Data: 23.10.2017 Name: Naiman Alexandru-Nicolae

**Problem statement:**

A family wants to manage their monthly expenses. In order to complete this task, the family needs an application to store, for a given month, all their expenses. Each expense will be stored in the application using the following elements: day (of the month in which it was made, between 1 and 30), amount of money (positive integer) and expense type (one of: housekeeping, food, transport, clothing, internet, others). The family needs an application that provides the following functionalities (each functionality is exemplified):

1. Add a new expense into the list.

add <sum> <category>

insert <day> <sum> <category>

e.g.

add 10 internet – add to the current day an expense of 10 RON for internet.

insert 25 100 food – insert to day 25 an expense of 100 RON for food.

2. Modify expenses from the list.

remove <day>

remove <start day> to <end day>

remove <category>

e.g.

remove 15 – remove all the expenses for day 15.

remove 2 to 9 – remove all the expenses between day 2 and day 9.

remove food – remove all the expenses for food from the current month.

3. Write the expenses having different properties.

list

list <category>

list <category> [ < | = | > ] <value>

e.g.

list – write the entire list of expenses.

list food – write all the expenses for food.

list food > 5 - writes all expenses for food with an amount of money > 5.

list internet = 44 - writes all expenses for internet with an amount of money = 44

4. Obtain different characteristics of sublists.

sum <category>

max <day>

sort <day>

sort <category>

e.g.

sum food – write the total expense for category food.

max day – write the day with the maximum expenses.

sort day – write the total daily expenses in ascending order by amount of money spent.

sort food – write the daily expenses for category food in ascending order by amount of money spent

5. Filter the list of expenses.

filter <category>

filter <category> [ < | = | > ] <value>

e.g.

filter food – keep only expenses in category food.

filter books < 100 – keep only expenses in category books with amount of money < 100 RON

filter clothing = 59 – keep only expenses for clothing with amount of money = 59 RON

6. Undo the last operation that modified program data.

undo – the last operation that has modified program data will be reversed. The user has to be

able to undo all operations performed since program start by repeatedly calling this function.

**1.Feature list**

|  |
| --- |
| **Features** |
| F1. Add a new expense into the list. |
| F2. Modify expenses from the list. |
| F3. Write the expenses having different properties. |
| F4 . Obtain different characteristics of sublists. |
| F5. Filter the list of expenses. |
| F6. Undo the last operation that modified program data. |
| F7.Exit the application |

**2. Running scenario**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **User** | **Program** | **Description** |
| a |  | Give the command: | Shows characteristic message |
| b | list |  | Choose to list all family expenses |
| c |  | Day: 1 Sum: 20 Category: food  Day: 2 Sum: 20 Category: food  Day: 3 Sum: 30 Category: food  Day: 9 Sum: 30 Category: food  Day: 22 Sum: 30 Category: internet  Give the command: | Show list of expenses in a user-friendly format |
| d | add 20 food |  | Add 20 RON expenses to the current day |
| e |  | Expense added to the list  Give the command: | Shows message that confirms that the expense was added with success |
| f | insert 19 200 clothing |  | Add 200 RON expenses to the list  on the 19th  day of this month |
| g |  | Expense added to the list | Shows message that confirms that the expense was added with success |
| h | remove food |  | Choose to remove expenses from the ‘food’ category |
| i |  | Day expenses were removed | Shows message that confirms that the expenses were successfully removed |
| j | list food |  | Choose to list all the expenses from the ‘food’ category |
| k |  | All the expenses for the food are:  No expense here  Give the command: | Shows message that confirms  that there aren’t any expenses from the food category |
| l | rxit |  | The user writes an invalid command |
| m |  | Invalid command  Give the command: | Shows message to tell the user that the given command was wrong |
| n | exit |  | Choose the exit the application |
| o |  | Program ends here with success  Process finished with exit code 0 | Shows message that confirms the end of application |

**4 .Test case table for function addExpense(listOfExpenses, expense)**

|  |  |
| --- | --- |
| **Data: listOfExpenses, expense** | **Result: addExpense(listOfExpenses, expense)** |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [23, 400, ‘food] | True |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [24, 600, ‘food’] | True |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [9, 300, ‘food’] | False |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [23, 400, ‘internet] | True |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [26, 400, ‘clothing] | False |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [9, 400, ‘food] | False |

**5.Test case table for function insertExpense (listOfExpenses, expense)**

|  |  |
| --- | --- |
| **Data: listOfExpenses, expense** | **Result: insertExpense (listOfExpenses, expense)** |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [23, 400, ‘food] | True |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [24, 600, ‘food] | True |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [22, 300, ‘food] | True |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [23, 400, ‘internet] | True |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [26, 400, ‘clothing] | True |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], [9, 400, ‘food] | True |

**6. Test case table for function removeExpenseByDay (day, listOfExpenses)**

|  |  |
| --- | --- |
| **Data: day, listOfExpenses** | **Result: removeExpenseByDay (day, listOfExpenses)** |
| 3, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | True |
| 4, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | False |
| 9, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | True |
| 12, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | False |
| 40, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | False |
| 2, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | True |

**7. Test case table for function removeExpenseByStartEndDay (startDay, endDay, listOfExpenses)**

|  |  |
| --- | --- |
| **Data: startDay, endDay, listOfExpenses** | **Result: removeExpenseByStartEndDay (startDay, endDay, listOfExpenses)** |
| 3,7, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | True |
| 4,6, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | False |
| 8,14, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | True |
| 12,15, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | False |
| 40,50, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | False |
| 2, 8,[[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | True |

**8. Test case table for function removeExpenseByType (typeOfExepense, listOfExpenses)**

|  |  |
| --- | --- |
| **Data: startDay, endDay, listOfExpenses** | **Result: removeExpenseByType (typeOfExepense, listOfExpenses)** |
| ‘food’, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | True |
| ‘other’, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | False |
| ‘internet’, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | True |
| ‘clothing’, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | False |
| ‘housekeeping’, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | False |
| ‘clothing’ ,[[1, 20, 'food'], [2, 20,’clothing’], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']] | True |

**9. Test case table for function listExpenseByType (listOfExpenses, typeOfExpense)**

|  |  |
| --- | --- |
| **Data: listOfExpenses, typeOfExpense** | **Result: listExpenseByType (listOfExpenses, typeOfExpense)** |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘food’ | [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food']] |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘clothing’ | [] |
| ‘internet’, [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘internet’ | [[22, 30, 'internet']] |
| [[1, 20, ‘clothing’], [2, 20, 'food'], [3, 30, ‘clothing’], [9, 30, 'food'], [22, 30, 'internet']], ‘clothing’ | [[1,20,’clothing’],[22,30, ‘clothing’]] |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘internet’ | [[22,30,’internet’]] |
| [[1, 20, 'food'], [2, 20,’clothing’], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘others’ | [] |

**10. Test case table for function listExpenseByTypeAndCondition (listOfExpenses, typeOfExpense, condition, value)**

|  |  |
| --- | --- |
| **Data: listOfExpenses, typeOfExpense, condition, value** | **Result: listExpenseByTypeAndCondition (listOfExpenses, typeOfExpense, condition, value)** |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘food’, ‘>’, 10 | [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food']] |
| [[1, 20, 'food'], [2, 20, 'food'], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘internet, ‘<’, 10 | [] |
| [[1, 20, 'food'], [2, 20, ‘clothing’], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘internet’, ‘>’, 20 | [[22, 30, 'internet']] |
| [[1, 20, 'food'], [2, 20, ‘clothing’], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘clothing’, ‘<’, 10 | [] |
| [[1, 20, 'food'], [2, 20, ‘clothing’], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘internet’, ‘=’, 30 | [[22, 30, ’internet’]] |
| [[1, 20, 'food'], [2, 20, ‘clothing’], [3, 30, 'food'], [9, 30, 'food'], [22, 30, 'internet']], ‘food, ‘<’, 40 | [] |

**11.Tasks**

|  |  |
| --- | --- |
| **Id** | **Description** |
| T1 | Create the initial program structure with start functions and commands dictionary |
| T2 | Create the testInit() function to populate the program with some dummy data |
| T3 | Implement ‘exit’ functionality |
| T4 | Implement ‘add’ functionality that modifies directly the list and test it |
| T5 | Link ‘add’ functionality to the main function |
| T6 | Implement ‘insert’ functionality that modifies directly the list and test it |
| T7 | Link ‘insert’ functionality to the main function |
| T8 | Structure the ‘main remove’ function |
| T9 | Implement ‘remove by day’ function |
| T10 | Implement ‘remove between period of two given days’ function |
| T11 | Implement ‘remove by category’ function |
| T12 | Link all ‘remove’ functions to the main ‘remove’ function |
| T13 | Structure the ‘main list’ function |
| T14 | Implement ‘list all’ function |
| T15 | Implement ‘list by category’ function |
| T16 | Implement ‘list by category with condition’ function |
| T17 | Link all ‘list’ functions to the ‘main link’ function |
| T18 | Implement ‘help’ functionality |