Case

➢ Nonstructural Computation

1. Table for Seeking Prime Numbers with Sieve Method

- **Problem**

Sieve method is an old method to seek prime number, which aims to find out all the prime numbers not exceeding natural number N (N>1). The specific method is: Firstly arrange N natural numbers in order. The minimum prime number is 2, so cancel 1. From 2 (keeping 2), cancel all the numbers behind 2 that can be exactly divided by 2. In this way, the first number behind 2 that is not cancelled is 3, keep 3, and then cancel all the numbers behind 3 that can be exactly divided by 3. The first number behind 3 that is not cancelled is 5, keep 5, and then cancel all the numbers that can be exactly divided by 5. Go on and on like this, after each screening, the next adjacent number must be a prime number. In this way, it is possible to finally screen out all composite numbers that do not exceed N, and the numbers remained are all prime numbers that do not exceed N.
Please code in the esProc with this algorithm to find out all the prime numbers within 10,000.

- **Tip**

Rough train of thought: Declare an ordered sequence from 1 to 10,000, cancel 1 and loop from the minimum prime number 2; assign 0 to the numbers that can be exactly divided by the current number, keep the current number. The numbers in the final sequence that are not equal to 0 are all the prime numbers within 10,000.

1. Define a sequence in descending order from 1 to 10,000.
2. Assign 0 to the members in the sequence whose value is 1.
3. Loop the ascending sequence from 2, and assign 0 to the numbers in the sequence that can be exactly divided by the current recurring number in the loop body.
4. The numbers remained that are not 0 are prime numbers.

- **Code**

	A	B	C
1	10000		
2	=to(A1)		

Generate a sequence from 1 to 10,000.

3	>A2(1)=0			Assign 0 to members whose value is 1.
4	for A2	if A4>0		Loop Sequence A2. If the current number is not 0, it indicates that the number is prime number. Set as 0 all the numbers behind it that can be exactly divided by this number.
5			=A1.step(A4,A4+A4)	
6			>A2(C5)=0	
7	=A2.select(~>0)			The final remaining numbers that are not 0 are prime numbers.

- **Result**

Member
2
3
5
7
11
13
17
19
23
29
31

2. Fibonacci Series

- **Problem**

Fibonacci Series refers to such a series: 1, 1, 2, 3, 5, 8, 13, 21 …
From the third term in this series, each term is equal to the sum of the two terms before it.
Please output the first N terms of this series by insert function and three algorithms, namely, iterative, recursion, and esProc series.

- **Tip**

Rough train of thought: Define a sequence and assign 1 to the first two terms; then loop it for N times, respectively obtain the last term and the last but one term and add the sum of the two terms to the sequence. The result is just the generated Fibonacci Series.

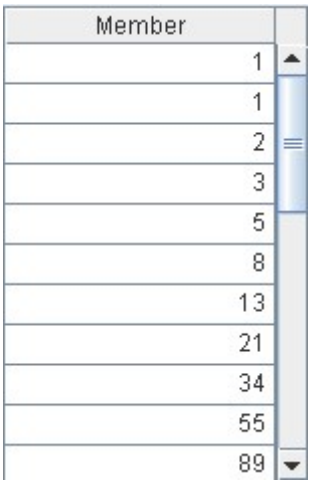
1. Define the quantity of Fibonacci Series.

2. Use the iterative algorithm to declare a series that has only two members whose values are both 1. Loop it for N-2 times, in the loop body, add the sum of the last term and the last but one term to the end of the series.
3. Use the recursion algorithm to declare two parameters *a* and *b*, assign 0 and 1 to them respectively. Loop it for N times, recursively compute *a* and *b*, compute New b = Old *a* + Old *b*, assign the Old *b* to the New *a*, and return the New *a* as the member of the result series.
4. Use the esProc subroutine, the parameter is N, and when the number of the series is ≦ 2, return [1,1] to end the computation. When the quantity of the series is ＞2, recursively invoke the subroutine and append the sum of the last two terms to the end of the sequence, the recursion ends when the quantity of the sequence is 2.

- **Code**

	A	B	C	
1	30			Declare the quantity of Fibonacci Series
2	/Iterative algorithm			
3	=[1,1]			Define the sequence.
4	for A1-2	=A3.m(-1)+A3.m(-2)	>A3=A3|B4	Loop it for A1-2 times, add the last term and the last but one term in A3, and add the sum to A3.
5	/Recursion algorithm			
6	>a=0,b=1			Declare two parameters, and assign 0 and 1 to them respectively.
7	=A1.((b=a+b,a=b-a))			Loop it for A1 times, recursively compute *a* and *b*, and compute New b = Old a + Old *b*, and assign the Old *b* to the New *a*, return the New a as the member of the result sequence.
8	/esProc function, the parameter is the quantity of the series.			
9	=func(A10,A1)			Invoke A10 subroutine, and the parameter is A1.
10	func	if A10<=2	return A10*[1]	Loop the parameter values and return sequence [1, 1] when the parameter values are smaller than 2.
11		=func(A10,A10-1)	return B11|(B11.m(-1)+B11.m(-2))	Recursively invoke A10 subroutine and overlap the sum of the last two terms to B11 until A10 is 2.

- **Result**



3. Repetend

- **Problem**

Repetend is the part of the numbers repeating in infinite repeating decimals. All infinite repeating decimals can be expressed as a fractional number. So as long as a fractional number is given, it is possible to seek out its repetend. Please develop a program to implement this algorithm.

- **Tip**

Rough train of thought: Seek the remainder of the operation between the dividend and the divisor, and judge whether the remainder appeared in the previous computation. If the remainder has not appeared yet, then multiply the remainder by 10 and take the result as the dividend. Continue the loop and repeat the computation to seek the remainder. If it appears, then end the loop, the quotient between the position where the same remainder appears last time and the position where the same remainder appears this time is the repetend in the infinite repeating decimals.

1. Seek the remainder by dividing the dividend by the divisor.
2. Firstly judge in the loop body whether the remainder appears in the preceding computation. If it appears, then suspend the loop.
3. If the remainder has not appeared, then add the current remainder to historical remainder sequence, and then multiply the current remainder by 10, take the result as the dividend, and divide the result by the divisor. Add the resulting quotient into the historical quotient sequence, replace the current remainder with the resulting remainder, and enter the next round of loop.
4. According to the bits from the position where the current remainder is found in the historical remainders to the position of the last bit, seek the quotient at the corresponding position in the historical quotient sequence, then we will get the repetend

- **Code**

	A	B	C	
1	99			Dividend
2	140			Divisor
3	=A1%A2			Obtain the remainder
4	for	=B6.pos(A3)		Define the infinite loop. If the remainder obtained in this loop has appeared in the remainders in previous loops, then end the loop, and B4 stores the position in which the remainder appears.
5		if B4>0	break	
6		=B6|A3		Store the remainder computed each time.
7		=B7|int(A3*10/A2)		Store the quotient computed each time.
8		>A3=A3*10%A2		Refresh A3 to appear as the current remainder.
9	= B7.to(B4).conj@s()			In B7, part from position B4 to the last number is the repetends. Take out the repetends and put them into string.

- **Result**

值
714285

Value

4. Simulate Multiple Recursion

- **Problem**

Please compile a program to answer the following 10 questions:
1. Which is the question whose first answer is b?
(a) 2; (b) 3; (c) 4; (d) 5; (e)6
2. The only question which consecutively has two same answers is:
(a) 2, 3; (b) 3, 4; (c) 4, 5; (d) 5, 6; (e) 6, 7;
3. Which is the question whose answer is the same as the answer to this question?
(a) 1; (b) 2; (c) 4; (d) 7; (e) 6
4. The number of questions with the answer a is:
(a) 0; (b) 1; (c) 2; (d) 3; (e) 4
5. Which is the question whose answer is the same as the answer to this question?
(a) 10; (b) 9; (c) 8; (d) 7; (e) 6
6. What is the answer to the questions whose number is the same as the number of question s whose answer is a?
(a) b; (b) c; (c)d; (d) e; (e) none of the above
7. According to the alphabetical order, how many letters are there between the answer to this question and the answer to the next question?

(a) 4; (b) 3; (c) 2; (d) 1; (e) 0. (Note: there is one letter between a and b)

8. The number of question s whose answer is vowel is:

(a) 2; (b) 3; (c) 4; (d) 5; (e) 6. (Note: a and e are vowels)

9. The number of questions whose answer is consonant is:

(a) a prime number; (b) a factorial number; (c) a square number; (d) a cubic number; (e) multiple of 5

10. The answer to this question is:

(a) a; (b) b; (c) c; (d) d; (e) e

- **Tip**

Rough train of thought: Record the options a, b, c, d, and e in the question respectively as 1, 2, 3, 4, and 5, so as to facilitate the increment of the answers. Firstly set all answers to the questions to 0, and then add, in turn, 1 to every answer to the question according to the sequence, and conduct judgment on various answers as required in the question. If the answer does not comply with the requirement of a certain question, then add 1 more to the answer till it complies with the requirement. If there is still no proper answer after 5 answers are all set for once, then return to reset the answer to the preceding question. When encountering question whose answer is 0, skip it directly and make no judgment.

i. Initialize answers to the 10 questions as 0, and the serial number of question is 1.
ii. Loop from the first question, firstly add 1 to the answer to the question, judge whether the answer to this question is 6. If it is 6, it indicates there are more than 5 answers, then set the answer to this question as 0, i-1, and return to the preceding question to keep the loop going.
iii. Then invoke sub-function to judge whether answers to the question are reasonable, and if they are not reasonable, then proceed to the next loop.
iv. If they are reasonable, i+1, and then judge whether i is equal to 11, and if it is equal to 11, it indicates 10 questions have been set. So convert the final answers into string for storage, and set i as i-1 and return to the preceding question to continue with the loop to see whether there is any other answer that satisfies the condition, and end the loop when i is equal to 1.
5. The sub-function judges question by question whether the answers are reasonable. The judgment order is firstly to see whether this question or the relevant question is 0. If it is 0, it indicates no answer is set, skip over the judgment; if it is not 0, then conduct judgment according to the aim of this question.

- **Code**

	A	B	C	D
1	=[0]*10	>i=1		

A1 is used to store the answers to 10 questions, the initial value is 0, and i is the serial number of question.

2	for i>0	>A1(i)=A1(i)+1			Add 1 to the answer to the question.
3		if A1(i)==6	>A1(i)=0,i=i-1	next	If the answer is still incorrect when the answers are set to the last option, then set the answer to the current question as 0, return to the preceding question, add 1 to the answer to the preceding question and loop again.
4		if !func(A8)	next		Invoke function to judge whether answers set this time are reasonable. If they are not reasonable, then enter the next loop, and continue to add 1 to the answers.
5		>i=i+1			If the setting of the answers to the question is reasonable, then set answers to the next question.
6		if i==11	=C6|A1.(char(asc("a")+~-1)).conj@s()	>i=i-1	If all the answers are reasonable when the answers to the last question has been set, convert the answers into letters and store them in C6. Set the value of i as i-1, return to the preceding question to see whether any other answer satisfies the requirement, End the loop when i is equal to 1.
7	/This is a judge function to check if the answer to every question is correct. If answer is 0, then it indicates there is no answer yet, and “true” will be returned.				
8	func	if A1(1)!=0			Jude whether answers are set for the first question.
9			=A1(A1(1)+1)		According to the answers to the first question, find the first question whose answer is b.
10			if C9!=0 && C9!=2 ||A1(to(A1(1))).pos(2)>0	return false	If the answer set for the question found in C9 is not b, or before this question, there is already an answer to this question, which is b, then return “false”.

11		if A1(2)!=0	=A1(2)+1		According to the answers to the second question, find the serial numbers of two consecutive questions which have the same answer.
12			if A1(C11)*A1(C11+1)>0 && A1(C11)!=A1(C11+1)	return false	/if answers set for the question found in C11 are not the same as the answers to the following question, return "false".
13			if A1.pselect(~>0 && ~==~[1] && #!=C11)>0	return false	/If the answers to another two questions are the same, it indicates that C11 is not unique, return "false".
14		if A1(3)!=0	=A1([1,2,4,7,6](A1(3)))		According to the answers to the third question, find the question answer whose answers are the same as those of the third question.
15			if C14!=0 && C14!=A1(3)	return false	/if C14 is not 0 and is not the same as the answers to the third question, return "false".
16		=A1.count(~==0)	=A1.count(~==1)		B16 computes out the number of questions for which no answers are set. C16 computes out number of questions whose answer is a.
17		if A1(4)!=0	=C16-A1(4)+1		C16 deducts the answer to the fourth question. Because there is a difference of 1 between the serial number of answer and the answer value, add 1.
18			if C17>0 || C17<0 && B16==0	return false	If the result computed by C17 is greater than 0, that is, the set number of questions whose answer is a is greater than the number of choices in the forth question; Or when answers are set for all questions, but C17 is still less than 0, that is, the answer given in the set choice of

					the fourth question (to count the questions whose answer is a) is greater than the actual number of questions whose answer is a return “false”.
19		if A1(5)!=0	=A1([10,9,8,7,6](A1(5)))		According to the answer to the fifth question, find the question whose answer is the same as that of the fifth question.
20			if C19!=0 && C19!=A1(5)	return false	If C19 is not 0 and is not the same as the answer to the fifth question, return “false”.
21		if B16==0	=A1.align([2,3,4,5]).(~.len())		C12 computes the number of questions whose answers are respectively 2, 3, 4, and 5.
22			if !((A1(6)!=5 && C21(A1(6))==C16) || (A1(6)==5 && C21.pos(C16)==0))	return false	If the answer to the sixth question is 5 and the number of answers that are 5 is different from the number of all the questions of C12, then return true. If the answer to the sixth question is not 5 and the number of the questions to which its answer corresponds is consistent with that in C12, then return “true”. For other situations, return “false”.
23		if A1(7)!=0	if A1(8)!=0 && abs(A1(8)-A1(7))!=5-A1(7)	return false	If answers are set for the eighth question, and the number of choices between the answer to the eighth question and the answer to the seventh question is inconsistent with the answer choice for the seventh question, return “false”.
24		if A1(8)!=0	=C16+A1.count(~==5)-A1(8)-1		Compute out the sum of the numbers of questions whose answers are a and e and its difference value when subtracting the answers to the

					eighth question.
25			if C24>0 || C24<0 && B16==0	return false	If the result computed in C24 is greater than 0, that is, the existing vowel answers are more than the number of the choices in the eighth question, or when answers are set for all questions, but C24 is still less than 0. That is, answers are set for all questions, but the number of vowels is still less than the number of the choices in the eighth question, then return "false".
26		if B16==0	[[2,3,5,7],[1,2,6],[0,1,4,9],[0,8],[0,5,10]]		Cell C26 defines number of possible questions of various options in the ninth question.
27			if C26(A1(9)).pos(A1.count(~>=2 && ~<=4))==0	return false	If the computed number of consonants is not in C26's sequence, then return "false".
28		return true			

- **Result**

Member
cdebeedcba

5. Dealing Cards

- **Problem**

Four persons play poker with a pack of cards without big and little jokers. Shuffle the remaining 52 cards and deal out four hands of 13 cards each. Indicate the suits of poker with letters A, B, C, and D; indicate the number cards with 2 to 10; indicate J, Q, K, and A with 11, 12, 13 and 1 respectively. For example, A8 indicates Spade 8, B13 indicates Heart K. Write a program to implement the dealing process and print out four sequences.

- **Tip**

Rough thought: Arrange the numbers 1-52 randomly and divide them into 4

groups. Call a subroutine in which the current number is passed as an argument. Compute the card corresponding to this number in subroutine, and return a card indicated by a letter for the suit and this number. This is the dealing process.

i. Randomize the number 1-52.
ii. Divide the 52 numbers into 4 groups.
iii. Sort the numbers in each group. Repeatedly call the subroutine into which the number is passed and return a card indicated by a letter for the suit and this number.

- **Code**

	A	B	
1	=52.sort(rand())		Shuffle the cards.
2	=A1.group(int((#-1)/13))		Divide the cards into four groups.
3	=A2.(~.sort().(func(A5,~)))		Call the subroutine to return a result.
4			
5	func	=mid("KA234567890JQ",1+A5%13,1)	Get the number for the card.
6		=mid("ABCD",int((A5-1)/13+1),1)	Get the suit for the card.
7		return B6+B5	Return a card indicated by this number and the suit letter

- **Results**

Member
[A4,A6,A7, ...]
[AA,A3,A8, ...]
[A9,AK,B2, ...]
[A2,A5,AJ, ...]

6. Word Appearance Rate

- **Problem**

In a formal English document, words are separated by blank, comma, full stop, and carriage return, and the sign "-" is used to connecting the characters before and after the carriage return so as to constitute a whole word.

Now suppose there is a document characterized with the above-mentioned rules. Please count the total number of different words in the said document, make statistics on the frequency that each word appears, and pick out the word with the highest appearance frequency.

- **Tip**

Load the document content, break the document content into sequence consisting of single characters, and then convert all upper-case letters in the sequence into lower-case letters and change all non-letter characters in the sequence into blanks. Delete consecutive blanks into one blank, put sequence together to form a string, and then break the string again with blank into sequences, so they form sequences in which one word is one member. Group the same words into one group, and after the sub-group is returned, the member value with largest length is the word with the highest appearance frequency.

1. Read the document content.
2. Break the document content into sequence consisting of single characters, and then convert all upper-case letters in the sequence into lower-case letters and change all non-letter characters in the sequence into blanks.
3. Delete consecutive blanks into one blank, put sequence together to form a string, and then break the string again with blank into sequences, so they form sequences in which one word is a member.
4. Group the same words into one group, and after the sub-group is returned, the member value with largest length is the word with the highest appearance frequency.

- **Code**

	A	
1	E:\\esProc exercise\\word.txt	
2	=file(A1).read()	
3	=len(A2).(mid(A2,~,1)).(if(isalpha(~),lower(~)," "))	Break the document content into sequence consisting of single characters, and then convert all upper-case letters in the sequence into lower-case letters and change all non-letter characters in the sequence into blanks.
4	=A3.select(~!=" " || ~[-1]!=" ")	Delete consecutive blanks into one blank.
5	=A4.conj@s().array(" ")	Put sequences together to form a string, and then break the string again with blank into sequences, so they form sequences in which one word is a member.
6	=A5.group().maxp(~.len())(1)	Group sequences, query the member with the largest length after grouping, and it is the word with the highest appearance frequency.

- **Result**

Value
the

7. Perpetual Calendar

- **Problem**

Please give a full year calendar for a specific year to come (2020 for example) as designated. The outcome shall be strings in a format similar to calendar and be separated by month and year with the date and week day information included.

- **Tip**

Rough train of thought: Firstly, set the strings to cycle in a loop of 12 months of one year; calculate and fill the titles of the month and the week onto the perpetual calendar. Secondly, set the strings to cycle in a loop of all days of a specific month, concatenate the date in weeks and separate them with the \t. Once the concatenation for one week is done, fill the outcome onto the perpetual calendar and proceed to the next week.

i. Define the year and the outcome columns of the perpetual calendar, and calculate the week day of Jan 1
ii. Cycle through the 12 months, and set the header field for each month.
iii. Cycle the days of the current month through the loop and concatenate the strings in weeks
iv. After concatenating one week, fill the outcome onto the perpetual calendar and proceed to the next week
v. After cycling one month, fill the stings of the remaining days less than one week onto the perpetual calendar

- **Code**

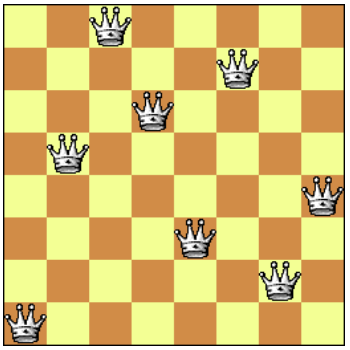
	A	B	C	D	
1	1984	=[string(A1)+"Year"]			Store the resulting perpetual calendar
2	=day@w(date(A1,1,1))				Store the week day of the current day in the loop, the initial value is Jan 1
3	for 12	>B1=B1|["\n","\t\t\t"+string(A3)+"Month","Day\tMon\tTue\tWed\tThu\tFri\tSat"]			Cycle the months and set the header field for each month
4		=fill("\t",A2-1)			Store the current week string during the process of loop. Regarding the first week of each month, the indentation of the first

					day must be considered.
5		for days(date(A1,A3,1))			Cycle all day in the current month and concatenate the week to the current week string B4 by day
6			>A2=A2+1		
7			>B4=B4+string(B5)		
8			if A2==8	>A2=1	Return to Monday after cycling through the whole week
9				>B1=B1|B4	Add the week string cycled through to the perpetual calendar
10				>B4=""	
11			else	>B4=B4+"\t"	Otherwise, add \t as the separator between days
12		if B4!=""	>B1=B1|B4		The last week of each month maybe less than 7 days. These days shall also be filled onto the perpetual calendar.

- **Outcome**

Member						
1984Year						
			1Month			
Day	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				
			2Month			
Day	Mon	Tue	Wed	Thu	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29			

8. Eight Queens



A solution for Eight Queens Problem

- **Problem**

The Eight Queens problem is an old and famous

problem. It is described as follows: Put eight queens on the chessboard with 8X8 cells such that they do not attack each other, i.e. any two queens cannot be in the same row, the same column, or the same oblique line. How many possible ways are there in total?

- **Tip**

Consider each row in the chessboard with 8X8 cells as a sequence, and assign numbers 1, 2, 3, 4, 5, 6, 7, 8 to each cell in the column in order.

i. Firstly put the first queen in the first row and first column, and mark it in the sequence.
ii. Go to the next row. If a queen exists in this row and this column, move it to the next column in this row. If it reaches the last column, return back to the previous row and move the queen in this row to the next column.
iii. Evaluate whether the queen in current row is in the same oblique line with the queens in all rows above this row. If they are in the same oblique line, move the queen in current row to the next column. If it reaches the last column, return back to the previous row and move the queen in this row to the next column.

- **Code**

	A	B	C	D	
1	=[0]*8	>i=1			A1 stores the place of queen in each row on the chessboard, and B1 initializes the starting row.
2	for i>0	>A1(i)=A1(i)+1			B2 sets the place of queen in Row i, and evaluates whether it is reasonable starting from 1 place by place.
3		if A1(i)==9	>A1(i)=0,i=i-1	next	If it is not reasonable until the queen in current row reaches the last column, remove the queen in this row and return back to the previous row, and adjust the place of queen in previous row.
4		if i==1	>i=2	next	Evaluating for the first row is not required; just go to the next row for evaluation.
5		=A1(i)	=A1(to(i-1))		
6		if C5.pos(B5)>0		next	If a queen already exists in the column that the queen in current row locates, move the queen in current row to the next column for reevaluation.
7		if C5.pselect(i-#==abs(B5-~))>0		next	If a queen already exists in the oblique line that the queen in current row locates, move the queen in current row to the next column for reevaluation.
8		>i=i+1			Go to the next row if the queen in this row meets the requirement.

9		if i==9	=C9|A1.conj@s()	>i=i-1	Save the result if eight rows are set completely. Go into the next loop by modifying the result of previous row to find out another placement of queens.
10	=C9.len()				Count the number of all possibilities to put the queens.

- **Results**

Value
92

9. Game of 24-Point

- **Problem**

The 24-point game is a classic puzzle game using poker to carry out the calculation. The game is detailed as follows: Take out the big and little jokers from a pack of poker, and extract any four from the remaining 52 cards. Use the addition, subtraction, multiplication, division and parentheses to operate the numbers on the four cards (J, Q, K, and A representing 11, 12, 13, and 1 respectively) to get the result of 24. Each card must be used once, but not repeatedly.

Write a program to operate any four cards for the result of 24, and print out its solution in the form of text. If no solution exists, print out "No solution".

- **Tip**

i. Arrange 4 numbers in different orders.
ii. In the first layer of loop, take any three from the 4 operators and arrange them in different ways.
iii. In the second layer of loop, insert the three operators into the four numbers.
iv. Enumerate the five cases of parentheses as follows (where A, B, C, and D represent numbers, # represents different operators):
   - ((A#B)#C)#D
   - (A#(B#C))#D
   - (A#B)#(C#D)
   - A#((B#C)#D)
   - A#(B#(C#D))

- **Code**

	A	B	C	D
1	[3,3,8,8]			
2	[1234,1243,1324,1342,1423,1432,2134,2143,2314,2341,2413,2431,3124,3142,3214,3241,3412,3421,4123,4132,4213,4231,4312,4321]			

3	for A2	=A1(4.(int(A3/power(10,4-#))%10))			Arrange the elements of A1 as those of A2.
4		for 64			Select any three from the four operators (+, -, *, /) which can be selected more than once. Therefore, any one operator has four choices, which means 4×4×4=64 possibilities exist.
5			=["+","-","*","/"](3.(int((B4-1)/power(4,3-#))%4+1))		Convert the sequence number of loop into the quaternary number and add 1 for each bit of it. The result is the current selection of operators.
6			>a=B3(1),b=B3(2),c=B3(3),d=B3(4),x=C5(1),y=C5(2),z=C5(3)		
7			=func(A28,[x,a,func(A28,[y,b,func(A28,[z,c,d])])])		Add parentheses into the expression: A#(B#(C#D)), and call the subroutine for evaluation.
8			if abs(C7-24)<0.0001		
9				=string(a)+x+"("+string(b)+y+"("+string(c)+z+string(d)+"))"	
10				=D10|D9	Store the result if the value of expression is equal to 24.
11			=func(A28,[x,a,func(A28,[z,func(A28,[y,b,c]),d])])		Add parentheses into the expression: A#((B#C)#D), and call the subroutine for evaluation.
12			if abs(C11-24)<0.0001		
13				=string(a)+x+"(("+string(b)+y+string(c)+")"+z+string(d)+")"	
14				=D14|D13	Store the result if the value of expression is equal to 24.
15			=func(A28,[y,func(A28,[x,a,b]),func(A28,[z,c,d])])		Add parentheses into the expression: (A#B)#(C#D), and call the subroutine for evaluation.
16			if abs(C15-24)<0.0001		
17				="("+string(a)+x+string(b)+")"+y+"("+string(c)+z+string(d)+")"	
18				=D18|D17	Store the result if the value of expression is equal to 24.
19			=func(A28,[z,func(A28,[y,func(A28,[x		Add parentheses into the

			,a,b]),c]),d])	expression: ((A#B)#C)#D, and call the subroutine for evaluation.
20			if abs(C19-24)<0.0001	
21			="(("+string(a)+x+string(b)+")"+y+string(c)+")"+z+string(d)	
22			=D22|D21	Store the result if the value of expression is equal to 24.
23			=func(A28,[z,func(A28,[x,a,func(A28,[y,b,c])]),d])	Add parentheses into the expression: (A#(B#C))#D, and call the subroutine for evaluation.
24			if abs(C23-24)<0.0001	
25			="("+string(a)+x+"("+string(b)+y+string(c)+"))"+z+string(d)	
26			=D26|D25	Store the result if the value of expression is equal to 24.
27	=[D10:D26].union().id()			Result of operating
28	func			In three arguments, the first is an operator, the second is a left operand, and the third is a right operand.
29		if A28(1)=="+"	return A28(2)+A28(3)	
30		if A28(1)=="-"	return A28(2)-A28(3)	
31		if A28(1)=="*"	return A28(2)*A28(3)	
32		if A28(1)=="/"	return A28(2)/A28(3)	

- **Results**

Member
8/(3-(8/3))

10. Inspecting Checksum of Barcode (#60)

- **Problem**

The barcode is scanned to form into a 13-digit string. In order to deal with the mistake in scanning, the checksum can be used to inspect whether mistakes exist in the barcode. Detailed calculation rules are as follows:

Extract the first 12 digits from the 13-digit string, add the number on each odd digit to obtain a sum S1, and add the number on each even digit to obtain a sum S2. Subtract S2 from S1, the difference mod 10 to obtain a remainder, and take its absolute value. The result should be equal to the 13^{th} digit of the barcode. Otherwise, this barcode is not correct.

Evaluate the correctness of a given barcode.

- **Tip**

Rough train of thought: Convert the 13 digits of the given barcode into a sequence, find out the sum of numbers on odd digits and that on even digits for the first 12 digits in the sequence, implement subtraction for the two sums, then the difference mod 10 to obtain a remainder, take its absolute value, finally evaluate whether the result is equal to the number on the last digit of the barcode.

- **Code**

	A	
1	="1234567890123"	The given barcode
2	=len(A1).(int(mid(A1,~,1)))	Convert the barcode into a sequence
3	=A2(to(12)).step(2,1).sum()	The sum of numbers on odd digits for the first 12 digits
4	=A2(to(12)).step(2,2).sum()	The sum of numbers on even digits for the first 12 digits
5	=abs((A3-A4)%10)	The difference between the two sums, mod 10, take the remainder's absolute value
6	=A5==A2.m(-1)	Evaluate whether the result of A5 is the same as the number on the last digit of the barcode

- **Results**

Value
false

➢ Database Computation

1. Play Matchmaker

- **Problem**

Romeo is an NS GSOH M Veronian (nonsmoking, good-sense-of-humor male who lives in Verona). Juliet WLTM (would like to meet) an NS GSOH M. Will Romeo do?

The Suitor table shows the names of each suitor:

NAME
ROMEO
PARIS

The Has table (Table 7-17) shows their qualities:

NAME	HAS_QUALITY
ROMEO	NS

ROMEO	GSOH
ROMEO	VERONIAN
ROMEO	M
PARIS	NS
PARIS	M
……	

The Wltm table shows the features that Juliet demands:

NAME	QUALITY
JULIET	NS
JULIET	GSOH
JULIET	M

Please find the suitor having all three qualities that Juliet required.

- **Tip**

Rough Train of Thought: You can retrieve the qualities of each suitor first, and then cycle every suitor to compare each suitor's qualities against the features demanded by Juliet. Finally, return the list of persons with the required qualities.

	A	
1	=file("C:\\txt\\Suitor.txt").import@t()	
2	=file("C:\\txt\\has.txt").import@t()	
3	=file("C:\\txt\\Wltm.txt").import@t()	
4	=A2.group(Name)	Group suitor by name
5	=A3.select(Name=="JULIET").(QUALITY)	Qualities required by Juliet
6	=A4.select(~.(Has_Quality).posi@p(A5)!=null)	Group of suitors meets Juliet 's demand
7	=A6.(Name)	Return the name list of suitors meeting the requirement

- **Result**

成员
ROMEO

2. Postage Computation

- Problem

A B2C website needs to compute the postages for orders. In most cases, the postage is computed by the total weight, and postage is free for orders over $300. Detail rules are given in the **mailCharge** table below:

FIELD	MINVAL	MAXVAL	CHARGE
COST	300	1000000	0

WEIGHT	0	1	10
WEIGHT	1	5	20
WEIGHT	5	10	25
WEIGHT	10	1000000	40

The table below records the postages of various value range. For example, the first record indicates that the postage is zero (deliver for free) if the value of cost field is between 300 and 1000000.The second record indicates that the value of weight field is between 0 and 1 kg, and the postage is 10 dollars.

The table below is about some orders placed at the Web site:

ID	COST	WEIGHT(KG)
JOSH1	150	6
DRAKE	100	3
MEGAN	100	1
JOSH2	200	3
JOSH3	500	1

Please compute the detailed postage for these orders.

- **Tip**

Rough Train of Thought: Respectively find the records whose value of filed field is the cost and the weight. Then cycle all order records. First, determine if the cost value in the order record is up to the free postage threshold. If not satisfied, then determine the postage level according to its weight.

- **Code**

	A	B	C	D
1	=file("C:\\txt\\mailCharge.txt").import@t()			/Select from **mailCharge** table
2	=file("C:\\txt\\testOrder.txt").import@t()			/Select from **testOrder** table
3	=A1.select(FIELD=="COST")			/Retrieve the record of free postage
4	=A1.select(FIELD=="WEIGHT").sort(MINVAL:-1)			/Retrieve the record of charging according to postage
5	>A2.derive(POSTAGE)			/Add **postage** record to the order table
6	for A2			
7		if A3.MINVAL<A6.COST		/When order price exceeds the level of free
8			>A6. POSTAGE=A3.CHARGE	/Free of postage
9			next	
10		for A4		
11			if A6.WEIGHT>B10.MINVAL	/Determine the postage level according to the weight
12				>A6. POSTAGE=B10.CHARGE
13				next A6

- **Result**

ID	ID1	COST	WEIGHT	POSTAGE
1	JOSH1	150	6	25
2	DRAKE	100	3	20
3	MEGAN	100	1	10
4	JOSH2	200	3	20
5	JOSH3	500	1	0

3. Generate the Text Histogram

- **Problem**

This problem is aimed to simulate the histogram via text.

Below is the Employee table, in which the **DEPTNO** field indicates the department the employee belongs to.

EMPNO	ENAME	DEPTNO
7934	MILLER	10
7782	CLARK	10
7839	KING	10
7902	FORD	20
7788	SCOTT	20
7876	ADAMS	20
7566	JONES	20
7369	SMITH	20
7900	JAMES	30
7844	TURNER	30
7654	MARTIN	30
7521	WARD	30
7499	ALLEN	30
7698	BLAKE	30

Now, we use text histogram to indicate the number of employees of each department, one "*" represents one employee, and the result set illustrated by the horizontal histogram should be like this:

DEPTNO	CNT
10	***
20	*****
30	******

The result set illustrated in the vertical histogram should be like this:

D10	D20	D30
		*
	*	*
	*	*

*	*	*
*	*	*
*	*	*

Please develop the program to generate the result set.

- Tip

i. Horizontal histogram: First, create a table sequence with **DEPTNO** and **CNT** fields. Group by department in the **DEPTNO** table. Cycle against the grouped results and insert the department and duplicate * for the times of department persons.

ii. Vertical histogram: First, create a new table sequence with the field name of dynamically retrieved department, make statistics on the employees of each department, retrieve the maximum value, and insert the maximum number of blank records to the table sequence. Against the table sequence, cycle by column, modify the table sequence vertically, and insert the * to the table sequence.

- Code

	A	B	
1	=file("C:\\txt\\DEPTNO.txt").import@t()		Load the department table
2	/Horizontal histogram		
3	=create(DEPTNO,CNT)		Construct the result sequence
4	=A1.group(DEPTNO)		Group by DEPTNO
5	for A4		
6		=A3.insert(0,A5.DEPTNO,fill("*",A5.count()))	Insert records to the sequence. The first field is DEPTNO, and the second field is the * repeated for the number of employees
7	/Vertical histogram		
8	=create(${A4.(DEPTNO).string()})		Result sequence; field name is the dynamically retrieved DEPTNO
9	=A4.(~.count())		Count the departmental employee numbers
10	=A9.max()		Count the employee numbers of the largest department
11	>A8.insert(A10)		Insert the blank record according to the maximum department numbers
12	for A9		
13		=A8.paste@h((A10-A12)*[null]|A12*["*"],#A12)	Fill in the * column by column

- **Result**

Horizontal histogram returned in **A3**

DEPTNO	CNT
10	***
20	*****
30	******

Vertical histogram returned in **A8**

10	20	30
		*
	*	*
	*	*
*	*	*
*	*	*
*	*	*

4. List the Management in Tree Structure

- **Problem**

The following is the Employee table of an enterprise, inclusive of the employee ID and name:

EMPNO	ENAME
7902	FORD
7788	SCOTT
7900	JAMES
7844	TURNER
7654	MARTIN
7521	WARD
7499	ALLEN
7934	MILLER
7876	ADAMS
7782	CLARK
7698	BLAKE
7566	JONES
7369	SMITH
7839	KING

The subordinate relations among employees are given in the below table. For example, the first record indicates that the manager of Employee 7902 is the Employee 7566. The 14th record indicates the Employee 7839 who has no manager. He is the highest manager.

EMPNO	MGR
7902	7566
7788	7566
7900	7698
7844	7698
7654	7698
7521	7698
7499	7698
7934	7782
7876	7788

7782	7839
7698	7839
7566	7839
7369	7902
7839	

Please return a result set to describe the hierarchical relationships of the whole table. This result set must be in the below structure:

EMP_TREE
KING
KING – BLAKE
KING – BLAKE – ALLEN
…
KING – CLARK
KING – CLARK – MILLER
…

- **Tip**

Rough train of thought: First, select out an employee A, and then select out its line manager B, and then the C that is line manager of B. After finding all these persons, loop until reaching the line manager that is the highest leader. Then, jump out of the inner loop, and search and cycle the line manager of the next employee.

Finally, sort the result and you will get the desired results.

- **Code**

	A	B	C	D	
1	=file("C:\\txt\\Employees2.txt").import@t()				Find the employee information
2	=file("C:\\txt\\Relationships.txt").import@t()				Find the relation between employees
3	=[]				Store the subordinates relation tree having been found
4	for A1				Loop against the employee table
5		=A4.EMPNO			Get the number of the current employee
6		=A4.ENAME			Get name of the current employee
7		for			
8			=A2.select(EMPNO==B5).MGR		Get the upper class of the current employee
9			if C8==null	break	If there is no line manager of the current employee, then put an end to the inner loop, and jump to the next employee
10			=A1.select(EMPNO==C8).ENAME		Get name of the line manager of the current employee
11			>B6=C10+"-"+B6		Add the present line manager of the employee to the cell **B6**
12			>B5=C8		Proceed to select out the line manager's line manager
13			next		
14		>A3=A3|B6			Store the found hierarchy tree in the cell **A3**
15	=A3.sort()				Sort the cell **A3** and display the desired result as required in the problem

- **Result**

成员
KING
KING-BLAKE
KING-BLAKE-ALLEN
KING-BLAKE-JAMES
KING-BLAKE-MARTIN
KING-BLAKE-TURNER
KING-BLAKE-WARD
KING-CLARK
KING-CLARK-MILLER
KING-JONES
KING-JONES-FORD
KING-JONES-FORD-SMITH
KING-JONES-SCOTT
KING-JONES-SCOTT-ADAMS

5. Top n Clients Accounting for Half of the Total Sales in This Year

- **Problem**

The table below is the historic sales contracts of an enterprise:

ContractNo	ActualSale	SellDate	Product	Quantities	Amount	Client	ApplyArea	ApplyMethod
10961	8	1997-03-19	52	6	1122.0	QUEEN	EastChina	SELF_USE
10962	8	1997-03-19	7	45	3584.0	QUICK	SouthChina	SELF_USE
10963	9	1997-03-19	60	2	68.0	FURIB	NorthChina	RESELL
10964	3	1997-03-20	18	6	2052.5	SPECD	SouthChina	RESELL
10965	6	1997-03-20	51	16	848.0	OLDWO	EastChina	SELF_USE
10966	4	1997-03-20	37	8	1255.6	CHOPS	SouthChina	SELF_USE
10967	2	1997-03-23	19	12	910.4	TOMSP	EastChina	RESELL
10968	1	1997-03-23	12	30	1408.0	ERNSH	SouthChina	SELF_USE
10969	1	1997-03-23	46	9	108.0	COMMI	NorthChina	RESELL
10970	9	1997-03-24	52	40	280.0	BOLID	Northeast	SELF_USE
10971	2	1997-03-24	29	14	1733.06	FRANR	EastChina	RESELL
10972	4	1997-03-24	17	6	251.5	LACOR	EastChina	SELF_USE
10973	6	1997-03-24	26	5	291.55	LACOR	EastChina	RESELL
10974	3	1997-03-25	63	10	439.0	SPLIR	NorthChina	SELF_USE

The table below is the Client table:

ID	Name	City	State	Country	SalesGoal
ALFKI	Alfreds Futterkiste	Shenyang	Liaoning	China	3
ANATR	Ana Trujillo Emparedados	Dalian	Liaoning	China	1
ANTON	Antonio Moreno Taquería	Yingkou	Liaoning	China	10
AROUT	Around the Horn	benxi	Liaoning	China	6
BERGS	Berglunds snabbk?p	Shijiazhuang	Hebei	China	5
BLAUS	Blauer See Delikatessen	Langfang	Hebei	China	8
BLONP	Blondesddsl père et fils	tangshan	Hebei	China	5
BOLID	Bólido Comidas preparad	jinan	Shandong	China	8
BONAP	Bon app'	qingdao	Shandong	China	9
BOTTM	Bottom-Dollar Markets	dongying	Shandong	China	10
BSBEV	B's Beverages	Ankleiqi	Alaska	USA	5
CACTU	Cactus Comidas para llev	Fermi	Alaska	USA	7
CENTC	Centro comercial Moctezu	Los Angeles	California	USA	9
CHOPS	Chop-suey Chinese	Hollwwood	California	USA	9

Clients ranked by sales value in a certain year. The top n clients accounting for half of the total sales value is called as Key account. Please list the key account of this enterprise in the year of 1998.

- **Tip**

Rough train of thought: Firstly, group the Contract table by Client, compute the total sales value achieved by each client and arrange the results in descending order, then work out the half value of the total sales value. Finally, scan this table and aggregate sales value during this process till reaching half of the total sales value. Then, the a few top clients will be the key accounts.

i. In order to facilitate the client name retrieval, update the Client field in the Contract table to the corresponding Client records.
ii. Select out the Contract record in the 1998
iii. Group by Client for the screened table. Then, the sales contract of each client will be grouped together, and a new table sequence will be generated. Please sum up the amount of each group.
iv. Sort this table sequence in descending order
v. Compute the total sales volume of all clients, that is, half of the total annual sales value.
vi. Traverse every record top-down with pselect function, and sum up the amount till the amount reaches half of the total sum. Once traverse stopped, you will get the number of key accounts.
vii. Retrieve the name of top n clients, and these clients are the key account of this year.

- **Code**

	A	
1	=file("C:\\txt\\Contract.txt").import@t()	**Contract** table
2	=file("C:\\txt\\Client.txt").import@t()	**Client**
3	>A2.primary(ID), A1.switch(Client,A2)	Update the client field of contract table to client record
4	=A1.select(year(SellDate)==1998)	Screen out the sales record in **1998**

5	=A4.group@i(Client;~.sum(Amount):Amount)	Group by **client** in the table, and sum up the amount of each group
6	=A5.sort(Amount:-1)	Sort by amount in descending order
7	=A5.sum(Amount)/2	Half sales volume of this year
8	=0	Temporary variable of aggregate amount
9	=A5.pselect((A8=A8+Amount,A8>=A7))	Find the half position of the aggregate amount
10	=A5(to(A9)).(Client.Name)	Retrieve the names and they are the names of big client of this year

- **Result**

成员
Ana Trujillo Emparedados y helados
Antonio Moreno Taquería
Around the Horn
Berglunds snabbk?p
Blauer See Delikatessen
Blondesddsl père et fils
Bólido Comidas preparadas
Bon app'
Bottom-Dollar Markets
B's Beverages
Cactus Comidas para llevar
Chop-suey Chinese
Comércio Mineiro

6. Stock Rise to the Limit for 3 Consecutive Days in a Month

- **Problem**

To list the daily closing price record of a stock exchange in a month, in which the **CODE** column is used to contain the stock code, **DT** is the date, and **CL** is the closing price level.

Please select out the stock rise to the limit for 3 consecutive days in this month. In order to avoid the error caused by rounding off, the rate of rise-up limit is 9.5%.

CODE	DT	CL
001001	2009-01-01	4.0
001026	2009-01-01	2.13
001028	2009-01-01	20.09
001070	2009-01-01	14.95
001107	2009-01-01	3.74
001134	2009-01-01	10.68
001137	2009-01-01	42.31
001147	2009-01-01	19.61
001206	2009-01-01	14.01
001213	2009-01-01	40.94
001228	2009-01-01	9.69
001236	2009-01-01	134.61
001242	2009-01-01	21.1

- **Tip**

Rough train of thought: Sort the record by code and date, and group by the stock code. Thus, you will get a monthly price list of a stock. In this way, you can easily compute the rate of ups and downs for each stock. By comparing the rate of ups and downs, you can judge if the stock rises to its limit and make the final statistics on stocks rising to its limit for 3 consecutive days.

1. First, add a field to each record in the journal table to record the rate of ups and downs comparing with that of the previous day. The value of this field is left empty temporarily.
2. Sort by code and date in the table. The purpose of this operation is to guarantee the grouped sequence in the step followed. In each group, the data is sorted by date.
3. Group by stock code in the table. Because the table is already sorted by code, no sorting will be performed again.
4. For each record in each group, work out the rate of ups and downs. Please note if this is the first record in the group, then there will be no previous record. The rise and drop shall be taken as 0.
5. Select the stock rising to its limit for consecutive 3 days for once or above times, and retrieve the code.

- **Code**

	A	
1	=file("C:\\txt\\StockRecords.txt").import@t()	Stock exchange journal
2	0.095	Limit of rising-up
3	=A1.derive(:UP)	Add a column to record the ranges of ups and downs
4	=A1.sort(CODE,DT)	Sort the table by code and date
5	=A4.group@o(CODE)	Group by code
6	=A5.run(~.run(UP=if(#==1,0,(CL-CL[-1])/CL[-1])))	Work out the rate of ups and downs, and write it to the UP field
7	=A6.select(~.count(UP>A2 && UP[-1]>A2 && UP[-2]>A2)>0).(CODE)	Select out the codes of stocks rising to its limit for consecutive 3 days for once and above times

- **Result**

成员
201745
550766
600045
700071

7. Count the Couple whose Total Ages are Over 70

- **Problem**

An enterprise is planning to allocate the welfare house with low price to its couple employees. This welfare is only for couple employees both currently serve in the enterprise, and total ages of

the spouses is not less than 70. The table below is the Employee table:

ID	Name	Gender	Post	Birthday	AccountNo	BasePay
1	Mike	Female	Sale	1968-12-08	53693689129	5600.0
2	Jake	Male	Vice Presider	1962-02-19	96410767719	2500.0
3	Lucy	Female	Sale	1973-08-30	66524824514	10800.0
4	Andy	Male	Sales Manag	1968-09-19	65002886054	7500.0
5	Jim	Male	Sales Manag	1965-03-04	44138024798	4700.0
6	David	Male	Sale	1967-07-02	86091675703	9300.0
7	Jessica	Male	Sale	1960-05-29	56889966430	2900.0
8	Lily	Female	Inside Sales	1969-01-09	42884466796	3500.0
9	Mary	Female	Sale	1969-07-02	96244795103	7200.0
10	Tiger	Male	General Man	1970-10-10	25549086420	4300.0
11	Kate	Female	Human Resc	1980-04-05	68398410697	7600.0
12	Al	Male	Sale	1980-03-06	73902984872	7500.0
13	Ben	Male	R&D Leader	1984-09-13	04521217732	10700.0

The table below is the employee relation table, in which the relations of couple employees are recorded. Both **Emp1** and **Emp2** fields are used to hold the employee ID. There is no duplicate relation in this table:

Emp1	Emp2	Relationship
21	22	Spouse
10	1	Spouse
5	19	Spouse
16	3	Spouse

Try to find the couple of which the sum of both spouses' ages attain or exceed 70.

- **Tip**

Rough train of thought: Regarding this issue, all you need is to update the Employee field in the relation table to the corresponding employee record. Then, you can directly retrieve the birthday of the relevant employee. Then, it is much easier to compute the age, age sum, and filter the results.

i. Firstly, select a group of employees whose relation is spouse from the relation table.
ii. Secondly, replace the employee field of selected record with the corresponding employee record in the employee table according to employee ID. Then, you can retrieve the birthday of each employee from this table directly and thus get the age.
iii. Generate a new table sequence on the basis of this table, including the name and total age of the two employees.
iv. Select out the record whose total age is not less than 70.

- **Code**

	A	
1	=file("C:\\txt\\Employees.txt").import@t()	Employee table
2	=file("C:\\txt\\EmpRel.txt").import@t()	Employee Relation table
3	=A2.select(Relationship=="Spouse")	Select out the employee couple
4	>A1.primary(ID)	Set the primary key for the employee

5	>A3.switch(Emp1,A1; Emp2,A1)	Replace the records of the two employee fields in the Employee Relation table with the corresponding record
6	=A3.new(Emp1.Name:Emp1,Emp2.Name:Emp2, age(Emp1.Birthday)+age(Emp2.Birthday):TotalAge)	Generate a new table sequence by utilizing the employee relation table and sum up the ages of every couple
7	=A6.select(TotalAge>=70)	Screen out the couple whose total age is over 70

- **Result**

Emp1	Emp2	TotalAge
Tiger	Mike	82
Jim	Howard	78

8. Employees who have worked in New York for over 5 years cumulatively

- **Problem**

The table below is an employee info list of a company.

ID	Name	Gender	Post	Birthday	AccountNo	BasePay
1	Mike	Female	Sale	1968-12-08	53693689129	5600.0
2	Jake	Male	Vice Presider	1962-02-19	96410767719	2500.0
3	Lucy	Female	Sale	1973-08-30	66524824514	10800.0
4	Andy	Male	Sales Manag	1968-09-19	65002886054	7500.0
5	Jim	Male	Sales Manag	1965-03-04	44138024798	4700.0
6	David	Male	Sale	1967-07-02	86091675703	9300.0
7	Jessica	Male	Sale	1960-05-29	56889966430	2900.0
8	Lily	Female	Inside Sales	1969-01-09	42884466796	3500.0
9	Mary	Female	Sale	1969-07-02	96244795103	7200.0
10	Tiger	Male	General Man	1970-10-10	25549086420	4300.0
11	Kate	Female	Human Resc	1980-04-05	68398410697	7600.0
12	Al	Male	Sale	1980-03-06	73902984872	7500.0
13	Ben	Male	R&D Leader	1984-09-13	04521217732	10700.0

Employees in the company are transferred regularly due to business reasons. To record the transfer history, a table as below is created to record the work transfer of each employee ever since his on-boarding. (Those employees who had quitted the company are not included in the list)

Employee	Date	Area	Type
1	1995-01-11	Philadelphia	Onboarding
5	1995-03-31	Dallas	Onboarding
18	1995-11-27	San Diego	Onboarding
4	1996-01-15	Las Vegas	Onboarding
13	1996-03-01	Washington, D. C.	Onboarding
16	1996-11-16	Houston	Onboarding
5	1996-12-19	San Diego	Post transfer
12	1997-01-07	Chicago	Onboarding
9	1997-01-12	Las Vegas	Onboarding
13	1997-05-03	Los Angeles	Post transfer
10	1997-06-13	Boston	Onboarding
1	1997-06-20	Dallas	Post transfer
22	1997-07-08	Las Vegas	Onboarding

Each transfer date and destination is included in the table. Please find out employees who have worked in New York for over 5 years cumulatively.

- **Tip**

Rough train of thought: The key to this problem is to change the table structure from recording by employees transfer event to by working years. And then, the working years of each phase can be operated and filtered easily.

1. Firstly, update the Employee field with corresponding records in the employee table. This step facilitates the problem solving greatly.
2. Group the transfer table by Employee to create a new table sequence. Reserve a field AccWorkDays_NYC to record the accumulated working years of employees in New York.
3. Loop every group, i.e., every employee.
4. Create a new table sequence by employee transfer table, which should include fields of StartDate, EndDate, and WorkArea. StartDate refers to the start time of the transfer. EndDate refers to the time for taking the next record. The last record should be the current time. The WorkArea refers to the transferred city of current record. In this way, a work history table logging by working year is created.
5. Select the working years in New York and work out the total days of working. Write the result to the AccWorkDays_NYC field.
6. At the end of the loop, select those records whose AccWorkDays_NYC is over 5 years.
7. To facilitate viewing, a new table sequence can be built to only include the employee names and working years in New York.

- **Code**

	A	B
1	=file("C:\\txt\\Employees.txt").import@t()	Employee table
2	=file("C:\\txt\\EmpTransfer.txt").import@t().sort(Date)	Employee transfer table
3	>A1.primary(ID), A2.switch(Employee,A1)	Update the Employee field with corresponding record in the transfer table

4	=A2.group@i(Employee;~:g,:AccWorkYear_NYC)		Group by employees and reserve a field of WorkYear
5	for A4	=A5.g	Loop every employee
6		=B5.new(~.Date:StartDate, if(#==B5.len(),now(),B5(#+1).Date):EndDate, Area:WorkArea)	Change the table to display by date. The last record uses the current date as End Date.
7		=B6.select(WorkArea=="New York")	Select the working years of employees who have ever worked in New York.
8		>A5.AccWorkDays_NYC=B7.sum(daysAfter(StartDate,EndDate))	Sum up total working days and write it to the reserved field.
9	=A4.select(AccWorkDays_NYC>=5*365)		Select employees who have worked in New York for over 5 years.
10	=A9.new(Employee.Name:Employee, int(AccWorkDays_NYC/365):AccWorkYears_NYC)		Change the format to make it more intuitive.

- **Result**

Employee	AccWorkYears_NYC
Jake	6
David	8

9. Commodities with Longest Accumulated Out-of-Stock Time in a Month

- **Problem**

The tables below are from a simplified supermarket stock control system. To check if the purchase policy is proper, the supermarket needs to find out the commodity with the longest accumulated out-of-stock time in 6 months. The supermarket opens at 8 a.m. and closes at 9:30 p.m. The out-of-stock time of non-business hours will not be counted. The desired table is as follows:

ID	Name	Catagory	Price
20001	----	Food	63
20002	----	Drink	87
20003	----	Drink	26
20004	----	Fresh	88
20005	----	Electrical equipment	57
20006	----	Food	62
20007	----	Electrical equipment	63
20008	----	Fresh	18
20009	----	Food	90
20010	----	Vegetables	55
20011	----	Electrical equipment	75
20012	----	Vegetables	20
20013	----	Drink	26

The supermarket replenishes its stock at 5 a.m. every day as shown in the below purchase table which records the volume of every piece of commodity.

Datetime	Commodity	Volume
2009-06-21 05:00:00	20064	232
2009-06-21 05:00:00	20006	164
2009-06-21 05:00:00	20031	376
2009-06-21 05:00:00	20066	696
2009-06-21 05:00:00	20037	12
2009-06-21 05:00:00	20094	120
2009-06-21 05:00:00	20065	1128
2009-06-21 05:00:00	20097	1044
2009-06-21 05:00:00	20016	100
2009-06-21 05:00:00	20090	648
2009-06-21 05:00:00	20099	900
2009-06-21 05:00:00	20095	776
2009-06-21 05:00:00	20053	792

The table below is about the remaining stock table by the last day of May:

Date	Commodity	Stock
2009-05-31	20001	188
2009-05-31	20002	61
2009-05-31	20003	83
2009-05-31	20004	184
2009-05-31	20005	60
2009-05-31	20006	157
2009-05-31	20007	186
2009-05-31	20008	136
2009-05-31	20009	99
2009-05-31	20010	164
2009-05-31	20011	38
2009-05-31	20012	174
2009-05-31	20013	182

The table below is a detailed sales record of the supermarket:

Datetime	Commodity	Volume
2009-06-01 08:05:00	20077	28
2009-06-01 08:11:40	20056	47
2009-06-01 08:18:20	20094	34
2009-06-01 08:21:40	20020	19
2009-06-01 08:41:40	20013	42
2009-06-01 08:45:00	20077	1
2009-06-01 08:51:40	20069	19
2009-06-01 09:05:00	20011	22
2009-06-01 09:08:20	20007	22
2009-06-01 09:11:40	20005	39
2009-06-01 09:18:20	20085	31
2009-06-01 09:21:40	20054	8
2009-06-01 09:28:20	20011	49

- **Tip**

Rough train of thought: Create a table to record the remaining stocks and the out-of-stock commodities first. Group the purchase table and sales table by date and loop every day. Traverse every purchase and sales record to get statistics of out-of-stock commodities and accumulated time of out-of-stock commodities. Then, the total out-of-stock time and the commodity with longest accumulated out-of-stock time can be found out.

1. Create a table sequence to record all dates of the month and align it with the purchase and sales tables. This is to ensure that the purchase table and sales table are aligned completely, so as to avoid the mistake that no commodities had been replenished is shown one day.
2. Align the purchase table and sales table with the date sequence. Two table sequences will be created which are grouped by dates.
3. Create a table sequence A to record the remaining stocks and the out-of-stock commodities. Table sequence A must comprise **Commodity**, **Stock**, **OosTime**, and **TotalOosTime** fields.
4. Set the primary key of this table as Commodity to facilitate the future searches.
5. Write the remaining stocks of last month to table A as the initial stock.
6. Loop the date sequence of this month, that is, loop each day.
7. In the loop body, retrieve the morning purchase data of this day and add it to table A.
8. Find out the commodities whose stock is 0 in table A. The OosTime is the opening time of the supermarket, that is, 8 a.m.
9. Loop all sales records of the day.
10. In the loop body, find out the commodity in table A and count it as stock changes.
11. Check if the Stock value turns to be 0. If so, then it is out of stock on that day. Write down the OosTime as current time in table A.
12. At the end of the loop, select all out-of-stock commodities of the day in table A. The closing time of the supermarket is the end time. Work out the out-of-stock time point and add it to the field of TotalOosTime in table A.
13. The accumulation of table A is over at the end of the date loop. Choose the commodity

with the longest TotalOosTime.

- **Code**

	A	B	C	D	
1	=file("C:\\txt\\Stock.txt").import@t().select(month(Datetime)==6)				**Stock** table of June
2	=file("C:\\txt\\Sale.txt").import@t().select(month(Datetime)==6)				**Sales** table of June
3	=file("C:\\txt\\Storage.txt").import@t().select(month(Date)==5)				Remaining stocks at the end of May
4	=file("C:\\txt\\Commodity.txt").import@t()				**Commodity** table
5	"08:00:00"		"21:30:00"		Business hours of the supermarket
6	=periods@d(date("2009-6-1"), date("2009-6-30"), 1)				Create a sequence with all days of the month
7	=A1.align(A6:~,date(Datetime))				Align the purchase table by dates to get the record grouped by dates.
8	=A2.align(A6:~,date(Datetime))				Align the sales table by dates once again.
9	=A4.new(ID:Commodity,0:Stock,:OosTime,0:TotalOosTime)				Create a table to get statistics of remaining stocks and out-of-stock commodities.
10	>A9.primary@b(Commodity)				Set the primary key of the table.
11	=A3.run(A9.find(Commodity).Stock=Stock)				Write the remaining stocks of May to table A as the initial stocks.
12	for A6	=A7(#A12).run(A9.find(Commodity).run(Stock=Stock+Volume))			Loop every day. Summarize commodity purchase volumes in every morning first.
13		=A9.select(Stock<=0).run(OosTime=string(A12)+" "+A5)			Find out the out-of-stock commodities. The opening hour of the supermarket should be recorded as the start point of out-of-stock time.
14		for A8(#A12)	=A9.find(B14.Commodity)		To make it further, loop every sales record of the day.
15			>C14.run(Stock=Stock-B14.Volume)		Accumulate all stocks changes.
16			if C14.Stock<=0	>C14.OosTime=B14.Datetime	If the commodities run out of stock after transactions, write down the current time as the start point of out-of-stock.
		=A9.select(Stock<=0)			Select all commodities that run out of stock today.
18		>B17.run(TotalOosTime=TotalOosTime+interval@s(OosTime,string(A12)+" "+C5))			Count the out-of-stock time with the closing time of the supermarket as the end point.
19	=A9.maxp(TotalOosTime).Commodity				Select out the commodity with the longest out-of-stock time.

- **Result**

值
20055

10. Average Residence Time before Certain Types of Commodities Sold in the Supermarket

- **Problem**

The tables below are some tables from simplified supermarket stock control system. To investigate the sales status in a supermarket, it is required to compute the average time before the "Fresh" type commodities are sold. The required tables are given as below:

The first table is a commodity list for recording the detailed information of commodities.

ID	Name	Catagory	Price
20001	----	Food	63
20002	----	Drink	87
20003	----	Drink	26
20004	----	Fresh	88
20005	----	Electrical equipment	57
20006	----	Food	62
20007	----	Electrical equipment	63
20008	----	Fresh	18
20009	----	Food	90
20010	----	Vegetables	55
20011	----	Electrical equipment	75
20012	----	Vegetables	20
20013	----	Drink	26

The second table is the purchase table with the purchase records of each piece of commodity. The supermarket replenishes its stock at 5 a.m. every day.

Datetime	Commodity	Volume
2009-06-21 05:00:00	20064	232
2009-06-21 05:00:00	20006	164
2009-06-21 05:00:00	20031	376
2009-06-21 05:00:00	20066	696
2009-06-21 05:00:00	20037	12
2009-06-21 05:00:00	20094	120
2009-06-21 05:00:00	20065	1128
2009-06-21 05:00:00	20097	1044
2009-06-21 05:00:00	20016	100
2009-06-21 05:00:00	20090	648
2009-06-21 05:00:00	20099	900
2009-06-21 05:00:00	20095	776
2009-06-21 05:00:00	20053	792

The third table is a detailed sales table of the supermarket.

Datetime	Commodity	Volume
2009-06-01 08:05:00	20077	28
2009-06-01 08:11:40	20056	47
2009-06-01 08:18:20	20094	34
2009-06-01 08:21:40	20020	19
2009-06-01 08:41:40	20013	42
2009-06-01 08:45:00	20077	1
2009-06-01 08:51:40	20069	19
2009-06-01 09:05:00	20011	22
2009-06-01 09:08:20	20007	22
2009-06-01 09:11:40	20005	39
2009-06-01 09:18:20	20085	31
2009-06-01 09:21:40	20054	8
2009-06-01 09:28:20	20011	49

- **Tip**

Rough train of thought: First, decide the way to work out the "average residence time before sold". The supermarket cannot make a separate record for every piece of commodity, so counting the residence time of every piece of commodity individually cannot work out. According to mathematical conversion, it is known that the "average residence time" is the difference between the average sold-out time and the average purchase time. The problem can be solved by this algorithm. As long as the purchase and sold-out records of specified item has been filtered, it will be much easier to make statistics on the average value of time.

1. Update the **Commodity** field in the purchase table and sales table with corresponding records of the commodity list.
2. Filter the purchase record by category Fresh.
3. Filter the sales record by category Fresh.
4. Count the total purchase volume.
5. The start point should be the beginning of the month. Work out the time difference between the purchase time and the start point, and multiply it with the purchase volume.
6. Count the total sales volume.
7. The start point should be the beginning of the month. Work out the time difference of the sales time and the start point, multiply it with the sales volume, and sum up.
8. For the remaining commodities (the unsold commodities), set their "sales time" as the end of the month.
9. The average sales time minus the average purchase time is the average residence time.
10. Finally, adjust the result format, convert the unit of time to "days", and reject excessive decimal numbers to get the result.

- **Code**

	A	
1	=file("C:\\txt\\Stock.txt").import@t().select(month(Datetime)==6).dup@t()	Stock table of June
2	=file("C:\\txt\\Sale.txt").import@t().select(month(Datetime)==6).dup@t()	Sales table of June
3	=file("C:\\txt\\Commodity.txt").import@t()	Commodity list

4	Fresh	Specified category
5	>A1.primary(Commodity), A2.primary(Commodity), A3.primary(ID)	Set the primary key of the 3 tables
6	>A1.switch(Commodity,A3), A2.switch(Commodity,A3)	Update the fields of Commodity in the purchase table and sales table with corresponding records of the commodity list.
7	=A1.select(Commodity.Category==A4)	Filter out the purchase record of specified category
8	=A2.select(Commodity.Category==A4)	Filter out the sales record of specified category
9	=A7.sum(Volume)	Sum up the total purchase volume.
10	=A7.sum(long(interval@s("2009-6-1 00:00:00",Datetime)*Volume))	Work out the total time difference between purchase time and starting point. The starting point is the beginning of the month.
11	=A8.sum(Volume)	Sum up the total sales value.
12	=A8.sum(long(interval@s("2009-6-1 00:00:00",Datetime)*Volume))	Work out the total time differences between sales time point and start time point. The start time point should be the beginning of the month.
13	=long((A9-A11)*interval@s("2009-6-1 00:00:00","2009-7-1 00:00:00"))	For the remaining commodities, the "sales time" can be set as the end of the month.
14	=(A12+A13)/A11-A10/A9	The average sales time minus the average purchase time is the average residence time.
15	=floor(A14/60/60/24,1)	Change the unit of time to days and reject excessive decimal numbers for easier viewing.

- **Result**

值
3.5

11. Salesperson Achieving the Highest Sales Values during Promotion

- **Problem**

This is a database problem that occurs in a department store. A database of the department store contains two tables, namely, the **Promotion** table and the **SalesRecord** table.

The **Promotion** table is a calendar for promotion:

promo_name	start_date	end_date
Feast of St. Fred	1995-02-01	1995-02-07
National Pickle Pageant	1995-11-01	1995-11-07
Christmas Week	1995-12-18	1995-12-25

The **SalesRecord** table is used to list the sales for salesperson in a year:

clerk_name	sale_date	sale_amt
Johnson	1995-01-01	349
Larry	1995-01-01	3432
Dow	1995-01-01	4695
Jenny	1995-01-01	2575
Baker	1995-01-01	2310
Mary	1995-01-01	279
Tom	1995-01-01	2551
Steven	1995-01-01	3963
Harry	1995-01-01	3328
Bill	1995-01-01	16
Johnson	1995-01-02	2362
Larry	1995-01-02	4354
Dow	1995-01-02	4856

Please select the salesperson who achieved the highest sales value in each promotion, so as to reward him/her the performance bonus.

- **Tip**

Rough train of thought: Loop the Promotion table. In each loop, firstly select all the sales records about the current promotion from the SalesRecord table; secondly, group these records by sales_name and calculate the total sales_value so as to search for the salesperson achieving the highest sales value; thirdly, record the corresponding sales_name value in the new table sequence, and then this is the result.

1. Create a result table sequence before loops begin;
2. Loop through the records in Promotion table;
3. In each loop body, select all sales records about the current promotion from the SalesRecord table;
4. Group the results by sales_name, count the total sale_ value for each salesperson, and create a new table sequence;
5. Select out the sales record in which the salesperson having achieved the highest sale_ value;
6. Write the names of sales_name and the current promo_name to a result table sequence outside the loop body;
7. The loop is ended, and the details is the enemy just the answer.nce.

- **Code**

	A	B	
1	=file("C:\\txt\\Promotion.txt").import@t()		The Promotion table
2	=file("C:\\txt\\SalesRecord.txt").import@t()		The SalesRecord table
3	=create(promo_name,best_sale)		The result table sequence
4	for A1		Loop the Promotion table
5		=A2.select(sale_date>=A4.start_date && sale_date<=A4.end_date)	Select all the sales records about the current promotion from the SalesRecord table
6		=B5.group@i(clerk_name;~.sum(sale_amt):total_amt)	Group the records by clerk_name and count the total sale_amt for each clerk

7		=B6.maxp(total_amt)	Search for the clerk who won the highest sale_amt
8		>A3.insert(0,A1.promo_name,B7.clerk_name)	Store the clerk_name and the current promo_name in the result table sequence
9	=A3		Answer

- **Result**

promo_name	best_sale
Feast of St. Fred	Larry
National Pickle Pageant	Steven
Christmas Week	Dow

12. Customers who Ordered All Types of Products

- **Problem**

There are some database tables on the order status as given below.

- A **Customers** table which records **customer IDs** and account balances of customers:

customer_id	acct_balance
1	9844
2	8301
3	6851
4	4076
5	4641
6	6416
7	5905
8	8106
9	7397
10	7355

- The **Orders** table **records IDs** of orders and **corresponding IDs** of customers:

customer_id	order_id
10	20000
1	20001
4	20002
7	20003
8	20004
9	20005
1	20006
1	20007
5	20008
7	20009
5	20010
4	20011
3	20012

- The **OrderDetails** table records the detailed information about each order, including the **product IDs** and order quantities. For details, please refer to the below table:

order_id	item_id	item_qty
20000	1022	9
20001	1011	5
20002	1023	5
20003	1014	10
20004	1025	7
20005	1021	5
20006	1009	8
20007	1027	9
20008	1011	8
20009	1017	4
20010	1005	10
20011	1002	4
20012	1002	10

- The **Products** table records all product information of the enterprise:

item_id	item_qty_on_hand
1000	564
1001	419
1002	506
1003	588
1004	503
1005	416
1006	106
1007	572
1008	331
1009	390
1010	557
1011	527
1012	285

Please calculate the average account balance (acct_balance) for the customers who have ordered all types of products of the company as well as for other customers.

- **Tip**

Rough train of thought: Firstly, materialize the association between tables by taking advantage of an esProc features - the pointer reference. In other words, update such fields as **item_id**, **order_id** and **customer_id** with the corresponding records, so that from any record in the resulting **Orders** table, you can directly reference and retrieve any records in the other associated tables. Secondly, group the **Orders** table by **customer_id**. Thirdly, count the purchase quantities of all types of products without duplicate by every customer individually and compare the respective result with the count of all types of products listed in the **Product** table so as to search for the customers with the counts of the types of products purchased by them equaling to the number of records in the

Products table. Those customers qualified are the ones who purchased all types of products; the rest shall be those who didn't purchase all types of products. Finally, calculate the average account balances for oups of customers respectively.

1. Set *_id field as the primary keys for the **OrderDetails**, **Customers**, and **Products** tables respectively;
2. Call the switch function to update the **order_id** field value of the **Orders** table with the corresponding records in the **OrderDetails** table, update the **customer_id** field value of the **Orders** table with the corresponding records in the **Customers** table, and update the **item_id** field value of the **OrderDetails** table with the corresponding records in the **Products** table.
3. Group the **Orders** table by **customer_id**, count the types of products purchased by customers respectively, and create a new table sequence.
4. Select the **customers** with the counts of the types of products purchased by them equaling to the number of records in the Products table;
5. Calculate the average account balance for these customers.
6. Calculate the average account balance for other customers.

- **Code**

	A	
1	=file("C:\\txt\\Orders.txt").import@t()	The **Orders** table
2	=file("C:\\txt\\OrderDetails.txt").import@t()	The **OrderDetails** table
3	=file("C:\\txt\\Customers.txt").import@t()	The **Customers** table
4	=file("C:\\txt\\Products2.txt").import@t()	The **Products** table
5	>A3.primary(customer_id), A2.primary(order_id), A4.primary(item_id)	Set primary keys for tables
6	>A1.switch(order_id,A2; customer_id,A3), A2.switch(item_id,A4)	Replace the ID fields in the **Orders** table and the **OrderDetails** table with the corresponding records
7	=A1.group@i(customer_id:customer;~:orders,~.id(order_id.item_id).len():items_cnt, ~.id(order_id.item_id):t)	Group the **Orders** table by **customer_id**, and count the types of products purchased by every customer
8	=A7.select(items_cnt==A4.len())	Select the customers with the counts of the types of products purchased by them equaling to the number of records in the **Products** table
9	=A8.avg(customer.acct_balance)	The average account balances of these customers
10	=(A7\A8).avg(customer.acct_balance)	The average account balances of other customers

- **Result**

The average account balances (acct_balance) for the customers who have ordered all types of products from the company:

值
4990.5

The average acct_balances for the customers who have not ordered all products of the company:

值
7363.875

13. Employees who has Holidayed for over 2 Days for Consecutive 3 Times in a Year

- **Problem**

The **LeaveRecords** table is used to record the leaves of employees in 2009, including the start date and end date for each leave of every employee.

Employee	StartDate	EndDate	Remarks
22	2009-01-04	2009-01-08	
10	2009-01-04	2009-01-05	
4	2009-01-04	2009-01-06	
9	2009-01-07	2009-01-08	
3	2009-01-19	2009-01-21	
5	2009-01-20	2009-01-21	
5	2009-01-26	2009-01-29	
8	2009-01-28	2009-01-30	
2	2009-01-31	2009-02-04	
14	2009-01-31	2009-02-01	
21	2009-02-02	2009-02-03	
13	2009-02-03	2009-02-04	
18	2009-02-07	2009-02-11	

The **Employees** table is as below:

ID	Name	Gender	Post	Birthday	AccountNo	BasePay
1	Mike	Female	Sale	1968-12-08	53693689129	5600.0
2	Jake	Male	Vice Presider	1962-02-19	96410767719	2500.0
3	Lucy	Female	Sale	1973-08-30	66524824514	10800.0
4	Andy	Male	Sales Manag	1968-09-19	65002886054	7500.0
5	Jim	Male	Sales Manag	1965-03-04	44138024798	4700.0
6	David	Male	Sale	1967-07-02	86091675703	9300.0
7	Jessica	Male	Sale	1960-05-29	56889966430	2900.0
8	Lily	Female	Inside Sales	1969-01-09	42884466796	3500.0
9	Mary	Female	Sale	1969-07-02	96244795103	7200.0
10	Tiger	Male	General Mani	1970-10-10	25549086420	4300.0
11	Kate	Female	Human Resc	1980-04-05	68398410697	7600.0
12	Al	Male	Sale	1980-03-06	73902984872	7500.0
13	Ben	Male	R&D Leader	1984-09-13	04521217732	10700.0

Please find **employees** who have holidayed for over 2 days for consecutively 3 times within a year.

- **Tip**

Rough train of thoughts: Firstly, compute the duration of each leave for every record in the **LeaveRecords** table respectively. By using the coordinate translation of the sequence, search for the consecutive 3 records with the durations of leave longer than 2 days.

1. Group the **LeaveRecords** table by Employee, and create a new table sequence including the Length field. Make statistics on the duration of each leave and create a table sequence composed of these results.
2. Call the select function to filter out the consecutive 3 records with the Length value greater than 2
3. Retrieve the employee name of the above selected records from the **Employees** table for easier view.

- **Code**

	A	
1	=file("C:\\txt\\LeaveRecords.txt").import@t()	The **LeaveRecords** table
2	=file("C:\\txt\\Employees.txt").import@t()	The **Employees** table
3	=A1.group@i(Employee;~.(interval(StartDate,EndDate)):Length)	Group the **LeaveRecords** table by **Employee**, and calculate the duration of leave for each record
4	=A3.select(Length.count(#>2 && ~>2 && Length(#-1)>2 && Length(#-2)>2)>0)	Select the consecutive three records with the Length values greater than 2
5	=A4.(A2.find(Employee)).(Name)	Replace **ID** with **Name** for easier view

- **Result**

成员
Mike
Ed
Frank

14. Prepare Test Data for Sales Management System

- **Problem**

To prepare the simulation data for a sales management system, you are required to generate 4 tables (**test_Sale** table, **test_Client** table, **test_Product** table, and **test_Contract** table) with the data requirement as given below:

The **salesperson** table of test_Sale is structured as given below:

ID	Name	Education	Area

In the table, there are 50 salesperson (Name and ID), 5 areas (Area) of NorthChina,

SouthChina, CentralChina, Southwest and Northwest, and 4 educational backgrounds (Education) of Senior High School, Associate Degree, Bachelor, and Master.

The client table of **test_Client** is structured as given below:

ID	Name	Contact	Address	Phone

There are 200 client records in the table.

The product table of **test_Product** is structured as given below:

ID	Name	Price

There are records about 30 types of products in the table.

The contract table of **test_Contract** is structured as given below:

ContractNo	Client	Product	Sale	SellDate	Quantity

There are 10000 sales records for the whole year of 2009.

- **Tip**

Rough train of thought: There are two key problems on constructing the test data: first, how to input mass of data, second, how to guarantee that the randomness of data to be inserted. You can use the loop and insert function of esProc to solve the first problem and the rand function to solve the second problem.

1. Construct a **test_Sale** table. The **ID** field can be populated with numbers from 1 to 50. The Name field values are just concatenated character strings and numbers. The values of the **Education** and **Area** fields are randomly selected with given options by calling the rand function.
2. Construct a **test_Client** table. The **ID** field value is numbers from 1 to 50; the values of the **Name** field are generated randomly; the values of the **Contact** and **Address** fields are not important so you can just use multiple "-" as values. Similarly, the values of the Phone field are not important either, and they can be any numbers.
3. Construct a **test_Product** table. The rule for **ID** and **Name** field values are similar to the above-mentioned two tables. The value of the Price field is generated by calling the rand function, taking 1000 as the lower limit so as to increase the sense of reality.
4. Construct a **test_Contract** table. The values of the **ID** field are serial numbers. The values of the Client, Product, and Sales fields are picked randomly with the rand function from the above tables generated along with the ID field values retrieved. The values of the **SellDate** field can be generated with the **relDate** and rand functions to guarantee that the values are for the year of 2009. The values of the Quantity field are randomly picked out within a certain range. In this case, they

are picked from 1 to 5.

5. Create the corresponding table in the database, and write the result table sequence into the database.

- **Code**

	A	B	C
1	[NorthChina,SouthChina,CentralChina,Southwest,Northwest]		/The Area sequence
2	[Senior High School,Associate Degree,Bachelor,Master]		/The Education sequence
3	50		/The number of salesperson
4	=create(ID,Name,Education ,Area)		/The test_Sale table
5	=A4.insert(0:A3,~,"Salesperson_"+string(#),A1(int(rand()*A1.len()+1)),A2(int(rand()*A2.len()+1)))		/Insert a record about the salesperson
6	200		/The number of clients
7	=create(ID,Name,Contact,Address,Phone)		/The test_Client
8	=A7.insert(0:A6,~,"Client"+string(#),"---","------","87654321")		/The record about a client
9	30		/The quantity of products
10	=create(ID,Name,Price)		/The test_Product table
11	=A10.insert(0:A9,~,"Product"+string(#),int(rand()*90)*100+1000)		/To insert a record about a product
12	1000		/The quantity of contracts
13	=create(ContractNo,Client,Product,Sale,SellDate,Quantity)		/The test_Contract table
14	for A12		/Insert 1000 records in a loop way
15		=A7(int(rand()*A7.len()+1)).ID	/Randomly get the ID of a client
16		=A10(int(rand()*A10.len()+1)).ID	/Randomly get the ID of a product
17		=A4(int(rand()*A4.len()+1)).ID	/Randomly get the ID of a salesperson
18		=relDate("2009-1-1",int(rand()*365)+1)	/Randomly generate a date within the year of 2009
19		=A13.insert(0,A14,B15,B16,B17,B18,int(rand()*5)+1)	/Insert a record about a contract
20			
21	/Insert the table sequences generated into the txt		
22	=file("C:\\test_Sale.txt").export@t(A4)		
23	=file("C:\\test_Client.txt").export@t(A7)		
24	=file("C:\\test_Product.txt").export@t(A10)		
25	=file("C:\\test_Contract.txt").export@t(A13)		
26	=file("C:\\test_Sale.txt").import@t()		/Browse the test_Sale table
27	=file("C:\\test_Client.txt").import@t()		/Browse the test_Client table
28	=file("C:\\test_Product.txt").import@t()		/ Browse the test_Product table
29	=file("C:\\test_Contract.txt").import@t()		/ Browse the test_Contract table

- **Result**

(Since data are picked randomly, the data generated after each run may vary.)

The test_Sale table is as below:

ID	Name	Education	Area
1	Salesman_1	Southwest	Bachelor
2	Salesman_2	SouthChina	Senior High School
3	Salesman_3	NorthChina	Master
4	Salesman_4	Southwest	Associate Degree
5	Salesman_5	SouthChina	Senior High School
6	Salesman_6	Northwest	Bachelor
7	Salesman_7	NorthChina	Associate Degree
8	Salesman_8	Southwest	Senior High School
9	Salesman_9	NorthChina	Master
10	Salesman_10	Northwest	Bachelor
11	Salesman_11	CentralChina	Master
12	Salesman_12	NorthChina	Master
13	Salesman_13	Northwest	Associate Degree

The test_Client table is as below:

ID	Name	Contact	Address	Phone
1	Client1	---	------	87654321
2	Client2	---	------	87654321
3	Client3	---	------	87654321
4	Client4	---	------	87654321
5	Client5	---	------	87654321
6	Client6	---	------	87654321
7	Client7	---	------	87654321
8	Client8	---	------	87654321
9	Client9	---	------	87654321
10	Client10	---	------	87654321
11	Client11	---	------	87654321
12	Client12	---	------	87654321
13	Client13	---	------	87654321

The test_Product table is as below:

ID	Name	Price
1	Product1	4300.0
2	Product2	3400.0
3	Product3	1000.0
4	Product4	9300.0
5	Product5	7000.0
6	Product6	9700.0
7	Product7	4700.0
8	Product8	1000.0
9	Product9	4600.0
10	Product10	6600.0
11	Product11	3600.0
12	Product12	6500.0
13	Product13	6700.0

The test_Contract table is as below:

ContractNo	Client	Product	Sale	SellDate	Quantity
1	180	14	24	2009-03-31	3
2	22	14	49	2009-11-03	3
3	200	7	34	2009-12-28	2
4	92	1	41	2009-03-29	1
5	25	7	34	2009-04-24	4
6	70	9	30	2009-03-08	3
7	162	23	49	2009-05-14	2
8	154	9	48	2009-01-30	5
9	78	16	26	2009-05-14	4
10	158	17	31	2009-03-13	4
11	103	20	45	2009-04-13	1
12	131	22	28	2009-06-02	4
13	199	29	18	2009-08-01	2

15. Prepare the Olympic Games Gold Medalist

- **Problem**

The table below records a game data of a certain Olympic Games.

The **Country** table is as given below:

ID	Country
1	China
2	Egypt
3	Ethiopia
4	Brazil
5	Belgium
6	Germany
7	Russia
8	France
9	Finland
10	Cuba
11	Korea
12	Netherlands
13	Canada

The **Athlete** table records the country of each athlete, as given below:

Athlete	Country
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
14	1

The **MatchResult** table records the results of matches, as given below:

Event	Ranking	Athlete
1	1	516
1	2	748
1	3	10
1	4	928
1	5	913
1	6	431
1	7	266
1	8	139
1	9	2
1	10	181
1	11	290
1	12	302
1	13	430

Please make the gold medal list for the current Olympics Games, including the numbers of gold medals, silver medals, and bronze medals of each country/district respectively; sort the list by the number of gold medals and display the ranking. Please note that two or more countries may be in a tie in placing.

- **Tip**

Rough train of thought: First, associate Ranking with Country by replacing the values of the Country field of the **Athlete** table with the corresponding records in the **Country** table and then replacing the values of the Athlete field of the **MatchResult** table with corresponding records in the **Athlete** table; then, group the **MatchResult** table by Country; third, count the numbers of gold medals, silver medals, and bronze medals won by countries respectively; fourth, sort the sequence composed of three variables (keywords): the number of gold medals, the number of silver medals, and the number of bronze medals. Maybe you should use the ranki function to solve the problem that two or more countries may be tied each other for the same places.

1. Replace the values of the Country field of the **Athlete** table with the corresponding

records in the **Country** table, and the values of the Athlete field of the **MatchResult** table with corresponding records in the **Athlete** table;

2. Group the **MatchResult** table by Country;

3. Create a new table sequence based on the sequence formed after grouping. The table sequence shall include countries names, and the numbers of gold medals, silver medals, and bronze medals won by countries respectively, and preserve a "Ranking" field. Because two or more countries may be tied each other for the same places, the values of the Ranking field shall be preserved.

4. Sort the sequence composed of three variables (keywords): the number of gold medals, the number of silver medals, and the number of bronze medals; run the ranki function to fill the Ranking field.

5. Sort the table by Ranking.

- **Code**

	A	
1	=file("C:\\txt\\Country.txt").import@t()	The **Country** table
2	=file("C:\\txt\\Athlete.txt").import@t()	The **Athlete** table
3	=file("C:\\txt\\MatchResult.txt").import@t()	The **MatchResult** table
4	>A2.switch(Country,A1), A3.switch(Athlete,A2)	Replace the specified fields in the **Athlete** table and the **MatchResult** table with the corresponding records
5	=A3.group(Athlete.Country)	Group athletes by Country
6	=A5.new(:Ranking, Athlete.Country.Country:Country, ~.count(Ranking==1):Gold,~.count(Ranking==2):Silver, ~.count(Ranking==3):Bronze)	Create a new table sequence, calculate the numbers of golden medals, silver medals, and bronze medals won by each country, pick out the country names, and preserve a ranking field with the value Null.
7	=A6.([Gold,Silver,Bronze])	List the sequence composed of numbers of gold medals, silver medals, and bronze medals won by countries respectively
8	>A6.run(Ranking=A7.ranki([Gold,Silver,Bronze]))	Rank countries according to the above sequence
9	=A6.sort(Ranking)	Sort the table by Ranking

- **Result**

Ranking	Country	Gold	Silver	Bronze
1	USA	19	18	19
2	Russia	15	19	16
3	Netherlands	15	8	12
4	Brazil	11	8	5
5	Germany	8	5	7
6	Egypt	7	11	14
7	Ethiopia	7	9	7
8	Cuba	7	4	2
9	Finland	4	5	3
10	France	4	4	4
11	Canada	2	2	4
12	Belgium	1	3	4
13	China	0	3	1

16. Compute Salary Payable based on Attendance and Performance Data

- **Problem**

The finance department of an enterprise needs to compute the staff salary and export the data to the bank. To compute the salary, firstly, make a reference to the employee absenteeism deduction and performance bonus. The detail rule is as follows:

Basic salary * (1- absenteeism + performance)

Second, deduct the personal income tax from the salary payable. The computation method of personal income tax is a bit complicated: deduct 2000 dollars of tax free part from the income, and the remaining part is the tax base for which the tax will be calculated according to the tax bracket. The tax rate of each period is as given below:

Levels	Containing tax grade difference	Tax rate(%)	Tax reduction factor
1	below 500	5	0
2	500-2000	10	25
3	2000-5000	15	125
4	5000-20000	20	375
5	20000-40000	25	1375

For example, suppose an employee's salary is 6000 dollars. Subtracting 2000 dollars, we will have 4000 dollars as the tax base to tax according to its tax bracket. The taxable amount is:

500*5%+1500*10%+2000*15%=475 dollars

You can also use the tax reduction factor to compute:

4000*15%-125=475 dollars

The table below is the employee information table of the enterprise:

ID	Name	Gender	Post	Birthday	AccountNo	BasePay
1	Mike	Female	Sale	1968-12-08	53693689129	5600.0
2	Jake	Male	Vice Presider	1962-02-19	96410767719	2500.0
3	Lucy	Female	Sale	1973-08-30	66524824514	10800.0
4	Andy	Male	Sales Manag	1968-09-19	65002886054	7500.0
5	Jim	Male	Sales Manag	1965-03-04	44138024798	4700.0
6	David	Male	Sale	1967-07-02	86091675703	9300.0
7	Jessica	Male	Sale	1960-05-29	56889966430	2900.0
8	Lily	Female	Inside Sales	1969-01-09	42884466796	3500.0
9	Mary	Female	Sale	1969-07-02	96244795103	7200.0
10	Tiger	Male	General Man	1970-10-10	25549086420	4300.0
11	Kate	Female	Human Resc	1980-04-05	68398410697	7600.0
12	Al	Male	Sale	1980-03-06	73902984872	7500.0
13	Ben	Male	R&D Leader	1984-09-13	04521217732	10700.0

The table below is the **Absenteeism** table of this month. If an **employee** was not absent in this month, then no record of the **employee** will be found in this table.

Employee	Absenteeism
1	0.13
2	0.13
5	0.06
6	0.13
7	0.13
8	0.03
9	0.33
12	0.2
14	0.13
15	0.13
16	0.33
17	0.3
20	0.26

The table below is the **Performance** table of employees in this month. Similarly, not all employees have its **performance** record in this table.

Employee	Performance
1	0.01
2	0.01
3	0.14
4	0.15
5	0.05
6	0.06
7	0.14
8	0.04
9	0.08
10	0.06
11	0.13
12	0.07
13	0.14

Please compute the salary amount payable for every employee, and export the result as a TXT file for the bank. The format of TXT file should be:

Company name
TOAM= *Total payroll*
COUT= *Number of employee*

Bank account | *Salary payable* | *Name*
Bank account | *Salary payable* | *Name*
……

- **Tip**

1. Create a new table sequence based on the **Employee** table and keep the Name, BasePay,

and AccountNo fields. Add the **Absenteeism**, **Performance**, and SalaryPayable fields, in which the value of **Absenteeism** field is the corresponding records from the **Absenteeism** table, and so is **Performance** field. As for the SalaryPayable field, write 0 to it.

2. Compute the salary payable for every employee with formula, and write it to the SalaryPayable fields. As for employees without absence or **performance** bonus record, the **Absenteeism** or **Performance** field will be empty and be taken as 0 during processing, which meets our requirement.
3. Copy the TaxRate table to the code cellset and convert it to a table sequence to facilitate the computation.
4. Compute the personal income tax for every employee. Firstly, find the salary bracket to which the employee belongs; secondly, deduct the tax reduction factor from the tax base directly to get the tax amount; finally, subtract it from the salary.
5. Load the TXT file. Firstly, create a TXT file object, then write the company name, salary payable in total, and number of persons to be transferred.
6. Cycle the above table and write the record to the TXT file one by one in the required format.

- **Code**

	A	B	C	
1	=file("C:\\txt\\Employees.txt").import@t()			**Employee** table
2	=file("C:\\txt\\Absenteeism.txt").import@t()			**Absenteeism** table
3	=file("C:\\txt\\Performance.txt").import@t()			**Performance** table
4	/Salary	/Tax Rate(%)	/Tax Reduction Factor	Tax rate table
5	500	5	0	
6	2000	10	25	
7	5000	15	125	
8	20000	20	375	
9	40000	25	1375	
10	D:\Bank Pay File.txt			Path to export TXT file
11	=A1.new(Name,A2.select@1(Employee==ID).Absenteeism:Absenteeism,A3.select@1(Employee==ID).Performance:Performance,BasePay,AccountNo,0:SalaryPayable)			Based on the **Employee** table, newly create the table sequence of the new field structure and select the **absenteeism** and attendance data
12	>A11.run(SalaryPayable=BasePay*(1-Absenteeism+Performance))			Compute the salary payable of each employee
13	=create(Salary,TaxRate,TRF).record([A5:C9],0)			Convert the TaxRate table to the table sequence
14	for A11	=A13.select@1(Salary>=A14.SalaryPayable)		Cycle the Salary table and find the tax bracket for each employee
15		>A14.SalaryPayable=round(A14.SalaryPayable-(A14.SalaryPayable*B14.TaxRate/100-B14.TRF),2)		Compute the tax for each employee and deduct it from the salary payable
16	=file(A10)			Create the file object
17	>A16.write("RAQSOFT INCORPORATION")			Write the company name
18	>A16.write@a("\r\nTOAM="+string(round(A11.sum(SalaryPayable),2))			Write the total salary payable continuously
19	>A16.write@a("\r\nCOUT="+string(A11.count()))			Write the total persons to transfer
20	>A16.write@a("\r\n----------"			Write the separator line

	)	
21	>A11.run(A16.write@a("\r\n"+string(AccountNo)+" | "+string(SalaryPayable)+" | "+Name))	Respectively write the account, salary and name of each **employee**

- **Result**

```
RAQSOFT INCORPORATION
TOAM=122670.25
COUT=23
--------------------------------------
5369368912986521 | 4313.8 | Mike
9641076771943034 | 1995.0 | Jake
6652482451405780 | 10224.6 | Lucy
6500288605409660 | 7275.0 | Andy
4413802479802512 | 4080.05 | Jim
8609167570309329 | 7294.2 | David
5688996643039205 | 2614.65 | Jessica
4288446679630632 | 3129.75 | Lily
9624479510397273 | 4695.0 | Mary
```

17. Output the Data as Column Layout for Report Display

- **Problem**

This example can be regarded as a request for report processing. With this example, the flexible computation mechanism of esProc can be fully demonstrated. esProc is even capable of handling the request on handling the reports of irregular layout.

The problem is pretty simple: to change the layout of the below Names table to display in 3 columns, namely, the **name1**, **name2**, and **name3** columns. It is also expected that the code can also enable us to display the table in columns of any number if possible.

name
Al
Ben
Charlie
David
Ed
Frank
Greg
Howard
Ida
Joe
Ken
Larry
Mike

- **Tip**

Rough Train of Thought: There is a record function in esProc that can just meet the need, as long as you complete the Name column by appending null records till it is the integral multiple of the target fields.

1. Complete the Name column by appending null records till its number is the integral multiple of the target fields.
2. With the record function, write the Name sequence to the new table directly.

- Code

	A	
1	=file("C:\\txt\\Names.txt").import@t()	**Names** table
2	=create(name1,name2,name3)	Create the multi-column target table
3	3	number of fields of the target table
4	=A1.(name)|(A4-A1.len()%A4)*[null]	Append the name table sequences of integral multiple of target columns
5	=A2.record@i(A4,0)	Use record function to update the new table

In these codes, if you want to implement the display of more columns, you can simply write the desired display structure in cell A3, and write the number of fields for this structure in column A4. Re-compute the cellset, and the name will be populated according to the new structure.

- **Result**

name1	name2	name3
Al	Ben	Charlie
David	Ed	Frank
Greg	Howard	Ida
Joe	Ken	Larry
Mike		

18. Find the Consecutive Vacant Seat in a Restaurant

- **Problem**

In a restaurant capable of holding 1000 people, the seats are numbered in order and recorded in a database table. For any customer being already seated, the corresponding seat will be marked respectively with a tag in the **Seat** table, and the tag will be removed once the customer leaves.

SeatNo	Availability
1	true
2	true
3	true
4	true
5	false
6	false
7	false
8	false
9	false
10	false
11	true
12	true
13	false

Because it frequently happens that some customers patronize in company, in order to find the immediate neighboring seats for them, the restaurant administer needs a Vacant Seat Sections table to synchronously log the sections of vacant seats and the location of these sections.

Please help the restaurant administer out with the Vacant Seat Sections table generated from the Seat table above.

- **Tip**

Rough Train of Thought: By incorporating its optional parameters with no requirement on order, the Group function of esProc can not only merge the same contents, but also just merge the same neighboring contents only. Grouping without order requirement can simply solve this issue perfectly.

- **Code**

	A	
1	=file("C:\\txt\\Seatings.txt").import@t()	**Seatings** table
2	=A1.group@io(Availability;~.count():NumberOfSeats,~.SeatNo:StartingSeatNo)	Group by customer existence with no requirement on order
3	=A2.select(Availability)	Select the area of vacant seats

- **Result**

Availability	NumberOfSeats	StartingSeatNo
true	4	1
true	2	11
true	3	23
true	2	34
true	1	40
true	12	46
true	13	59
true	6	85
true	4	95
true	2	100
true	3	103
true	2	109
true	10	114

19. Calculate User Growth Rate under Charge-Renewing Mode

- **Problem**

  An antivirus software company's product is sold according to the time of use, a sales mode in which a user must continually buy "license key" so that he/she can continue using the software. The "license key" includes half-a-year authorization, one-year authorization and lifetime authorization. For details, please see the following data:

ID	Key Type	TermOfValidity	Price
1	Six months	6 months	20
2	One Year	12 months	36
3	Unlimited	Infinity	500

The following is the sales record of this company in the first few years after the adoption of this kind of sales mode:

Date	User	KeyID
2007-01-01 02:46:40	18414	2
2007-01-01 13:53:20	11392	2
2007-01-02 01:00:00	13146	1
2007-01-02 09:20:00	17811	2
2007-01-02 23:13:20	10108	2
2007-01-03 02:00:00	10520	3
2007-01-03 10:20:00	14383	1
2007-01-04 03:00:00	17781	1
2007-01-04 05:46:40	13668	1
2007-01-04 19:40:00	17094	2
2007-01-04 22:26:40	11614	1
2007-01-05 01:13:20	15484	2
2007-01-05 06:46:40	14967	1

If the user fails to continue buying the authorization upon the expiration of the preceding authorization, then company stops the use of the software and no longer deems this user as a user of the company. Please calculate the annual user growth rate of this company in the first few years.

- **Tip**

Rough train of thought: As this question involves the settlement at the end of the year and the year is uncertain, it is necessary to loop it according to day, and at the same time, use another table sequence to synchronously record the current customer. In the loop of each day, delete expired users, add new customers, renew charges for old users, and just settle the number of users at the end of the year.

1. Firstly write out the valid extension time of each kind of key and write them into the table sequence. As the extension time is uncertain (such as, a solar month of 31 days, a solar month of 30 days, common year, and leap year), here we sue **after** function to write down the equation for its calculation method, which can be just invoked directly in subsequent calculations.
2. Firstly make some preparations before the loop, create a table sequence that records the current valid users, and then establish a result table that records the annual number of users and growth rate.
3. Firstly set the scope before and after the loop each day, namely, the date scope of the sales record.
4. Repeat each day within the date scope of the sales record. In the loop body, firstly delete the customers having already exceeded the validity date on the current day.
5. Find out the sales records of the current day.
6. Loop these records and find out the corresponding customers in the customer table. If there is none, create a customer record.
7. According to the key category bought in the sales record, calculate the validity period of the user.
8. At the end of each year, write the number of the current customers into the preceding result table.
9. After the loop is over, use the result table to calculate the growth rate of the number of

customers each day.

- **Code**

	A	B	C	D	
1	=file("C:\\txt\\AVwareSales.txt").import@t().sort(Date)				sales record table
2	/ID	/Pattern			write out the equations of various kinds of Keys for future use.
3	1	after@m(? ,6)			
4	2	after@y(? ,1)			
5	3	"null"			
6	=create(ID,Pattern).record([A3:B5],0)				write **KeyID** and equation into the table sequence.
7	=create(User,TermOfValidity)				create a current user table.
8	=create(Year,NoOfUsers,GrowthRate)				create a result table that records growth rate.
9	=date(A1.Date)		=date(A1.m(-1).Date)		loop the start date and the end date
10	for A1	>A7.delete(A7.select(TermOfValidity!=null && TermOfValidity<A9))			loop each day. Delete users having exceeded the validity date on the current day.
11		=A1.select(date(Date)==A9)			find out the sales record of the current day.
12		for B11	=A7.find(B12.User)		loop these sales records, and find out the corresponding sales record.
13			if C12==null	>A7.insert(0,A10.User,A9)	if there is none, create a record.
14				>C12=A7.m(-1)	
15			>C12.run(TermOfValidity=eval(A6.find(A10.KeyID).Pattern,TermOfValidity))		according to the corresponding equation of key, calculate new validity period.
16		if month(A9)==12 && day(A9)==31			at the end of the year
17			>A8.insert(0,year(A9),A7.count())		write down the number of the current users.
18		>A9=after@d(A9,1)			loop variable increases by one day.
19		if A9>C9	break		if the loop exceeds the date scope of the sales record, just end the loop.
20	=A8.run(if(#>1,GrowthRate=round((NoOfUsers-NoOfUsers[-1])/NoOfUsers[-1],2)))				calculate out the annual customer growth rate.

- **Result**

Year	NoOfUsers	GrowthRate
2007	725	
2008	1264	0.74
2009	1743	0.38

20. Calculate the Rate of Client Churn

- **Problem**

All records about the sales contracts of an enterprise are listed as below:

ContractNo	ActualSale	SellDate	Product	Quantities	Amount	Client	ApplyArea	ApplyMethod
10961	8	1997-03-19	52	6	1122.0	QUEEN	EastChina	SELF_USE
10962	8	1997-03-19	7	45	3584.0	QUICK	SouthChina	SELF_USE
10963	9	1997-03-19	60	2	68.0	FURIB	NorthChina	RESELL
10964	3	1997-03-20	18	6	2052.5	SPECD	SouthChina	RESELL
10965	6	1997-03-20	51	16	848.0	OLDWO	EastChina	SELF_USE
10966	4	1997-03-20	37	8	1255.6	CHOPS	SouthChina	SELF_USE
10967	2	1997-03-23	19	12	910.4	TOMSP	EastChina	RESELL
10968	1	1997-03-23	12	30	1408.0	ERNSH	SouthChina	SELF_USE
10969	1	1997-03-23	46	9	108.0	COMMI	NorthChina	RESELL
10970	9	1997-03-24	52	40	280.0	BOLID	Northeast	SELF_USE
10971	2	1997-03-24	29	14	1733.06	FRANR	EastChina	RESELL
10972	4	1997-03-24	17	6	251.5	LACOR	EastChina	SELF_USE
10973	6	1997-03-24	26	5	291.55	LACOR	EastChina	RESELL
10974	3	1997-03-25	63	10	439.0	SPLIR	NorthChina	SELF_USE

The lost client in a year are the clients whose corresponding sales values in the **Amount** field for previous year are not 0 but those for current year are 0. By dividing the number of lost clients by the total number of clients of the previous year, you can get the rate of client churn in that year. Please count the lost clients in 1998 and calculate the rate of client churn for the year 1998.

Tip

Rough train of thoughts: Because one or more contracts may have been signed with one client, and the one who have not signed any contract shall not be regarded as a client, it is quite effective and efficient to combine the computations on client quantity through the group function, and then the computation of the client churn rate will be much easier.

1. Select all contracts signed in 1997, group these records by client, and list the clients;
2. Similarly, select all contracts signed in 1998, and list the clients;
3. Subtract the client list 1998 from the client list 1997. The result will be the lost client;
4. Divide the number of lost clients by the total clients of 1997. Then, you will get the churn rate of client.

Code

	A	
1	=file("C:\\txt\\Contract.txt").import@t()	The **Contract** table
2	=A1.select(year(SellDate)==1997)	Select the contracts signed in 1997
3	=A2.group@i(Client;)	Group by **Client**, and list clients
4	=A1.select(year(SellDate)==1998)	Select the contracts signed in 1998
5	=A4.group@i(Client;)	Group by **Client**, and list clients
6	=A3.(Client)\A5.(Client)	Remove clients in the list for 1998 from the list for 1997; the result is the lost clients
7	=A6.len()/A3.len()	Calculate the rate of client churn

Result

Lost client

Member
ALFKI
LACOR
ROMEY

Rate of client churn

Value
0.037037037037037035