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**).

Family Expenses - Feature list:

~: **add** <sum> <category> - F1
~: insert <day> <sum> <category> - F2
~: **remove** <day> - F3
~: **remove** <start day> to <end day> - F4
~: **remove** <category> - F5
~: **list -** F6
~: **list** <category> - F7
~: **list** <category> [< | = | >] <value> - F8
~: **sum** <category> - F9
~: **max day** - F10
~: **sort day** - F11
~: **sort** <category> - F12
~: **filter** <category> - F13
~: **filter** <category> [< | = | >] <value> - F14
~: **undo** - F15

Iteration plan

Iteration	Planned features
I1	**F1** - add to the current day an expense of <sum> RON for <category>. **F2** - insert to day <day> an expense of <sum> RON for <category>. **F3** - remove all the expenses for day <day>. **F4** - remove all the expenses between <start day> and <end day>. **F5** - remove all the expenses for <category> from the current month.
I2	**F6 -** write the entire list of expenses. **F7** - write all the expenses for <category>. **F8** - writes all expenses for <category> with an amount of money > **F9** - write the total expense for category <category> **F10** - write the day with the maximum expenses. **F11** - write the total daily expenses in ascending order by amount of money spent. **F12** - write the daily expenses for category <category> in ascending order by amount of money spent.
I3	**F13** - keep only expenses in category <category>. **F14** - keep only expenses in category <category> with amount of money < > or = to <value> **F15 -** 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.

Running scenarios

I1.

	User	Program	Description
1		Available commands: ~: add <sum> <category> ~: insert <day> <sum> <category> ~: remove <day> ~: remove <start day> to <end day> ~: remove <category> ~: list ~: list <category> ~: list <category> [< | = | >] <value> ~: sum <category> ~: max <day> ~: sort <day> ~: sort <category> ~: filter <category> ~: filter <category> [< | = | >] <value> ~: undo To see the commands simply type "help"	This menu will appear on screen as soon as the program is executed.
2	add 10 food		Adds the value 10 to the category “food” for the current day.
3		Added successfully!	After addition there will be a confirmation message.
4	list	DAY 15: food 10 RON	Lists the days alongside the categories and values
5	remove 15		This initializes all the categories from day 15 with the value 0, in other words it removes all the expenses for that day.
6		Successfully removed!	After removal there will be a confirmation message.
7	list		Lists the days alongside the categories and values
8		No results!	Since there are no results, a message informing the user of this matter is printed.
9	insert 29 10 food		Inserts the value 10 to the category “food” for day 29.
10		Inserted successfully!	After insertion there will be a confirmation message.
11	list	DAY 29: food 10 RON	Lists the days alongside the categories and values
12	remove 29 to 30		This initializes all the categories from day 1 to day 4 with the value 0, in other words it removes all the expenses for those days.

	User	Program	Description
13		Successfully removed!	After removal there will be a confirmation message.
14	list		Lists the days alongside the categories and values
		No results!	Since there are no results, a message informing the user of this matter is printed.
15	remove food		This initializes all of the categories named “food” from day 1 to day 30.
16		Successfully removed!	After removal there will be a confirmation message.
17	list		Lists the days alongside the categories and values
18		No results!	Since there are no results, a message informing the user of this matter is printed.

I2.

	User	Program	Description
1		Available commands: ~: add <sum> <category> ~: insert <day> <sum> <category> ~: remove <day> ~: remove <start day> to <end day> ~: remove <category> ~: list ~: list <category> ~: list <category> [< l = l >] <value> ~: sum <category> ~: max <day> ~: sort <day> ~: sort <category> ~: filter <category> ~: filter <category> [< l = l >] <value> ~: undo To see the commands simply type "help"	This menu will appear on screen as soon as the program is executed.
2	list		List the day alongside the categories and values.
3		DAY 1 internet 10 RON DAY 5 food 20 RON DAY 7 others 80 RON DAY 12 food 20 RON DAY 13 food 100 RON internet 50 RON others 100 RON DAY 14 clothing 100 RON DAY 15 housekeeping 100 RON DAY 16 internet 25 RON DAY 17 food 137 RON others 10 RON DAY 18 food 30 RON DAY 29 transport 100 RON	Days printed in chronological order containing the categories and values that indeed have values.
4	list food		List a specific category for all of the days.

	User	Program	Description
5		DAY 5 food 20 RON DAY 12 food 20 RON DAY 13 food 100 RON DAY 17 food 137 RON DAY 18 food 30 RON	Days that contain the category 'food' inserted by the user printed in chronological order containing the category and value.
6	list food > 20		List the days where the value for food is greater than 20.
7		DAY 13 food 100 RON DAY 17 food 137 RON DAY 18 food 30 RON	Days that contain the category food with the value greater than 20 printed in chronological order.
8	list transport = 100		List the days where the value for transport is exactly 100.
9		DAY 29 transport 100 RON	Days that contain the category transport with the value equal to 100 printed in chronological order.
10	list food = 30		List the days where the value for food is exactly 30.
11		No results!	Since there are no "food" categories containing the value "30" the program will print the result "No results!".
12	sum food		List the sum for food throughout the whole month.
13		The sum of food is 307 RON	The program will print the expected sum.
14	max day		List the maximum value in a day.
15		The maximum expense (250 RON) occurred on day 13	The program prints the day with the most expenses.
16	sort day		List the days sorted by the sum of the values in a day.

	User	Program	Description
17		Sorting by day... Day 2: 0 RON Day 3: 0 RON Day 4: 0 RON Day 6: 0 RON Day 8: 0 RON Day 9: 0 RON Day 10: 0 RON Day 11: 0 RON Day 19: 0 RON Day 20: 0 RON Day 21: 0 RON Day 22: 0 RON Day 23: 0 RON Day 24: 0 RON Day 25: 0 RON Day 26: 0 RON Day 27: 0 RON Day 28: 0 RON Day 30: 0 RON Day 1: 10 RON Day 5: 20 RON Day 12: 20 RON Day 16: 25 RON Day 18: 30 RON Day 7: 80 RON Day 14: 100 RON Day 15: 100 RON Day 29: 130 RON Day 17: 147 RON Day 13: 250 RON	The program will print the sorted days in ascending order.
18	sort food		List the days sorted by the value of the “food” category in a day.

	User	Program	Description
19		Sorting by food... Expenses for food on day 1: 0 RON Expenses for food on day 2: 0 RON Expenses for food on day 3: 0 RON Expenses for food on day 4: 0 RON Expenses for food on day 6: 0 RON Expenses for food on day 7: 0 RON Expenses for food on day 8: 0 RON Expenses for food on day 9: 0 RON Expenses for food on day 10: 0 RON Expenses for food on day 11: 0 RON Expenses for food on day 14: 0 RON Expenses for food on day 15: 0 RON Expenses for food on day 16: 0 RON Expenses for food on day 19: 0 RON Expenses for food on day 20: 0 RON Expenses for food on day 21: 0 RON Expenses for food on day 22: 0 RON Expenses for food on day 23: 0 RON Expenses for food on day 24: 0 RON Expenses for food on day 25: 0 RON Expenses for food on day 26: 0 RON Expenses for food on day 27: 0 RON Expenses for food on day 28: 0 RON Expenses for food on day 30: 0 RON Expenses for food on day 5: 20 RON Expenses for food on day 12: 20 RON Expenses for food on day 18: 30 RON Expenses for food on day 29: 30 RON Expenses for food on day 13: 100 RON Expenses for food on day 17: 137 RON	The program will print the sorted days in ascending order by the values of the “food” category.

I3.

	User	Program	Description
1		Available commands: ~: add <sum> <category> ~: insert <day> <sum> <category> ~: remove <day> ~: remove <start day> to <end day> ~: remove <category> ~: list ~: list <category> ~: list <category> [< l = l >] <value> ~: sum <category> ~: max <day> ~: sort <day> ~: sort <category> ~: filter <category> ~: filter <category> [< l = l >] <value> ~: undo To see the commands simply type "help"	This menu will appear on screen as soon as the program is executed.
2	filter food		Filter the food, meaning this will delete every other category other than 'food'
3		Data was filtered successfully!	Confirmation message from the program.
4	list		List the content in the files.
5		DAY 5 food 20 RON DAY 12 food 20 RON DAY 13 food 100 RON DAY 17 food 137 RON DAY 18 food 30 RON DAY 29 food 30 RON	The content of the files, ascending by day, category and value
6	filter food > 20		Filter the food with the value greater than 20, meaning that every other category than 'food' is deleted and all the food categories with the value smaller than 20

	User	Program	Description
7		Data was filtered successfully!	Confirmation message from the program.
8	list		List the content in the files.
9		DAY 13 food 100 RON DAY 17 food 137 RON DAY 18 food 30 RON DAY 29 food 30 RON	The content of the files, ascending by day, category and value
10	filter food < 101		Filter the food with the value less than 101, meaning that every other category than 'food' is deleted and all the food categories with the value higher than 101.
11		Data was filtered successfully!	Confirmation message from the program.
	list		List the content in the files.
		DAY 13 food 100 RON	The content of the files, ascending by day, category and value
	undo		This initiates the undo feature which reverses the last operation that was applied (add, insert, remove, filter).
		Undone!	Confirmation message from the program.
	list		List the content in the files.
		DAY 13 food 100 RON DAY 18 food 30 RON DAY 29 food 30 RON	The content of the files, ascending by day, category and value

	User	Program	Description
	undo		This initiates the undo feature which reverses the last operation that was applied (add, insert, remove, filter).
		Undone!	Confirmation message from the program.
	list		List the content in the files.
		DAY 13 food 100 RON DAY 17 food 137 RON DAY 18 food 30 RON DAY 29 food 30 RON	The content of the files, ascending by day, category and value
	undo		This initiates the undo feature which reverses the last operation that was applied (add, insert, remove, filter).
		Undone!	Confirmation message from the program.
	list		List the content in the files.
		DAY 5 food 20 RON DAY 12 food 20 RON DAY 13 food 100 RON DAY 17 food 137 RON DAY 18 food 30 RON DAY 29 food 30 RON	The content of the files, ascending by day, category and value
	undo		This initiates the undo feature which reverses the last operation that was applied (add, insert, remove, filter).
		Cannot complete operation. Nothing to undo.	Error message from the program.

	Work Items/Tasks	Description
1	getCommand(s)	This function returns the COMMAND (add, insert, ...) inserted by the user.
2	checkIntegrityOfTheFiles(categoryList)	The way that this program works is it hold a file for each day of the month, that is 30 files, each containing a dictionary <category> <day>. This function assures that everything is fine with the files, if one is missing or corrupt, it replaces it with a new one.
3	passSpaces(s, i, l)	This function returns the index of the next string character other then space (' ')
4	passCommand(s, i, l)	Just like the 'passSpaces' function, this function returns the index but of the position of the next space, thus passing a group of characters other than (' ')
5	passInput(s, i, l)	The role of this function: returns the index of the next space and the text between the last space and the next space
6	getCategAndSum(s)	This function is used alongside the “add” command and function, providing it with the <category>, <sum> and if there are other characters entered it will also return those remaining ones
7	getDayCategAndSum(s)	This function is used alongside the “insert” command and function, providing it with the <day>, <category>, <sum> and if there are other characters entered it will also return those remaining ones
8	getRemainder(s)	This function returns the remainder after the first word from a string
9	initializeDictionary(day)	This function returns the dictionary containing the <categories> and the <values> specific to the day <day> from the text files
10	fileUpdate(fileName, auxDictionary)	This function updates the file with the name “fileName.txt” with the auxDictionary.
11	buildCategList()	This function is responsible for building the list of categories in case I need a list later on
12	buildCmdList()	In case the user needs help understanding how the program works, wants to know the commands By typing "help" this function builds a <list of commands>

	Work Items/Tasks	Description
13	removeTo(s)	This function treats the case in which the <remove> command was entered and also containing the string " to " in it input: string - which is a part of the user input output: returns the <start day>, <end day> and if there are any other characters after the valid instruction the function returns it as well
14	removeDayOrCateg(s)	input: string - a part of the user input output: the day or category after the <remove> command followed by the remainder in case there are any characters after the day or category
15	removeExpenses(a, b, undo_steps, step_count)	This function removes all the expenses for day in the interval [a, b] pass in by parameters (updates the files) input: a, b - start day and end day
16	removeExpensesByCategory(categ, undo_steps, step_count)	The function removes all the expenses for a certain category from day 1 to day 30 (updates all the files with the default dictionary) input: the category
17	printExpenses(category, t, symbol, val)	This function iterates through all of the files and prints all the expenses if there are any in this format: DAY x internet y others z ...
18	getSum(category)	This function returns the sum of all the values corresponding to each day's category
19	getExpensesForDay(auxDict)	This function returns the sum of all the expenses for this dictionary input: a dictionary containing categories and values output: the sum of all the values
20	updateMax(maxExp, auxExp, i, day)	This is basically a “max” function used for alongside the “MAX” command.
21	maximumExpenses()	This function iterates thorough all of the files, check the expenses and memorizes the day with the most expenses

	Work Items/Tasks	Description
22	sortByDayOrCateg(categ, op)	This function allows for the iteration from 1 to 30 For each day, the program calculates the expenses for that day and stores it into a list Or calculates the expenses for each day for a specific category This list is sorted at the end of the function and returned output: The sorted list of values.
23	deleteAllExcept(category, symbol, value, undo_steps, step_count)	This function formats all the categories except the one that was passed through the parameter input: the category that we would like to filter. This function will filter all of the categories leaving the category from the parameters intact if and only if it the condition regarding the sign is fulfilled input: the category, sign and value.
24	userHelp(cmdList)	This function prints all of the commands that the user may enter.
25	add(userInput, categoryList, undoSteps, step)	This function treats the case in which the user has entered the “add” command.
26	insert(userInput, categoryList, undoSteps, step, undo)	This function treats the case in which the user has entered the “insert” command.
27	remove(userInput, categoryList, undo_steps, step_count)	Remove all the expenses for a certain category for the whole month.
28	list(userInput, categoryList)	List all of the content from the data files.
29	sum(userInput, categoryList)	Write the total expense for a certain category.
30	max(userInput, categoryList)	Write the maximum expense of a day in a month.
31	sort(userInput, categoryList)	Sort data from the data files based on the conditions imposed by the user.
32	filter(userInput, categoryList, undo_steps, step_count)	Keep only expenses in a specific category.
33	undo(userInput, categoryList, undo_steps, step_count)	This function reverses the last operation that was applied (add, insert, remove, filter).
34	main()	This is the main function.
35	initializeFile(i):	This function initializes file 'i.txt' with the default categories and values.

	Work Items/Tasks	Description
29	stepCountUpdate(undo_steps, sum, categ, day, step_count)	This function updates the undo_steps list with the most recent operation inserted by the user and at the same time the length of the list