, the data are sorted and stored one after the other in the memory. Furthermore, the types of the elements can be different and combine integers, strings, and floats. An array, by contrast, can only regularly store elements of the same type (Muzumdar, 2022; TechDiferrences, 2022). The following figure demonstrates an example of the desired contact list:

Figure 1: Contact list (own representation based on Muzumdar, 2022)

The contact list is arranged by name (head), phone number, email address, and birthday (tail).

In case of adding a new contact, the user needs to create a new list by applying the correct input. The user will be asked to enter the information to keep the order step by step. Due to the complexity, the information input should be only done in strings. The following pseudocode represents the insertion of the asked information for adding a contact to a list. In addition, the global list “contacts” constitutes the list for every contact that the user adds.

```
contacts = empty list

DEF addContact():
    contactList = empty list
    inputName = INPUT (Enter Given name and Surname)
    inputPhone = INPUT (Enter phone number)
    inputEmail = INPUT (Enter email address)
    inputBirthday = INPUT (Enter birthday)
    APPEND inputName to contactList
    APPEND inputPhone to contactList
    APPEND inputEmail to contactList
    APPEND inputBirthday to contactList
    APPEND contactList to contacts
    Declare contacts
END
```

Since the list “contacts” begin for each contact with the name, the list can quickly be sorted alphabetically by the name with the method sort.

After calling the function, the list “contacts” should include the information of the first contact. For the next feature, reading the contact, the user should be able to search for the contact by entering any information about the desired contact.

```
DEF searchContact(contacts, searchInfo):
    Assign the value of searchInfo in lowercase
    SET i to 0
    searchList = empty list
    WHILE i < the length of contacts:
        Assign the i entry of contacts to searchList
        FOR each information in searchList:
            IF (information in lowercase == searchInfo):
```

```
            Declare the i entry of contacts
        SET i to i + 1
    Declare “Contact does not exist’”
END
```

To edit or delete a contact, the user needs firstly find the desired contact with the search contact function. If the contact exists, the user can choose to change the information of one of the four categories or to delete the entire contact. By any further wrong inputs, the user can enter the contact again and repeat the process.

```
DEF editContact(contacts):

    editList = searchContact(contacts, searchInfo = INPUT( Enter Name or Phone
    Number or Email Address or Birthday)
    IF (editList == “Contact does not exist”):
        Declare “Contact does not exist”
    Else:
        Assign the INDEX of contacts with the search value editList to indexNumber
        editInfo = INPUT (Enter the number for editing the information of the
        following categories, enter '4' for deleting the contact or '5' to exit the program:

          0) Name
          1) Phone Number
          2) Email Address
          3) Birthday
          4) Delete Contact
          5) Quit

        IF (editInfo == “4”):
            POP the INT value of indexNumber from contacts
            Declare contacts
```

```
    IF (editInfo == “5”):
        Declare “No editing completed”
    TRY:
        POP the INT value of editInfo from editList
        INSERT the INT value of editInfo with the INPUT (Enter new information)
        Assign editList to the indexNumber entry of contacts
        Declare contacts
    EXCEPT:
        PRINT “Input wrong, try again”
        Declare function editContact(contacts)
END
```

For the test plan, checkpoints will be set under each function. Then, the program can be tested with the help of calling the functions and the output keyword "PRINT". The following pseudocode shows an example and the expected test results.

For adding two contacts to the list "contacts":

```
addContacts()
[Gianluca Cannone, 0176 84078863, gc22299@essex.ac.uk, 03.10.1995]

addContacts()
[Arron Fox, 0151 81743283, arron.fox@gmail.com, 04.12.1990]

PRINT contacts
```

Output:

[[Gianluca Cannone, 0176 84078863, gc22299@essex.ac.uk, 03.10.1995], [Arron Fox, 0151 81743283, arron.fox@gmail.com, 04.12.1990]]

```
SORT contacts
PRINT contacts
```

Output:

[[Arron Fox, 0151 81743283, arron.fox@gmail.com, 04.12.1990], [Gianluca Cannone, 0176 84078863, gc22299@essex.ac.uk, 03.10.1995]]

PRINT searchList(contacts, INPUT (“Gianluca Cannone”)

Output:

[Gianluca Cannone, 0176 84078863, gc22299@essex.ac.uk, 03.10.1995]

The user wants to change the email address from the contact Gianluca Cannone to gianluca.cannone@gmail.com:

PRINT editContacts(contacts):

The user will be asked to enter the following input:

searchList(contacts, INPUT (“Gianluca Cannone”)
editInfo = INPUT (“2”)
editList = INPUT (“gianluca.cannone@gmail.com”)

Output:

[Gianluca Cannone, 0176 84078863, gianluca.cannone@gmail.com, 03.10.1995]

PRINT contacts

Output:

[[Arron Fox, 0151 81743283, arron.fox@gmail.com, 04.12.1990], [Gianluca Cannone, 0176 84078863, gianluca.cannone@gmail.com, 03.10.1995]]

References:

Fitrian, R., Taufik, I., Ramadhan, M., Mulyani, N., Hutahaean, J., Sitio, A., Sihotang, H. (2019) Digital Dictionary Using Binary Search Algorithm. *Journal of Physics: Conference Series* 1255(1): 1-7. DOI: https://doi.org/10.1088/1742-6596/1255/1/012058

Brookshear, J. G., Brylow, D. (2019), Computer Science: An Overview, Global Edition, 13th Edition, Pearson (Intl). Available from: vbk://9781292263441 [Accessed 28 April 2022].

Muzumdar, A. (2022) *Operating Systems & Data Structures* [Lecturecast]. LSC_PCOM7E MARCH 2022 Launching into Computer Science March 2022. University of Essex Online.

TechDifferences (2022) Difference Between Linear and Non-linear Data Structure. Available from: https://techdifferences.com/difference-between-linear-and-non-linear-data-structure.html [Accessed 30 April 2022].