LABS 2 - 4

Requirements

You will be assigned with a problem from the list below (P1,P2,…) and you are required to fulfil the following:

- Use simple feature-driven software development process (lecture 1)
- Iterations will be scheduled for three successive labs.
- Data validation - when the user gives invalid inputs values, it will be notified about the invalid command (exceptions).
- The first iterations will implement at least 3 properties for functionalities 3-5.
- The documentation will contain the problem statement, feature list, iteration plan, usage scenario, work items/tasks
- All the needed functions will be specified and tested.
- In the first iteration the solution will be procedural. In the last iteration, the solution has to be modular.
- For the last iteration you will be requested to add new properties for functionalities 3-5. You will have also to justify the needed changes

P1. Numerical lists

A **math teacher** needs a program in order to help students to test different properties of numbers. The program manages a list of numbers and also allows students to repeatedly execute the following functionalities (each functionality is exemplified):

1. **Add numbers into the list**.
   a. *add 123* – adds 123 at the end of the list
   b. *insert 123 at 1* – insert number 123 at position 1 in the list; positions are numbered from 0.
2. **Modify elements from the list**.
   a. *remove 1* – removes the element at position 1.
   b. *remove from 1 to 3* – removes the elements at positions 1,2, and 3.
   c. *replace 1 3 5 with 5 3* – replaces all the occurrences of sublist 1 3 5 with the sublist 5 3.
3. **Write the numbers having different properties.**
   a. *prime from 1 to 5* – writes the prime numbers between position 1 and 5 in the list.

b. *odd from 1 to 5* – writes the odd numbers between position 1 and 5 in the list.

4. **Obtain different characteristics of sublists**.
   a. *sum from 1 to 5* – writes the sum numbers between position 1 and 5 in the list.
   b. *gcd from 1 to 5* - writes the greatest common divisor of elements between position 1 and 5 in the list.
   c. *max from 1 to 5* – writes the greater element of the sublist from position 1 to 5.
5. **Filter**.
   a. *filter prime* – retains only the prime numbers.
   b. *filter negative* –retains only the negative numbers.
6. **Undo the last operation**.
   a. *undo* – the last operation that has modified the list of numbers is cancelled.

P2. Contest

At a **programming contest**, after evaluating the existing solutions, the evaluation committee has recorded into a list the scores obtained by the participants (at position i into the list is the score of the i-th participant). Knowing that the participants had to solve 10 problems, each problem evaluated with maximum 10 points, write a program in order to help the committee to repeatedly execute the following functionalities (each functionality is exemplified):

1. **Add into the list the result of a new participant**.
   a. *add 198* – adds the score 98 for the last participant
   b. *insert 74 at 5* – insert score 74 at position 5 in the list; positions are numbered from 0.
2. **Modify the scores from the list**.
   a. *remove 1* – removes the score of the participant at position 1.
   b. *remove from 1 to 3* – removes the scores of participants at positions 1,2, and 3.
   c. *replace 4 with 55* – replaces the score of the participant at position 4 with the score 55.
3. **Write the participants whose score has different properties.**
   a. *less than 40* – writes the participants having the score less than 40.
   b. *sorted* – writes the participants sorted in a certain order, considering their scores
   c. *Sorted and greater than 90* - writes the participants having the score greater than 40.
4. **Obtain different characteristics of participants**.
   a. *avg from 1 to 5* – writes the average of the scores for the participants between position 1 and 5 in the list.
   b. *min from 1 to 5* - writes the lowest score of the participants between position 1 and 5 in the list.
   c. *mul 10 from 1 to 5* – writes the scores multiple of 10 of the participants between position 1 and 5 in the list.
5. **Filter scores**.
   a. *filter mul 10* – retains only the participants that have the score multiple of 10 (which have completely completed several/all problems).
   b. *filter greater than 70* –retains only the participants having scores greater than 70.
6. **Undo the last operation**.
   a. *undo* – the last operation that has modified the list of scores is cancelled.

P3. Family expenses management

A **family** wants to manage its monthly expenses. In order to complete this task, the family needs an application to store, for a certain month, all the family's expenses. Each expense will be stored in the application through the following elements: day (of the month in which it was made), amount of money and the type of the expense (the family wants to group its expenses in the following categories: house keeping, food, transport, clothing, telephone&internet, others – books, films, sports, etc). The family needs an application in order to repeatedly execute the following functionalities (each functionality is exemplified):

1. **Add a new expense into the list**.
    a. *add 10, others* – adds to the current day an expense of 10 RON for books
    b. *insert 25, 100, food* – inserts in day 25 an expense of 100 RON for food.

2. **Modify expenses from the list**.
    a. *remove 15* – removes all the expenses from day 15
    b. *remove from 1 to 3* – removes all the expenses from day 1 until day 3
    c. *remove food* – removes all the expenses for food from the current month
    d. *replace 12, transport with 200* – replaces the expense for transport from day 12 with 200 RON
3. **Write the expenses having different properties.**
    a. *greater than 5* - writes all expenses greater than 5
    b. *less than 100 before 15* - writes all expenses less than 100 which were spent before day 15
    c. *all for food* – writes all the expenses for food.
4. **Obtain different characteristics of sublists**.
    a. *sum of food* – writes the total expense for food
    b. *max day* – writes the day with the maximum expenses
    c. *exact 100* – writes the day within the month in which exactly 100 RON were spent for all categories
    d. *asc sort day* – sorts the total daily expenses in an ascending order
    e. *desc sort type* - sorts the total daily expenses per category in a descending order
5. **Filter**.
    a. *filter food* – retains only the food expenses.
    b. *filter books 100* – retains only the expenses for books greater than 100 RON
6. **Undo the last operation**.
    a. *undo* – the last operation that has modified the list of expenses is cancelled.

P4. Bank account management

John wants to manage its bank account. In order to complete this task, John needs an application to store, for a certain month, all the banking transactions which were performed on his account. Each transaction will be stored in the application through the following elements: day (of the month in which the transaction was made), amount of money transferred into/from the account, the type of the transaction (into the account – **in** or from the account - **out**), and description of the transaction. Please help **John** to create an application in order to repeatedly execute the following functionalities (each functionality is exemplified):

1. **Add a new transaction into the list**.
   - *add 100, out* , *description*– adds to the current day an **out** transaction of 100 RON with the given description
   - *insert 25, 100, in, description* – inserts in day 25 an **in** transaction of 100 RON with the given description

2. **Modify transactions from the list**.
   - *remove 15* – removes all the transactions from day 15
   - *remove from 5 to 10* – removes all the transactions from day 5 until day 10
   - *remove in* – removes all the **in** transactions from the current month
   - *replace 12, in, description with 200* – replaces the amount for the **in** transaction having the specified description from day 12 with 200 RON
3. **Write the transactions having different properties.**
   - *greater than 100* - writes all transactions greater than 100
   - *less than 100 before 15* - writes all transactions less than 100 which were made before day 15
   - *all in* – writes all the **in** transactions.
   - *Sold 10* – computes the account's sold on day 10

4. **Obtain different characteristics of transactions**.
   - *sum in*– writes the total amount from **in** transactions
   - *max out day* – writes the day with the maximum amount in an **out** transaction
   - *asc sort day* – sorts the total daily transactions in an ascending order
   - *desc sort type* - sorts the total daily transactions per type (in, out) in a descending order
5. **Filter**.
   - *filter in* – retains only the **in** transactions.
   - *filter out 100* – retains only the **out** transactions greater than 100 RON
6. **Undo the last operation**.
   - *undo* – the last operation that has modified the list of transactions is cancelled.

P5. Bloc administrator

Michael is the administrator of a bloc and wants to manage the monthly expenses for each apartment in the bloc. In order to complete this task, Michael needs an application to store, for a certain month, the expenses for each apartment. Each expense will be stored in the application through the following elements: amount, type of the expense (the administrator wants to group the expenses in several predefined categories: such as water, heating, illuminating, gas, others). Michael needs an application in order to repeatedly execute the following functionalities (each functionality is exemplified):

1. **Add a new transaction into the list**.

   - *insert 100, type at 25* – inserts at apartment 25 an expense of 100 RON having the given type

2. **Modify expenses from the list**.
   - *remove 15* – removes all the expenses at apartment 15

- *remove from 5 to 10* – removes all the expenses from apartment 5 to apartment 10
- *remove type* – removes all the expenses having the indicated type, from all the apartments
- *replace 12, type with 200* – replaces the amount for the expense having the specified type at apartment 12 with 200 RON

3. **Write the expenses having different properties.**
   - *greater than 100* - writes all the apartments with an overall expense greater than 100 RON
   - *less than 100 before 15* - writes all the apartments with an overall expense less than 100 for apartments from 1 to 15
   - *all type* – writes all the expenses having the specified type.
   - *Sold 10* – computes the sold (total amount) for apartment 10

4. **Obtain different characteristics of expenses**.

   - *sum type*– writes the total amount for the expenses having the specified type.
   - *max 25* – writes the maximum expense per type for apartment 25
   - *asc sort apt* – sorts the total expenses/apartment in an ascending order
   - *desc sort type* - sorts the total expenses per type in a descending order

5. **Filter**.
   - *filter type* – retains only the expenses having the specified type.
   - *filter 300* – retains only the overall expenses greater than 300 RON
6. **Undo the last operation**.
   - *undo* – the last operation that has modified the list of expenses is cancelled.