Homeworks 1-11 and two tasks completed.

```
/* NAME: Ashmitha
SURNAME: Bonaventure
e-mail:  ashmitha.bonaventure@gmail.com
*/

/*
1.1 Create a data set with two numerical variables x and y, with the values from the uniform distribution
on [−1, 1]. Find the arithmetic averages and the maximal values of x and y. Create a data set with the
maxima of the pair x, y in the consecutive observations (the same for the averages).
*/
data ex1a;

avgx = 0;
avgy = 0;
maxx = 0;
maxy = 0;

do i = 0 to 5;
        x = rand('UNIFORM', -1, 1);
        y = rand('UNIFORM', -1, 1);
        output;

        avgx = avgx + x;
        avgy = avgy + y;

        if x > maxx then do;
                maxx = x;
        end;
```

```
        if y > maxy then do;
                maxy = y;
        end;
end;

avgx = avgx / 5;
avgy = avgy / 5;

put 'The average of x is ' avgx= ' and the average of y is ' avgy=;
put 'The max of x is ' maxx= 'and the max of y is ' maxy=;
keep x y;
run;

data ex1b;
retain max_val 0;
set ex1a;
if x > y then max_val = x;
else max_val = y;

avg = (x + y) / 2;

run;



/*
1.2 Basing on the set a write a data step that, for every value of the variable id, will
create x copies of the
observation containing the value (see the set b). */
data ex2;
 set Lab01.a;
 do i = 1 to x;
 output;
 end;
```

```
drop i;
run;


/*
1.3 Classify the values of id from "rank" with respect to decreasing values of x. The same values of x should
have the same value of the classifying variable (created by the programmer). You can assume that rank
is sorted in decreasing order with respect to x. */


data ex3;
set Lab01.rank;
	retain rank 1;
	if _n_ ne 1 then
		do;
			if x < lag(x) then /*lag_i_(x) -> returns ith observation back from the current of x*/
				rank = rank + 1;
		end;
	output;
run;



/*
1.4 Basing on cb find the numbers of increases and decreases of cb. */
data ex4;
set Lab01.cb;
retain inc 0;
retain dec 0;
if _n_ ne 1 then do;
	if cb > lag(cb) then inc = inc + 1;
	else dec = dec + 1;
end;
run;
```

```
/*
```

1.5 The data set zbior is given. Write data steps that will generate in the Log window the same messages as

in the textfiles k1.txt, k2.txt and k3.txt. (The variables v3 and v4 are the sums of v1 and v2; you cannot

directly assign them missing values. You can use statements SET, RETAIN and PUT.) */

```
data _NULL_; *k1;
set Lab01.zbior end=done;
if done then do;
        temp1 = v1;
        temp2 = v2;
end;

v1 = lag(v1);
v2 = lag (v2);
v3 = v1 + v2;

put v1= v2= v3=;
if _n_ ne 1 then do;
        put v1= v2= v3=;
end;

if done then do;
        v1 = temp1;
        v2 = temp2;
        v3 = v1 + v2;
        put v1= v2= v3=;
        put v1= v2= v3=;
end;

run;
```

```
/*
1.6 Basing on the set cb find the number of local maxima of cb. */
data _null_;
        set Lab01.cb end = done;
        retain maxima 0;
        if lag2(cb) < lag(cb) and  lag(cb) > cb  then do; *lag(cb) is found to be a local
maxima;
                maxima= maxima + 1;
                end;
        if done then
                put maxima=;
run;
```

```
/*
1.7 The data set brak has the numerical variable x with some isolated missing values
(additionally we assume
that the first and the last observations are non-missing). Fill in the places with the missing
values with
the arithmetical averages of the previous and next observations. */

data ex7;
set lab01.brak;
retain found 0;

if found = 1 then do;
x = sum( lag1(x) , lag2(x) ) / 2;
found = 0;
end;
if x = . then do;
found = 1;
end;
run;
```

```
data ex7;
	set lab01.brak end=done;

	data1 = lag(x);
	data2 = lag2(x);

	if data1 = . then do;
	    newX = (data2 + x)/2;
	end;
	else do;
	    newX = data1; *because our table is shifted, so current newx is actually
previous one;
	end;
	drop data1 data2;

	tmpX=x;
	x = lag(x);
	if _N_ > 1 then output; *important! we always calculate for PREVIOUS x, but our
prints are shifted;
	if done then do;
	    x = tmpX;
	    newX = x;
	    output;
	end;
	drop tmpX;
run;

/*
1.8 Interweave the values from bezkropek with a random number of missing values (for an
example, see
kropki). */

data ex8;
set LAB01.bezkropek;
```

```
randx = int(ranuni(0) * 2);
randy = int(ranuni(0) * 2);


if randx = 1 then do;
        tmp = int(ranuni(0) * 2 + 1); *random numbers from 1 - 3;
        do i = 1 to tmp;
                if i ne 1 then  x = .;
                if randy = 1 then y = .;
                output;
        end;
end;
else if randy = 1 then do;
        tmp = int(ranuni(0) * 2 + 1); *random numbers from 1 - 3;
        do i = 1 to tmp;
                if i ne 1 then y = .;
                if randx = 1 then x = .;
                output;
        end;
end;
else if randx = 0 and randy = 0 then output;
keep x y randx randy;
run;
```

```
/*
2.1 Create a data set structured like grupy (we assume that the created set is sorted with
respect to the
grouping variable x). Find the average of y for every value of x
(a) not using BY;


(b) using BY. */
```

```
data ex1;
  do x = 'A', 'B', 'C', 'D', 'E';
        n = floor(ranuni(0) * 5 + 3);
        do i = 1 to n;
          y = ceil(ranuni(0)*20);
          output;
        end;
  end;
  drop i n;
run;

data ex1a;
retain sum 0;
retain counter 0;
set ex1;
        if x = lag(x) or _n_ = 1 then do;
                counter = counter + 1;
                sum = sum+ y;
        end;
        else do;
                if _n_ > 1 then do;
                        group = lag(x);
                        avg = sum / counter;
                        output;
                        sum = 0;
                        counter = 0;
                end;
        end;

keep group avg sum counter;
run;
```

```
data ex1b;
retain sum 0;
retain counter 0;
set ex1;
by x;
counter = counter + 1;
sum = sum+ y;
if last.x then do;
	group = lag(x);
	avg = sum / counter;
	output;
	sum = 0;
	counter = 0;
end;

keep group avg sum counter;

run;
/*
```

2.2 Generate 100 observations from the uniform distribution U(0, 100). Divide the observations into five groups:

the first group should contain the valuesfrom the interval (0,20), the second – from (20,40) etc. (Clearly,

the numbers of observations in each group are random.) Find the maximal values in each group. */

```
data ex2;
	do i = 1 to 100;
		x = rand('UNIFORM', 0, 100);
		output;
	end;
	keep x;
run;
```

```
proc sort data=ex2 out=ex2;
by x;
run;

data ex2group;
set ex2;
retain group 1;

if x > group * 20 then group = group + 1;

run;

data ex2max;
retain max 0;
set ex2group;
by group;

if x > max then max = x;

if last.group then do;
output;
max = 0;
end;

run;
/*
```

2.3 The data set b has only the numerical observation x and is sorted. Create a grouping variable to divide

the set into groups such that all the groups, perhaps without the last one, will contain 10 different values

of x. Next, for every group find the maximum of the most frequent values. */

```
data ex3;
```

```
retain group 1;
set Lab02.b;
array counters(200) _temporary_;
if x ne lag(x) then do;
        group = 1;
        do while ( max (counters(group), 10) = counters(group) );
         group = group + 1;
        end;
end;
else do;
        counters(group) = sum( counters(group), 1);
        group = group + 1;
end;
run;

proc sort data=ex3 out=ex3;
by group;
run;

/*
2.4 Pick such u from a that appear in at least two groups formed by x. */

proc sort data = Lab02.a out = Work.a nodupkey;
        by x u;
run;

data ex4;
array counters(10) _temporary_ (10 * 0);
retain chosen;
set a end=done;
counters(u + 1) = counters(u + 1) + 1;

if done then do;
```

```
        do i = 1 to dim(counters);
        if counters(i) > 1 then do;
                picked = i - 1;
                put picked=;
        end;
end; end;

keep x u;
run;

/*
2.5 Pick such u from a that appear in the largest number of groups formed by x. */
proc sort data = Lab2.a out = Work.a nodupkey;
        by x u;
run;

data ex5;
array counters(10) _temporary_ (10 * 0);
retain largest 0;
retain maxcounter 0;
set a end=done;
counters(u + 1) = counters(u + 1) + 1;

if done then do;
        do i = 1 to dim(counters);
                if counters(i) > maxcounter then do;
                        maxcounter = counters(i);
                        largest = i;
                end;
        end;
        counter = counters(largest);
        largest = largest - 1;
        put largest= counter=;
```

```
end;


keep x u;
run;


/*
2.6 For every value of x (the data set a) compute the arithmetic average of the first five values of u. */


data ex6;
set Lab02.a;
by x;
retain counter 0;
retain sum 0;
if first.x then do;
counter = 0;
sum = 0;
end;
avg = 0;
group = '';
if counter < 5 then sum = sum + u;
if counter = 4 then do;
        avg = sum;
        group = x;
        output;
end;
counter = counter + 1;
keep group avg;
run;


/*
2.7 For every value of x (the data set a) compute the arithmetic average of the last five values of u. */
data ex7;
```

```
set Lab02.a;
by x;
array values(10) _temporary_ (10*0);
retain counter 1;
values(counter) = u;
counter = counter + 1;

if last.x then do;
        sum = 0;

        do i = (counter - 5) to counter;
                sum = sum + values(i);
        end;
        avg = sum / 5;
        group = x;
        output;
        counter = 1;
end;

keep group avg;
run;

/*
2.8 Pick the values of x from a for which u does not take the values 0 and 9 together. */
data ex8;
retain counter 0;
set Lab02.a;
by x;

if u = 0 or u = 9 then do;
        counter = counter + 1;
end;
put counter= x=;
```

```
if last.x then do;
	if counter =  0 then do;
		group = x;
		output;
	end;
	counter = 0;
end;

keep group;
run;

/*
2.9 Verify if funkcja defines a function on the set of values of x. */

proc sort data=Lab02.funkcja out=ex9;
by x;
run;



data _null_;
set ex9 end=done;

retain check 0;
if x = lag(x) then check = 1;

if done then do;
	if check ne 0 then put 'This is NOT a function.';
	else put 'This is a function.';
end;
run;
```

```
/*
3.1 The SAS data set ankieta contains the answers (A, B or C) given by some 100 people to some 10 questions
of a poll. Create relevant data sets to answer the following questions:
• how many times each person gave the answers A, B, C?
• how many times every answer A, B, C was given in the poll?
• how many times were the answers A, B, C given to each question?*/
data ex1a;
set lab03.ankieta;
array observation(*) pytanie1-pytanie10 _temporary_;
A = 0;
B = 0;
C = 0;

do i = 1 to dim(observation);
        if observation(i) = 'A' then A = A + 1;
        else if observation(i) = 'B' then B = B + 1;
        else if observation(i) = 'C' then C = C + 1;
end;

keep A B C;
run;

data ex1b;
set lab03.ankieta end=done;
array observation(*) pytanie1-pytanie10 _temporary_;
retain A 0;
retain B 0;
retain C 0;

do i = 1 to dim(observation);
        if observation(i) = 'A' then A = A + 1;
        else if observation(i) = 'B' then B = B + 1;
        else if observation(i) = 'C' then C = C + 1;
```

```
end;

if done then output;

keep A B C;
run;



data _null_;
set lab03.ankieta end=done;
array observation(*) pytanie1-pytanie10 _temporary_;
array answers(10, 3) _temporary_;

if _n_ = 1 then do;
        do i = 1 to dim1(answers);
                do j = 1 to dim2(answers);
                answers(i, j) = 0;
                end;
        end;
end;



do i = 1 to dim(observation);

        if observation(i) = 'A' then answers(i, 1) = answers(i, 1) + 1;
        else if observation(i) = 'B' then answers(i, 2) = answers(i, 2) + 1;
        else if observation(i) = 'C' then answers(i, 3) = answers(i, 3) + 1;
end;

if done then do;
        do i = 1 to dim1(answers);
                put 'Question: ' i;
                put 'A ' answers(i,1);
```

```
                put 'B ' answers(i,2);
                put 'C ' answers(i,3);
        end;
end;


run;
/*
3.2 Pick randomly (both with and without replacement) l values of the variables x1, . . . , x10 from every
observation from a. */


/* with replacement */
data ex2a;
        set Lab03.a;
        array t(10) x1-x10;
        array tmp(10) _temporary_;
        do i=1 to dim(tmp);
                tmp(i) = t(i);
                t(i) = .;
        end;
        do i = 1 to l;
                rand = int(ranuni(0) * dim(tmp) + 1); /* random number from 1-10 */
                t(rand) = tmp(rand);
        end;
        drop i rand;
run;


/* without replacement */
*fix!;
data ex2b;
        set Lab03.a;
        array t(10) x1-x10;
        array tmp(10) _temporary_;
        array rands(10) _temporary_;
```

```
	do i=1 to dim(tmp);
		tmp(i) = t(i);
		t(i) = .;
	end;
	do i = 1 to l;
		rand = int(ranuni(0) * dim(tmp) + 1); /* random number from 1-10 */

		check = 0;
		do while (check = 0);
			if rand in rands then;
				rand = int(ranuni(0) * dim(tmp) + 1); /* find a rand that was
not picked yet */
			else check = 1;
		end;

		rands(i) = rand;
		t(rand) = tmp(rand);
	end;
	drop i rand;
run;

/*
3.3 Write a program that generates Pascal's triangle, in which the last row contains the
coefficients
k = 0, 1, . . . , n. */

*trololololo lololo lololo;

/*
3.4 Create a data set z with 100 variables z1, . . . , z100 (with the values from the uniform
distribution U(−1, 1))
and with one observation. Transform z into pz in such way that 10 tens of elements z would
be 10 10 rows
of pz. Next, exchange randomly picked 25 elements of pz with missing values. */
```

```
data z;
array z(100) (100*0);
do i = 1 to 100;
	z(i) = rand("UNIFORM", -1, 1);
end;
drop i;
run;

data pz;
set z;
array observation(*) z1-z100 _TEMPORARY_;
array z (10);
counter = 0;
do i = 1 to dim(z);
	do j = 1 to dim(z);
		z(j) = observation(j + 10 * counter);

		if j = dim(z) then do;
			counter = counter + 1;
			output;
		end;
	end;
end;

keep z1-z10;
run;

data pzm;
set pz end=done;
array all(100) _temporary_;
array observation (*) z: _temporary_;

do i = 1 to 10;
```

```
        all(i + 10 * (_n_ - 1)) = observation(i);
end;

if done then do;
        do i = 1 to 25;
                rand = floor(ranuni(0)*99 + 1);
                all(rand) = .;
        end;
        counter = 0;
        do i = 0 to 9;
                do j = 1 to 10;
                        array final(10);
                        final(j) = all(j + 10 * i);
                end;
                output;
        end;
end;

keep final:;
run;
/*
3.5 Order increasingly the elements of each row from a. */
data ex5;
set Lab03.a;
array observation(*) x:;

do i = 1 to dim(observation);
  do until (sorted);
    sorted=1;
    do i = 1 to dim(observation)-1;
      if observation(i) > observation(i+1) then do;
        temp = observation(i+1);
        observation(i+1) = observation(i);
```

```
        observation(i) = temp;
        sorted=0;
      end;
    end;
  end;
end;

keep x:;
run;

/* TODO
3.6 In the data set konwersja, exchange the variable data into a SAS date, and convert the variable liczba
into a numerical variable using the variable kod. */

data ex6; *todo;
set Lab03.konwersja;
*format data ddyymm10.;
sasdate = input(data, date9.); * MMDDYY6. substr(s,5,2)||substr(s,1, 4));
*tmp = scan(data, '.');
run;

/*
3.7 The rows of data set sysdwa are some numbers written in the binary system. Represent the numbers in
the decimal system. */
data ex7;
        set lab03.sysdwa;
        array a(*) b:;
        number = 0;
        do i=dim(a) to 1 by -1;
                if a(i) ne . then
                        do;
```

```
                        number = number + a(i) * (2 ** (dim(a) - i)); *??????;
                    end;
        end;
        keep number;
        output;
run;


/*fix!!
3.8 Basing on xa create a data set avg (the variable avgx i contains the averages of the observations xi in
groups formed by the variable ai). */
data ex8;
set Lab03.xa end=done;
array charact(*) _CHARACTER_;
array num(*) _NUMERIC_;
array a_val(5)  _temporary_ (5 * 0);
array counter(5) _temporary_ (5 * 0);
array names(5)$  _temporary_ ('A1' 'A2' 'A3' 'A4' 'A5');
array avgx_(5) _temporary_ (5 * 0);

do i = 1 to dim(charact);
        if charact(i) = 'A1' then do;
         a_val(1) = a_val(1) + num(i);
         counter(1) = counter(1) + 1;
        end;
        else if charact(i) = 'A2' then do;
         a_val(2) = a_val(2) + num(i);
         counter(2) = counter(2) + 1;
        end;
        else if charact(i) = 'A3' then do;
         a_val(3) = a_val(3) + num(i);
         counter(3) = counter(3) + 1;
        end;
        else if charact(i) = 'A4' then do;
```

```
        a_val(4) = a_val(4) + num(i);
        counter(4) = counter(4) + 1;
      end;
      else if charact(i) = 'A5' then do;
        a_val(5) = a_val(5) + num(i);
        counter(5) = counter(5) + 1;
      end;
end;

if done then do;
      a = 'A1';
      do i = 1 to 5;
            a = names(i);
            put a= names(i)=;
            /*avgx_1 = a_val(1)/counter(1);
            avgx_2 = a_val(2)/counter(2);
            avgx_3 = a_val(3)/counter(3);
            avgx_4 = a_val(4)/counter(4);
            avgx_5 = a_val(5)/counter(5);*/
            do j = 1 to 5;
                  avgx_(j) = a_val(j) / counter(j);
                  put a_val(j)= counter(j)=;
            end;
            output;
      end;
end;

keep a avgx_:;
run;


/*
```

3.9 For every observation from the data set a compute the average of three largest values of the variables

x1, . . . , x10. */

```
data ex9;
set ex5;
array observation(*) _all_;

sum = 0;
do i = dim(observation) - 2 to dim(observation);
        sum = sum + observation(i);
end;
avg = sum / 3;
keep avg;
run;
```

/*

3.10 Basing on the data set a1 create a data set a2 with the coordinates of missing values in a1. */

```
data ex10;
set Lab03.a1;
array val(*) _all_;
do i = 1 to dim(val);
        if val(i) = . then do;
        brak = 'w' || put(_n_, 2.) || '.k' || put(i,2.);
        output;
        end;
end;

keep brak;
run;
```

```
/*
4.1 Create a data set with (only) such observations from the data set A that their numbers are listed in the
first row of A. */

data ex1;
start = 1;
set Lab04.A point=start;
array indeces(6) _temporary_;
array k(10) k:;
do i=1 to 6;
	indeces(i) = k(i);
end;

do i=1 to 6;
	pt = indeces(i);
	set Lab04.A point = pt; *important! cannot be indeces(i);
	output;
end;

stop;
drop i;

run;

/*
4.2 The i-th row of the data set drzewo contains the numbers of ascendants of the i-th vertex in a binary
tree. Pick a random path in the tree (starting from the root, i.e. the vertex number 1). */
data _null_;
	if _n_ = 1 then do;
		start = 1;
		set Lab04.Drzewo point = start;
	end;
```

```
	side = floor(ranuni(0) * 2 );
	/* the idea is that set point keeps going to the point until stop or no more
observations. */
	if side  = 1 then do;
		if lewy = . then stop;
		put lewy;
		set Lab04.Drzewo point = lewy;
	end;
	else do;
		if prawy = . then stop;
		put prawy;
		set Lab04.Drzewo point = prawy;
	end;
run;



/*
4.3 Read the text files: p1.txt, p2.txt, p3.txt, p4.txt into data sets. */
filename p1 '/folders/myfolders/SAS/data/lab04/p1.txt'; *FILENAME is a statement;
data Lab04.p1;
 infile p1;
 input id $ name $;
run;

filename p2 '/folders/myfolders/SAS/data/lab04/p2.txt'; *FILENAME is a statement;
data Lab04.p2;
 infile p2;
 input id $ name $;
run;

filename p3 '/folders/myfolders/SAS/data/lab04/p3.txt'; *FILENAME is a statement;
data Lab04.p3;
 infile p3;
```

```
 input id name $ surname $ code $ date_of_birth $;
run;


filename p4 '/folders/myfolders/SAS/data/lab04/p4.txt'; *FILENAME is a statement;
data Lab04.p4;
 infile p4;
 input id name $ surname $ code $ date_of_birth $;
run;


/*
4.4 The text file eksperyment.txt contains some data describing a number of repeating experiments. The
rows finishing with the word START (STOP) denote the beginning (the end) of an experiment. All the
other rows contain some unimportant data from some intermediate phases of experiments. Create a data
set with duration times (counted in days) of consecutive experiments. Find the experiment with the largest
amplitude between the first and the last day of experiment. */


data Lab04.eksp;
 infile '/folders/myfolders/SAS/data/lab04/eksperyment.txt' missover;
 input date yymmdd10. num phase $;
 format date yymmdd10.;
run;


data ex4;
set Lab04.eksp;
retain start 0;
if phase = 'START' then start = date;
if phase = 'STOP' then do;
        duration = date - start;
        start = 0;
output;
end;
```

```
keep duration;
run;



/* FIX!!
4.5 Write a data step that reads the text file plikB.txt into the data set B. */


data Lab04.plikB;
     keep date r1 r2 r3 r4;
     retain old_date;
     infile '/folders/myfolders/SAS/data/lab04/plikB.txt' dsd truncover;
     input date ddmmyy10. col1 $ 12-13 val1 15-16 col2 $ 18-19 val2 21-22 col3 $ 24-25 val3
27-28 col4 $ 30-31 val4 33-34;
     format date ddmmyy10. ;
     if date ne . then old_date = date;
     else date = old_date;
     array col(4) col1-col4;
     array val(4) val1-val4;

     do i=1 to dim(col);
          if col(i)="r1" then r1=val(i);
          if col(i)="r2" then r2=val(i);
          if col(i)="r3" then r3=val(i);
          if col(i)="r4" then r4=val(i);
     end;
run;

/*
4.6 Write a data step that reads the text file plikC.txt into the data set C. */

data Lab04.plikC;
 infile '/folders/myfolders/SAS/data/lab04/plikC.txt' missover;
 input id code $ num1 $ num2 $ date ddmmyy10.;
 format date ddmmyy10.;
```

```
run;

/*
4.7 Write a data step that reads the text file plikD.txt into the data set D. */

data Lab04.plikD;
 infile '/folders/myfolders/SAS/data/lab04/plikD.txt' missover;
 input id $ num1 x1 $ num2  x2 $ num3  x3 $ num4 x4 $ ;
run;

data Lab04.plikD;
 infile '/folders/myfolders/SAS/data/lab04/plikD.txt' truncover missover;
     input id $ (a1-a8)($ ?);
     array w(8) a1-a8;

     do i=1 to 7;
          if w(i+1) eq 'k' then do;
               x=input(w(i), best.);
               leave;
          end;
     end;
     keep id x;
run;



/*4.8 Read from the text file p.txt into a data set only the rows with numbers listed in the
first row of the text
file. */

data Lab04.p;
 infile '/folders/myfolders/SAS/data/lab04/p.txt' missover;
 input x1-x10;
run;
```

```
data Lab04.p;
 infile '/folders/myfolders/SAS/data/lab04/p.txt'  truncover;
  array x(10) a1-a10 (10*0);
  if _n_ = 1 then input a1-a10;
  else input x1-x10;

  do i=1 to dim(x);
        if _n_ = x(i) then output;
  end;
  keep x:;
run;

/*
4.9 From each line of the text file braki.txt read the first three non-missing values into a
data sets. (The
selection should be done when reading the values from the text file.) */

data Lab04.braki;
 infile '/folders/myfolders/SAS/data/lab04/braki.txt' missover;
 do until(a1 ne .); * get a column by finding values that are not . ;
        input a1 @; * @ holds the record in the input buffer for the next INPUT statement.;
 end;

 do until(a2 ne .);
        input a2 @;
 end;

 do until(a3 ne .);
        input a3 @;
 end;

run;

/* FIX!!!
```

4.10 The text file bloki.txt has an unknown number of four-rows blocks that start from the numbers: 2004,

2005, 2006 or 2007. Create a data set with four variables (r2004 - r2007) that is a transposition of the

data from the text file bloki.txt. */

```
data Lab04.bloki;
 infile '/folders/myfolders/SAS/data/lab04/bloki.txt' truncover;
 input data x1-x12 @;
run;

proc transpose data=Lab04.bloki out=bloki;
*idlabel data;
run;

data ex10;
 set bloki ;
 array col(*) _numeric_;
 array years(12) _temporary_ (12* 0);
 *array out(4) r2004-r2007 (4 * 0);
 r2004 = 0;
 r2005 = 0;
 r2006 = 0;
 r2007 = 0;
 if _n_ = 1 then do;
  do i = 1 to dim(col);
          years(i) = col(i);
  end;
 end;
 else do;
 counter = 0;
           do i = 1 to dim(col);
                    if years(i) = 2004 then r2004  = col(i);
                    if years(i) = 2005 then r2005  = col(i);
```

```
                if years(i) = 2006 then r2006  = col(i);
                if years(i) = 2007 then r2007  = col(i);
                counter = counter + 1;
                if counter = 4 then    output;
        end;
 end;
 keep r2004-r2007;
run;



/*
5.1 Create a data set with 10 numerical variables z1,...,z10 and 20 observations. Then transpose it without
using PROC TRANSPOSE. */
data a;
array z(10) z1-z10;
do i = 1 to 20;
        do j = 1 to 10;
        z(j) = ranuni(0);
        end;
output;
end;
drop i j;
run;
/*
proc transpose data = a
                                out = b;
run;*/


proc iml;
use a;
read all var _ALL_ into a;
*a = z1 || z2 || z3 || z4 || z5 || z6 || z7 || z8 || z9 || z10;
a = a`;
```

```
create b2 from a;
append from a;
close b2;
run;


data ex1;
  set a end = done;
  array table(20, 10) _temporary_;
  array x(10) x: _all_;
  array y(20);
  retain counter 1;
  do i=1 to dim(x);
          table(counter, i) = x(i);
  end;
  counter = counter+1;
  if done then do;
            /* Transpose the set */
            do i=1 to 10;
                    do j=1 to 20;
                      y(j) = table(j, i);
                    end;
                    output;
            end;
          end;
  keep y:;
run;

/*
5.2 Transform the data set z1 into z2. */
proc transpose data=lab05.z1 out=ex2;
  by art;
  idlabel dat; *nazwy kolumn wartosci ze zmiennej make;
```

```
run;



/*
5.3 Order the variables in the data set z alphabetically. */
proc sql;
  select name
  from dictionary.columns
  where memname = 'Z'
  order by name asc;
quit;


/*
5.4 The data set a has the variables x and y.
(a) For every value of x find the most frequently appearing values of the variable y (do not
care for
ex-aequo situations, if they appear – just pick any of the most frequent values of y). */



proc sql;
  select x, y
  from lab05.a lab5
  group by x, y
  having count(y) = (select max(bla.ycount)
                                        from (select count(*) as ycount
                                                from lab05.a
                                                where lab5.x = x
                                                group by x, y) bla /* we need y here to get
the correct max count*/
                                        );
quit;



*(b) For every value of x list all the most frequent values of the variable y.;
```

```
proc sql;
  select x, y
  from lab05.a lab5
  group by x /* do not group by y */
  having count(y) = (select max(bla.ycount)
                                        from (select count(*) as ycount
                                                from lab05.a
                                                where lab5.x = x
                                                group by x, y) bla /* we need y here to get
the correct max count*/
                                        );
quit;

proc sql;
select x, y
  from lab05.a one
  where y in (select y
        from (select y, count(y) as ycount
          from lab05.a
          where x = one.x /* must be the same */
          group by y) as g
       where ycount = (select max(ycount) /* get maximum ycount */
       from (select y, count(y) as ycount /* same as above */
          from lab05.a
          where x = one.x
          group by y) as l)
     );
run;
*(c) Find all such values of x for which there exists exactly one smallest y.;

*(d) Find all such values of x for which there are no repeating values of the variable y.;

*(e) Find all such values of x that have the largest number of distinct values of y.;
```

*(f) Find all values of the variable x for which the values of y form the set {1, . . . , n} for some n ∈ N.;

*(g) Find all such x for which the distinct values of y form the set {1, . . . , n} for some n ∈ N.;

*(h) Find such values of y that correspond to at least half of the values of x that appear in the data set a.;

/*

5.5 The data set z3 has two variables id, year and sales.

*• Find all values of id that did not appear before 1993. */

*• Find all values of id that appeared both in the first and in the last year.;

*• Find all values of id that were present in every year.;

/*

5.6 The data sets z1 and z2 contain some observations of the common variable x. Count the number of distinct

values of x that appear: only in the first set, only in the second set, in both. */

```
proc sql;
 select zs1.cnt, zs2.cnt, zs1.cnt + zs2.cnt
 from (select count(tmp.art) as cnt from (select distinct art from Lab05.z1) as tmp) as zs1
 join (select count(tmp2.art) as cnt from (select distinct art from Lab05.z2) as tmp2) as zs2
 on 1=1; /* warunek kiedy mają się kolumny złączyć */
quit;
```

/*

5.7 The data set b has the variables a1, x1, a2, x2. */

*(a) Treating a1 and a2 as grouping variables, for each group formed by a1 pick such values of x1 that are

between the smallest and the largest value of x2 in the same group formed by a2.;

*(b) Identify the name of the group that appears b most frequently.;

/*

5.8 With a single SQL query, basing on the data set c: */

* (a) find the month in which r2 has the largest number of missing values,;

*(b) find the month in which the values of r1 are most scattered.;

/* 6.1 Two data sets are given: z1 and z2, both with the numerical variable x. Treating them as sequences (that

is sets in which order of elements matters), write a single DATA STEP that shows the number of differing

elements in the sets, in the Log window.*/

```
data z;
retain difference 0;
merge Lab06.z1 (RENAME=(x=z1)) Lab06.z2 (RENAME=(x=z2)) end=done;
if z1 NE z2 then do;
        difference = difference + 1;
        if done then do;
                put 'Number of different observations: ' difference;
        end;
        drop difference;
end;
run;
```

```
/* 6.2 (Table look-up I) What is the average of sales from duzy (big) computed only for
those values of id
that are in the set maly (small)?*/

proc sort data=lab06.duzy out=duzy;
by id;
run;

proc sort data=lab06.maly out=maly;
by id;
run;

data ex2;
 merge duzy (in=big) maly (in=small);
 by id;
 if big and small;
run;

data ex2avg;
set ex2 end=done;
retain sum 0;

sum = sum + sales;

if done then do;
avg = sum / _n_;
output;
end;

keep avg;
run;



/*
```

6.3 What is the average of the variable sales from the data set duzy (big) computed only for the observations

with the numbers listed in the data set numery (numbers)? */

```
proc sort data=lab06.numery out=numery;
by nr;
run;

data ex3;
set numery end=done;
set duzy point=nr;
retain sum 0;
sum = sum + sales;

if done then do;
        avg = sum/_n_;
        output;
end;

keep avg;
run;
```

/*

6.4 Delete all the missing values from kropki (dots), that is transform kropki (no-dots) into bezkropek. */

```
data z1;
 set Lab06.kropki;
 if z1 ne . then output;
 keep z1;
run;

data z2;
 set Lab06.kropki;
 if z2 ne . then output;
```

```
 keep z2;
run;


data z3;
 set Lab06.kropki;
 if z3 ne . then output;
 keep z3;
run;


data ex4;
set z1;
set z2;
set z3;
run;


/*
6.5 Create a data set liczby with 50 arbitrary numerical observations (with one numerical variable). Write a
single DATA STEP that will create (basing on the data set liczby) the data set sumy with one numerical
variable suma and 46 observations. The i-th observation in sumy should be the sum of observations from
liczby numbered {i, . . . , i + 4}. */


data liczby;
do i = 1 to 50;
        x = floor(ranuni(0) * 100);
        output;
end;
keep x;
run;



data sumy;
```

```
 set liczby;
        lag1 = lag(x);
        lag2 = lag2(x);
        lag3 = lag3(x);
        lag4 = lag4(x);

 if _n_ > 4 then do;
        * BŁĄD! jeśli lag tworzy kolejke od _n_ > 4 to poprzednie będą missing!!!;
        * sum = x + lag(x) + lag2(x) + lag3(x) + lag4(x);
        sum = x + lag1 + lag2 + lag3 + lag4;
        output;
 end;
 keep sum;
run;



/*
6.6 Write a single DATA STEP that acts as the following SQL query: */

proc sql noprint;
create table aa as
select *, count(y) as licznik /*count the occurances of y in each group */
from Lab06.a
group by x
having count(y)>5 /* take only if more than 5 */
;
quit;

data ex5;
 set Lab06.a;
 array values (20) _temporary_ (20 * 0);
 by x;
 retain licznik 0;
```

```
licznik = licznik + 1;
values(_n_) = y;

if first.x then do;
        licznik = 0;
end;
if last.x then do;
        if licznik > 5 then do;
        licznik = licznik + 1; * bo liczyliśmy od 0;
         do i = _n_ - licznik to _n_;
         put i=;
                y = values(i);
                output;
         end;
        end;
end;

 drop i;
run;
/*
6.7 Basing on the data sets zb1,..., zb5 create a data set wynik that has the whole year
2007 in the variable
data. */

data wynik;
        set Lab06.zb1 Lab06.zb2 Lab06.zb3 Lab06.zb4 Lab06.zb5;
run;

proc sort data=wynik out=wyniksorted;
by data;
run;

/*
```

6.8 The data sets jan, feb and mar have two variables : osoba (person) and wynik (result). Generate a data

set with the most current results for every person (the names of the sets refer to the names of months in

which the measurements were made). */

```
data ex8;
 merge Lab06.jan Lab06.feb Lab06.mar;
run;
```

/*

6.9 The data sets zx and zy have single numerical variable (x and y respectively), and the set zxy has both x

and y. Find the number of observations from zxy such that their values of x and y are equal to the values

of x from zx and y from zy with the same observation number.*/

```
data zxy;
merge Lab06.zx Lab06.zy Lab06.zxy (rename=(x=new_x y=new_y));
run;

data _null_;
 set zxy end=done;
 retain counter 0;
 if x = new_x and y = new_y then do;
  counter = counter + 1;
  put 'Matching row: ' _n_;
 end;

 if done then put 'Matching observations: ' counter;

run;
```

/*

6.10 Use a single DATA STEP to update pierwszy (first) with the data from drugi (second).

- If a given year and month are missing in both sets, they need not be placed in the updated set.

- If for a given year and month there is no data in pierwszy (or in drugi),

one should leave in the updated file the value of sales from drugi (pierwszy respectively) - if it exists.

- If some year and month are present in both files, the value of sales from pierwszy should be increased

by the relevant value from drugi.)

- One can assume that both files are sorted in ascending order with respect to year and that the names of

sets and the names and types of variables are known. */

```
data drugi;
set Lab06.drugi end=done;
 array months(*) Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec;
 do i = 1 to dim(months);
        month = i;
        sales_drugi = months(i);
        output;
 end;

keep year month sales_drugi;
run;

data ex10;
 merge Lab06.pierwszy (rename=(sales=sales_pierwszy)) drugi;
 if sales_drugi ne . and sales_pierwszy ne . then do;
 sales = sales_drugi + sales_pierwszy;
 output;
 end;
 else if sales_drugi = . and sales_pierwszy ne . then do;
 sales = sales_pierwszy;
 output;
 end;
 else if sales_drugi ne . and sales_pierwszy = . then do;
```

```
 sales = sales_drugi;
 output;
 end;
run;
```

/* The database of the car rental company has the tables: klienci (clients) (with the variable nr klienta that

identifies a client in a unique way), wypozyczenia (rentals) (with the variable nr wypozyczenia that identifies

a rental in a unique way), wypozyczalnie (rental offices) (with the variable nr miejsca that identifies a rental

office in a unique way), pracownicy (employees) (with a variable nr pracownika that identifies an employee in

a unique way) and samochody (cars) (with the variable nr samochodu that identifies a car in a unique way). */

/* 7.1 */

*(a) Find the rental office with the largest number of rentals between January 1, 1999 and June 6, 1999.;

```
proc sql;
	select nr_miejsca_wyp
	from lab07.wypozyczenia
	where input(data_wyp, yymmdd10.) >= input("1999/01/01", yymmdd10.) and input(data_odd, yymmdd10.) <= input("1999/06/06", yymmdd10.)
	group by nr_miejsca_wyp
	having count(*) = (select max(subq.counter)
			from (select count(*) as counter from lab07.wypozyczenia group by nr_miejsca_wyp) subq);
quit;
```

*(b) Find the names of the clients who have been renting cars more than once and have rented an Opel at

least once.;

```
proc sql;
```

```
  select clients.imie, clients.nazwisko
  from lab07.klienci as clients
  where clients.nr_klienta in (
                                                    select rentals.nr_klienta
                                                    from lab07.wypozyczenia as rentals
                                                    join lab07.samochody as cars
                                                    on rentals.nr_samochodu =
cars.nr_samochodu
                                                    where cars.marka = 'OPEL'
                                                    group by rentals.nr_klienta
                                                    having count(*) >= 1
                                                    intersect
                                                    select rentals.nr_klienta
                                                    from lab07.wypozyczenia as rentals
                                                    join lab07.samochody as cars
                                                    on rentals.nr_samochodu =
cars.nr_samochodu
                                                    group by rentals.nr_klienta
                                                    having count(*) > 1
                                                   );


quit;
```

*(c) For each rental office pick those cars that have been rented between October 1, 1998 and December

31, 1998 (for each rental office list the cars in the ascending order with respect to the length of the

rental).;

```
proc sql;

  select rentals.nr_miejsca_wyp, cars.nr_samochodu, rentals.data_wyp, rentals.data_odd
  from lab07.wypozyczenia as rentals
  join lab07.samochody as cars
```

```
  on rentals.nr_samochodu = cars.nr_samochodu
  where input(data_wyp, yymmdd10.) >= input('1999/01/01', yymmdd10.) AND
input(data_wyp, yymmdd10.) <= input('1999/06/06', yymmdd10.)
  order by (input(data_odd, yymmdd10.) - input(data_wyp, yymmdd10.)) asc;

quit;
```

*(d) Find the names of those clients that have rented cars more than once and each time have rented cars

of different makes.;

```
proc sql;

  select clients.nr_klienta, clients.imie, clients.nazwisko, count(*)
  from lab07.klienci as clients
  join lab07.wypozyczenia as rentals
  on clients.nr_klienta = rentals.nr_klienta
  join lab07.samochody as cars
  on rentals.nr_samochodu = cars.nr_samochodu
  where cars.marka not in (select cars2.marka
                                                from lab07.samochody as cars2
                                                join lab07.wypozyczenia as rentals2
                                                on cars2.nr_samochodu =
rentals.nr_samochodu
                                                where cars2.marka = cars.marka)
  group by clients.nr_klienta, clients.imie, clients.nazwisko
  having count(*) > 1;

quit;
```

*(e) Give the list of the employees that had not been involved in any car rental between October, 1999

and February, 2000.;

*(f) For the rentals in which the pick-up and return of the car were made in different rental offices, find

the names of the employees involved in the rentals.;

*(g) Find the employee who had been hired before 1998 and produced the highest gain to the company in

1999.;

*(h) Present history of the rentals of the car with the code 000006 (pick-ups dates, return dates, names of

clients, costs of rentals).;

/* 7.2 The data set pomiary (measurements) contains measurements made with instruments instrument in some

days of January 2007. For each pair (instrument, data) from daty find the closest (in time) measurement

(the variable pomiar) from the data set pomiary.*/

/* 7.3 (Table look-up II) Solve Problem 2 from Lab 7 with PROC SQL. */

/* 7.4 Write a query that imitates the following DATA STEP:*/

data razem;

merge a(in = ina) b(in = inb);

by a b c;

if inb;

if inb and not ina then indyk=1;

run;

/* 7.5 The data set studenci (students) has the variable that identifies students uniquely (id studenta) and the

variable (id przedmiotu) with the codes of the courses attended by students. For every student find those

other students that attend the courses from the list of her/his courses. */

```
/* 8.1 Basing on the data set magazyn (warehouse) create a text file with a report
containing dates, codes and

full names of the missing products (in order to do that one should define a format using
the dictionary

data set produkty (products)). */


data lab08.unsorted_dictionary;
        set lab08.produkty;
        start = product_code;
        label = product_name;
        fmtname = "myformat";
        keep start label fmtname;
run;


proc sort data=lab08.unsorted_dictionary nodupkey out=lab08.products_dictionary;
        by start;
run;


proc format library = work cntlin = lab08.products_dictionary;
run;


data _null_;
        set lab08.magazyn;
        product_name = put(product_code, myformat.);
        if quantity = 0;
   FILE '/folders/myfolders/SAS/data/lab08/missing_products.txt';
   PUT date product_code product_name;
run;



/* 8.2 (Table look-up III) Solve the problem 2 from Lab 6 using PROC FORMAT. */
/* 6.2 What is the average of sales from duzy (big) computed only for those values of id
```

```
that are in the set maly (small)?*/

data lab08.unsorted_dictionary2;
 set lab06.duzy;
          start = id;
          label = sales;
          fmtname = "my2format";
          keep start label fmtname;
run;

proc sort data=lab08.unsorted_dictionary2 nodupkey out=lab08.duzy_dictionary;
          by start;
run;

proc format library = work cntlin = lab08.duzy_dictionary;
run;

data maly_formatted;
 set lab06.maly;
 sales = input(put(id, my2format.), 8.);
 keep sales;
run;

data _null_;
 set maly_formatted end=done;

 retain sum 0;

 sum = sum + sales;

 if done then do;
  average = sum/_n_;
  put average=;
```

```
 end;


run;


/* 8.3 Create a data set a with five numerical variables z1,...,z5 and 50 observations. Make every element of a

random number from the normal distribution with the mean 100 and the variance 10. Next, basing on a,

create a data set stat, with the variable stat and the variables z1,...,z5 and with 55 observations. For the

observations numbered n = 1, . . . , 50, the variable stat should have (text) values W n, and the variables

z1,...,z5 should have exactly the same values as in the set a. For the observations numbered 51, . . . , 55

the variable stat should take the (text) values N, MIN, MAX, MEAN, ST D, and the variables z1,...,z5

should take the values of those statistics for the variables z1,...,z5. */


data lab08.a;
        array z(5);
        do o = 1 to 50;
                do i = 1 to dim(z);
                        z(i) = rannor(10) + 100; *normal distribution with the mean 100 and the variance 10;
                end;
                output;
        end;
        keep z:;
run;


proc means data=lab08.a noprint;
        var z:;
        output out=lab08.a_stat N= MIN= MAX= MEAN= STD= /autoname;
run;


data lab08.stat;
```

```
        retain stat;
        set lab08.a end = done;
        stat = 'W' || strip(put(_N_, $3.));
        output;
        if donethen
        do;
                array z(*) z:;
                set lab08.a_stat;
                stat = 'N';
                z(1) = z1_N; z(2) = z2_N; z(3) = z3_N; z(4) = z4_N; z(5) = z5_N;
                output;
                stat = 'MIN';
                z(1) = z1_Min; z(2) = z2_Min; z(3) = z3_Min; z(4) = z4_Min; z(5) = z5_Min;
                output;
                stat = 'MAX';
                z(1) = z1_Max; z(2) = z2_Max; z(3) = z3_Max; z(4) = z4_Max; z(5) = z5_Max;
                output;
                stat = 'MEAN';
                z(1) = z1_Mean; z(2) = z2_Mean; z(3) = z3_Mean; z(4) = z4_Mean; z(5) =
z5_Mean;
                output;
                stat = 'STD';
                z(1) = z1_StdDev; z(2) = z2_StdDev; z(3) = z3_StdDev; z(4) = z4_StdDev; z(5)
= z5_StdDev;
                output;
        end;
        keep stat z1 z2 z3 z4 z5;
run;


/*8.4 Modify the data set stat from the previous problem to contain the values of quartiles,
median and inte-
rquartile range for the variables z1,...,z5. */


proc summary data=lab08.a noprint;
```

```
	var z:;
	output out=lab08.a_summary Q1= MEDIAN= Q3= QRANGE= /autoname;
run;


data lab08.stat_summary;
	retain stat;
	set lab08.stat end = done;
	output;
	if donethen
	do;
		array z(*) z:;
		set lab08.a_summary;
		stat = 'Q1';
		z(1) = z1_Q1; z(2) = z2_Q1; z(3) = z3_Q1; z(4) = z4_Q1; z(5) = z5_Q1;
		output;
		stat = 'MED';
		z(1) = z1_Median; z(2) = z2_Median; z(3) = z3_Median; z(4) = z4_Median; z(5)
= z5_Median;
		output;
		stat = 'Q3';
		z(1) = z1_Q3; z(2) = z2_Q3; z(3) = z3_Q3; z(4) = z4_Q3; z(5) = z5_Q3;
		output;
		stat = 'IQR';
		z(1) = z1_QRange; z(2) = z2_QRange; z(3) = z3_QRange; z(4) = z4_QRange;
z(5) = z5_QRange;
		output;
	end;
	keep stat z1 z2 z3 z4 z5;
run;



/* 8.5 Write a code that reads texts like: January 22, 2001, October 3, 1956, and so on, as
genuine SAS dates. */
```

```
data ex5;
 array months {12} $ _temporary_ ('January' 'February' 'March' 'April' 'May' 'June' 'July'
'August' 'September' 'October' 'November' 'December');
 do i = 1 to 5;
  date = strip(months[int(ranuni(0) * 12 + 1)]) || put(int(ranuni(0) * 28 + 1), $3.) || ', ' ||
put(int(1900 + ranuni(0) * 100), $4.);
  output;
 end;
 drop i;
run;


proc format;
 invalue mymonthformat
   January = 1
   February = 2
   March = 3
   April = 4
   May = 5
   June = 6
   July = 7
   August = 8
   September = 9
   October = 10
   November = 11
   December = 12
   ;
run;


data ex5_format;
 set ex5;
 month = input(scan(date, 1, ' '), mymonthformat.);
 day = input(compress(scan(date, 2, ' '),,'p'), 2.);
 year = input(scan(date, 3, ' '), 4.);
```

```
 sas_date = mdy(month, day, year);


run;


/* 8.6 Basing on the set oceny (grades) create a data set srednie (averages) with the
average grades of each
student for each course. */


proc sql;
        create table lab08.srednie as
        select uczen, kod, avg(ocena) as srednia
        from lab08.oceny
        group by uczen, kod
        ;
quit;



/* 8.7 The data set dane (data) has the variables: grupa (group), x i y. Find the group for
which the (group)
averages of x and y are closest to the global averages of x and y (global means computed
for the whole
set dane). */


proc means data=lab08.dane noprint;
class group;
var x;
var y;
output out=ex7 (drop= _TYPE_ _FREQ_) mean(x) = m_x mean(y) = m_y;


run;


data _null_;
 retain globx 0;
 retain globy 0;
```

```
 retain diff 0;
 retain closest 'A';
 set ex7;
 if _n_ = 1 then do;
          globx = m_x;
          globy = m_y;
 end;
 else do;
          diffx = abs(globx - m_x);
          diffy = abs(globy - m_y);
          diffxy = diffx + diffy;

          if _n_ = 2 then diff = diffxy;

          if diffxy < diff then do;
           diff = diffxy;
           closest = group;
          end;
 end;
 put _n_= diffxy= closest=;
run;
```

/* 8.8 Define a format that displays numbers of the form m.n (m, n = 0, ..., 9) "in words". An example: 2.8
should be formatted as two point eight. */

```
data ex8;
 input num;
 cards;
 2.8
 3.4
 7.9
 ;
run;
```

```
proc format;
 value mynumwordformat
 1 = one
 2 = two
 3 = three
 4 = four
 5 = five
 6 = six
 7 = seven
 8 = eight
 9 = nine
 . = point
 ;
run;

data _null_;
 set ex8;
 char_num = put(num, 3.2);
 first = put(input(scan(char_num, 1, '.'), 2.), mynumwordformat.);
 second = put(., mynumwordformat.);
 third = put(input(scan(char_num, 2, '.'), 2.), mynumwordformat.);
 word = compress(first) || ' ' || compress(second) || ' ' || compress(third);

 put char_num= word= ;
run;

/* 8.9 Define an outlier of the order a as any observation which lies outside of the range
(med - a * Range, med + a * Range),
where med and Range are the median and interquartile range respectively. Write a code that, for a given
parameter alfa = 1 and any given data set with one numerical variable, will find all the outliers of the
order a. */
```

```
data ex9;
 input num;
 cards;
 324
 26
 13
 33
 7
 4
 8
 0
 23
 ;
run;

proc summary data=ex9 noprint;
        var num;
        output out=ex9_summary (drop = _freq_ _type_) MEDIAN= QRANGE= /autoname ;
run;


%macro exe9(set, a);

 proc summary data=&set noprint;
        var num;
        output out=ex9_summary (drop = _freq_ _type_) MEDIAN= QRANGE= /autoname ;
 run;

 data together;
  set ex9_summary &set;
 run;
```

```
 data _null_;
 set together;
 retain lower 0;
 retain upper 0;
 if _n_ = 1 then do;
 lower = num_Median - &a * num_QRange;
 upper = num_Median + &a * num_QRange;
 put lower= upper=;
 end;
 else do;
 if num < lower or num > upper then put num "is an outlier.";
 else put num "is not an outlier.";
 end;
run;


%mend;


%exe9(ex9, 1);


/* 8.10 Perform the normality test and use the Q-Q plot to verify whether the observations from the data set
probka (sample) are normally distributed . Present the histogram of values of x with the normal curve
on it.*/


proc univariate data=lab08.probka;
 var x;
 qqplot;
 histogram/ midpoints=-3 to 3  by .5;
 output out=probka_norm normaltest=s;
run;
```

/* 9.1 Write a macro %create(prefix,N,k,l) that creates N data sets with the names prefix1,...,prefixN. Each data

set should have k variables and l observations from the uniform distribution on (-1,1). */

```
%macro create (prefix, N, K, l);
 data %do i = 1 %to &N;
                  &prefix&i
                  %end; ;
 array k(&k) (&k * 0);

 do obs = 1 to &l;
         do var = 1 to &k;
         k[var] = ranuni(0) * 2 - 1;
         end;
         output;
 end;
 keep k:;
 run;

%mend;

%create(name, 3, 5, 10);
```

/* 9.2 Write a macro %count(set,variables,n) which for every variable from the data set set will find the number

of observations larger than the number n. */

```
%macro count(set, variable, n);
 data _null_;
  retain sum 0;
  set &set end=done;

  if (&variable > &n) then sum = sum + 1;
```

```
 if done then put sum=;

run;

%mend;

%count(name3, k2, 0);

* for multiple variables;

%macro count_multiple(set, variables, n);
 %let num = %eval(%sysfunc(countw(&variables));
 %put **&num**;
 data _null_;
  set &set end=done;
  array counters(&num) _temporary_ (&num * 0);
  array vars(&num) &variables;
  do i = 1 to &num;
  if (vars[i] > &n) then counters[i] = counters[i] + 1;
  end;

  if done then do;
            put "The counters for &variables are respectively:" ;

            do i = 1 to &num;
             counter = counters[i];
             put counter=;
            end;
  end;

 run;
```

```
%mend;


%count_multiple(name3, k2 k3 k5, 0);


/* 9.3 Transform the data set kropki (dots) into bezkropek (without dots). (That is, remove the missing values
for every variable in kropki.) */


%macro dotremover();
 %local varnum;
 proc sql noprint;
 select nvar
 into: varnum
 from dictionary.tables
 where libname='LAB09' and memname='KROPKI';
 quit;
 %put **&varnum**;

 %do i = 1 %to &varnum;
 data z&i;
  set lab09.kropki;
  if (z&i ne .) then output;

  keep z&i;
 run;
 %end;

 data bezkropek;
 merge %do i = 1 %to &varnum;
   z&i
 %end; ;

 run;
```

```
%mend;


%dotremover();



/* 9.4 Assume the data set a contains one numerical variable and 50 observations. Write a macro that creates a

data set srednie (averages) with one variable and 50 observarions. The ith observation in srednie should

be the average of the observations from a numbered {i, ..., 50}. The problem should be solved in two ways: */


data a;
 do i = 1 to 50;
  output;
 end;
run;


*(a) one can merge a number of copies of a (with appropriately shifted observations),;
%macro sillysolution();
 %do k = 1 %to 50;
  data a&k;
   set a;
   if (i >= &k) then output;
  run;
 %end;


%do k = 1 %to 50;
 proc means data=a&k noprint;
          var i;
          output out=a_mean&k (keep=i_mean) MEAN=i_mean;
 run;
 %end;
```

```
 data all_means;
   set %do k = 1 %to 50;
                    a_mean&k
                    %end;
                    ;
  run;
%mend;

%sillysolution();

*(b) one can transpose the set a (and count the averages from the relevant columns).;
%macro transposesolution();
 proc transpose data = a
                                    out = a_t (drop=_name_);
 run;

 data means;
 set a_t;
 array values(50) _numeric_;

 do i = 1 to 50;
          mean = 0;
          do j = i to 50;
                    mean = mean + values[j];
          end;
          mean = mean /(50-i + 1);

          output;
 end;
 keep mean;
 run;
%mend;
```

```
%transposesolution();


/*9.5 Write a macro that returns the number of words in a given macrovariable and a
macro that writes each

word from a given macrovariable into a separate macrovariable. */


%macro word_count(text);
  %let i = 1;
  %let word = %scan(&text, &i, ' ');
  %let word_1 = &word;
  %do %while (&word ne );
        %let i = %eval(&i + 1);
        %let word = %scan(&text, &i, ' ');
        %if (&word ne ) %then %let word_&i = &word;
  %end;
  %let i = %eval(&i - 1);
  %put **&i**;
  %put _user_;


%mend;


%word_count(ala ma kota a kot ma ale);


/* 9.6 Write a macro that computes the value of n! (the macro should not contain any DATA
STEP). */


%macro factorial(n);
 %let f = 1;
 %do i = 1 %to &n;
  %let f = %eval(&f * &i);
 %end;
 %put ***&f***;
%mend;
```

```
%factorial(4);


%macro factorial_recursive(n);
  %if &n eq 1 %then 1;
  %else %eval(&n*%factorial_recursive(%eval(&n-1)));
%mend;
%factorial_recursive(4);



/* 9.7 Write a macro that depends on two parameters names and chars and shows in the Log window all the
words from a given string names that DO NOT contain the characters listed in the parameter chars. */


%macro stringcheck(names, chars);
 %let names_cnt = %eval(%sysfunc(countw(&names));
 %let chars_cnt = %eval(%sysfunc(countw(&chars));


 data _null_;
  array names(&names_cnt) $ _temporary_;
  array chars(&chars_cnt) $ _temporary_;


  do i = 1 to dim(names);
   names(i) = scan("&names", i, " ");
  end;


  do i = 1 to dim(chars);
   chars(i) = scan("&chars", i, " ");
  end;


  do i = 1 to dim(names);
          found = 0;
             do j = 1 to dim(chars);
                    pos = find(names(i), compress(chars(j)));
```

```
                if pos > 0 then found = 1;


            end;
            if found = 0 then put names(i);
 end;


 run;
%mend;


%stringcheck(lubie placki ziemniaczane, ie oko);


/* 9.8 Generate N macrovariables named z1, .., zN so as to have a randomly chosen capital letter as the value

of each macrovariable. (Clearly, it may happen that the values of distinct macrovariables are identical.)

Show in the Log window all the macrovariables with distinct values. */


%macro rand_letter(N);
 data _NULL_;
  %do i = 1 %to &N;
   random = byte(floor(65+26*ranuni(0)));
   call symput("z&i", random);
 %end;
 run;


 data _null_;
 array letters(&N) $ _TEMPORARY_;
  %do i = 1 %to &N;
           if "&&z&i" not in letters then do;
            letters(&i) = "&&z&i";
            put "z&i = " letters(&i);
           end;
 %end;
```

```
 run;
 %put _USER_;
%mend;

%rand_letter(20);
```

/* 9.9 Write a macro %comb(n,k) that creates for given n, k ∈ N a data set combinations with k variables and

(n k)observations. The rows of combinations should contain k-element combinations of the set {1,. . . ,n} . */

```
%macro comb(n, k);
 %let block = %sysfunc(comb(&n, &k));

 proc plan;
     factors Block=&block ordered
           Treat= &k of &n comb;
     ods output Plan=combs;
 run;

 data combinations;
  set combs;
  drop Block;
 run;

%mend;

%comb(6, 4);
```

/* 10.1 Write a macro that divides any given data set into sets that have at most n observations. */

```
data a;
```

```
 do i = 1 to 55;
          output;
 end;
run;


%macro splitter(lib, set, n);
 %local obsnum;
 proc sql noprint;
 select nobs
 into: obsnum
 from dictionary.tables
 where libname=%upcase("&lib") and memname=%upcase("&set");
 quit;
 %put **&obsnum**;
 %let amount = %eval(%sysfunc(ceil(&obsnum / &n)));
 %do i = 0 %to &amount - 1;
            data a&i;
             set &set;
              if (i >= &i * &n and i < &n * (&i + 1)) then output;
             keep i;
            run;
 %end;


%mend;


%splitter(work, a, 10);


/* 10.2 Write a macro with two parameters id and sets that finds for a given id the most
current value of the

variable x in the data sets sets. (The parameter sets may contain any number of set names
separated by

spaces.) If the given id does not exist in the given sets, the macro should generate an
appropriate message

in the Log window. (For testing purposes one can use the sets a0236 – a0962.) */
```

```
%macro current(lib, sets, id);
 %let amount = %eval(%sysfunc(countw(&sets));
 %put **&amount**;

  data _null_;
   set %do i = 1 %to &amount;
                  &lib..%scan(&sets, &i, ' ')
        %end;
        end = done;
   retain curr_date 0;
   retain curr_x 0;

   if (id eq &id) then do;
    put date= date9.;
    if (curr_date ne '') then do;
        if date > curr_date then do;
                  curr_x = x;
                  curr_date = date;
        end;
    end;
    else do;
       curr_x = x;
         curr_date = date;
    end;
   end;
   if done then put curr_x= curr_date= date9.;
  run;

%mend;

%current(lab10, a0236 a0346 a0447 a0594 a0962, 0009);
```

/* 10.3 Write a macro that for a given data set and a given numerical variable from the set will create a new

format. For example, if in the set there exists the numerical variable x with the values {0.1, 3, 100}, then

the created format should display numbers from the interval (−∞, 0.1) as ,,I", numbers from (0.1, 3] as

,,II", numbers from (3, 100] as ,,III" and numbers from (100, +∞) as ,,IV". */

```
data ex3;
        input x;
        cards;
        -15
        24
        1
        50
        ;
run;

%macro reformat(set, var);
 proc sort data = &set out = &set;
        by &var;
 run;

 proc sql noprint;
        select count(a)
        into :counter
        from (
                select distinct &var as a
                from &set
                );

        select distinct &var
        into:v1-:v&sysmaxlong
        from &set
```

```
        ;
quit;

proc format;
 invalue xconverter

   -1000000000 - &v1 = 1
        %do i = 1 %to %eval(&counter - 1);
        %let new_i = %eval(&i + 1);
                &&v&i - &&v&new_i  = &i
        %end;
        &&v&new_i - &sysmaxlong = &new_i
 ;
run;



%mend;

%reformat(ex3, x);

data _null_;
 a = input(28, xconverter.);
 put a=;
run;

/* 10.4 Write a macro that removes from any given data set all the numerical variables
that have at least one
missing value. */
data C;
  do z=1 to 20;
    do j=1 to 10;
          x = round(ranuni(0) * 10);
          if x < 5 then
            y = .;
```

```
            else
              y = x;
          end;
          output;
  end;
  *drop x j;
run;


%macro varremover(lib, set);
 PROC SQL noprint;
          select distinct name
          into :names separated by ' '
          from dictionary.columns
          where libname=%upcase("&lib") and memname=%upcase("&set") and type='num'
          ;
          %put **&names**;
          select count(distinct name)
          into :counter
          from dictionary.columns
          where libname=%upcase("&lib") and memname=%upcase("&set") and type='num'
          ;
 QUIT;

          data _null_;
           set &lib..&set end=done;
           array vars(*) _NUMERIC_;
           array ismissing(&counter) _temporary_ (&counter * 0);

           do i = 1 to &counter;
                    if(vars[i] eq .) then do;
                     ismissing[i] = 1;
                     put i=;
```

```
                end;
        end;

        if done then do;
        missingvars = "";
                do i = 1 to &counter;
                  if ismissing[i] = 1 then missingvars = compress(missingvars || " " || put(i,
2.));
                end;
                call symput('todrop', missingvars);
         end;
        run;
        %put **&todrop**;

        data new_&set;
                set &lib..&set;

                %let count = %eval(%sysfunc(countw(&todrop)));

                %do i = 1 %to &count;
                        %let index = %scan(&todrop, &i, ' ');
                        %put **&names**;
                        %let val = %scan(&names, &index, ' ');
                        %put **&val**;
                        drop &val;
                %end;
        run;
%mend;

%varremover(work, c);

proc sql;
describe table dictionary.tables;
describe table dictionary.columns;
```

```
describe table sashelp.class;
quit;
```

/* 10.5 Write a macro %division(set,var) that divides a given data set zbior into as many sets as there are distinct

values of the variable var. The ith output set should be named zi and it should contain only such

observations from set for which var takes the ith of its values. */

```
%macro division(set, var);

	proc sql;
		select count(a)
		into :counter
		from (
			select distinct &var as a
			from &set
			)
		;
	quit;
	%put **&counter**;

	%do i = 1 %to &counter;
	 data z&i;
		set &set;* point=&i;
		if(_n_ eq &i) then	output;
		*stop;
	 run;

	%end;
%mend;

%division(c, y);
```

/* 10.6 Write a macro that for given: a data set, a numerical variable and a number n, counts the value of the

empirical distribution of the variable at the points x1, . . . , xn, where x1 and xn are the smallest and the

largest values of the variable in the set. The distances between the consecutive points xi and xi+1 should

be equal for all i. */

/* 10.7 Assume that a data set z with the text variable set and the numerical variable var is given; every observation

from z contains the name of variable (var) that should be removed from the set set. Write a macro that

reads the set z and the sets listed in z, removes the relevant variables from the listed sets and puts them,

side by side, into a single new data set. */

```
data z;
input set$ var$;
cards;
z1 z
z2 j
z3 x
z4 y
;
run;

%macro setvarremove();
 proc sql noprint;
        select set
        into :sets separated by ' '
        from z
        ;

        select count (set)
```

```
        into :count
        from z
        ;

        select var
        into :vars separated by ' '
        from z
        ;
quit;

%do i = 1 %to &count;
%let name = %scan(&sets, &i, ' ');
        data new_&name;
                set &name;
                %let var = %scan(&vars, &i, ' ');
                drop &var ;
        run;

 %end;
%mend;

%setvarremove();

/* 10.8 Write a macro that, for a given data set set and a given integer k, will create the set of all k-element
combinations of elements of set. */

/*11.1 Write a macro that will replace (in a given data set having only numerical variables ) n randomly picked
elements with the missing values. */

%macro randomdot(set, n);
  proc sql noprint;
```

```
 select name
 into :names separated by ' '
 from dictionary.columns
 where memname=%upcase("&set")
 ;

 select nobs
 into :obsnum
 from dictionary.tables
 where memname=%upcase("&set")
 ;
quit;

 %let amount = %eval(%sysfunc(countw(&names)));
 %put **&names* *&amount**;

%let indeces =  ;
%let count = 0;
%do i = 1 %to &obsnum;
        %let r = %eval(%sysfunc(round(%sysevalf(%sysfunc(ranuni(0))))); *random [0,1];
                %if &r eq 1 %then %do;
                        %let indeces = &indeces &i;
                        %let count = %eval(&count + 1);
                        %if &count = &n %then %let i = %eval(&obsnum);
                %end;
 %end;
 %put **&indeces**;

 data new_&set;
 set &set;

        %do i = 1 %to &n;
        %let index = %scan(&indeces, &i, ' ');
```

```
        if (_n_ eq  &index) then do;
          %let r = %eval(1 + %sysfunc(floor(%sysevalf((&n - 1) * %sysfunc(ranuni(0))))));
               %let var = %scan(&names, &r, ' ');
               %put **&var**;
               &var = .;
        end;
       %end;
  run;
%mend;


%randomdot(c, 5);


/*
Important errors:
Global sql not supported - there must be an extra semicolon somewhere.
xxx seems to be a text - a macro that is used there must have a wrong (empty) value. Check WHY.
Open code recursion - something is not wrapped in sysfunc.
*/


/* 11.2 Write a macro with the two parameters numofvars and numofgrps that will create a random data set with
the variables v 1 –v numofvars with the values in the set A={A1, A2, ..., Anumof grps}. The set A should be
a subset to the set of value for every variable. Furthermore, the values of every variable should be ordered
in the invcreasing order, like in the set Gen 3 6 which can serve as an example. */


/* 11.3 How to check whether there exists a global macrovariable with a given name? */
%put _global_;



/* 11.4 Write a macro that will find the maximum of all numerical variables from all data sets from a given library. */
```

```
%macro findmax(lib);

 proc sql;
  *create table with set names and their variable names;
  create table p as
  select memname, name
  from dictionary.columns
  where libname="&lib" and type='num'
  ;

  quit;

  data _null_;
   set p nobs=np;
   call symput('set'||compress(_N_),memname);
   call symput('var'||compress(_N_),name);
   call symput('hm',compress(np));
  run;

  %let maximum=.;

  %do i=1 %to &hm; *number of all variables from all sets;
   proc means data=&&set&i noprint; *for every set get the max;
    var &&var&i;
    output out=_amax (where=(_STAT_='MAX'));
  run;

  data _null_;
   set _amax;
   if &&var&i>&maximum then call symput('maximum',compress(&&var&i));
   run;
  %end;
```

```
  %put **Maximal value is &maximum**;



%mend;


%findmax(WORK);



/* 11.5 Write a macro that will remove from a given data set all variables with the names ending with a given
letter. */
data ex5;
input abavdgwera abefc asb dddd;
cards;
1 2 3 4
;
run;


%macro remove(set, l);
 proc sql noprint;
          select name
    into :names separated by ' '
    from dictionary.columns
    where memname=%upcase("&set")
   ;
 quit;


 %let amount = %eval(%sysfunc(countw(&names)));
 %put **&amount**;
 %let toremove =  ;
 %let count = 0;
 %do i = 1 %to &amount;
          %let name = %scan(&names, &i, ' ');
```

```
        %let length = %length(&name);
        %let pos = %eval(%sysfunc(find(name, &l, -&length)));
        %put **&name** **&length** **&pos**;

        %if &length eq &pos %then %do;
                %let toremove = &toremove &name;
                %let count = %eval(&count + 1);
        %end;
 %end;
 %put **&toremove**;

 %if &count > 0 %then %do;
        data new_&set;
                set &set;
                %do i = 1 %to &count;
                        %let var = %scan(&toremove, &i, ' ');
                        drop &var ;
                %end;
        run;
 %end;
%mend;

%remove(ex5, a);
```

/* 11.6 Write a macro with the two parameters lib and dir that will export (as text files) all SAS data sets from

the library lib into the directory dir. The names of the text files (with the extension txt) should be identical

with the names of the relevant data sets. (Hint: one might to use PROC EXPORT.) */

```
%macro export(lib, dir);
  proc sql;
  select distinct memname
  into :sets separated by ' '
```

```
 from dictionary.columns
 where libname=%upcase("&lib")
 ;
 quit;


 %let amount =  %eval(%sysfunc(countw(&sets)));
 %do i = 1 %to &amount;
 %let set = %scan(&sets, &i, ' ');
          proc export data=&lib..&set
           outfile="&dir/&set..txt"
           dbms=dlm;
           delimiter=' ';
          run;
 %end;


%mend;


%export(lab07, /folders/myfolders/SAS/data/lab11);


/* 11.7 Write a macro %howmany(lib,group,val) that will find all data sets from the library
lib that contain the

pair of numerical variables group and val, and will write to the Log window the value (or
values) of the

variable group that is associated with the largest number of distinct values of the variable
val in the found

sets. (We assume that in there can exist some data sets in the library lib without the given
pair of variables

group and val.) */


data a;

input x$ y;

cards;

a 1

a 2

b 1
```

```
b 2
b 3
b 4
;
run;

data b;
input x$ y;
cards;
a 13
c 5
d 1
d 6
d 7
d 8
d 10
;
run;

Options noQuoteLenMax;
%macro howmany(lib, group, val);
  proc sql;
        select distinct memname
        into :names separated by ' '
        from dictionary.columns
        where libname=%upcase("&lib")
  ;
  quit;

   %let amount = %eval(%sysfunc(countw(&names)));

        data all_sets;
              set %do i = 1 %to &amount;
```

```
                        %let setname = %scan(&names, &i, ' ');
                        &setname
                        %end;
                        ;
                by &group;
        run;


 proc sql;
        create table counters as
        select &group, count(distinct &val) as counter
        from all_sets
        group by &group
        ;
 quit;


 proc means data=counters;
   var &val;
   output out=max_group (where=(_STAT_='MAX'));
 run;
%mend;
Options noQuoteLenMax;


%macro howmany(work, x, y);


/* 11.8 Assume that some library has only the sets that contain at least one common variable (they can also have

some non-common variables). Write a macro (with the name of the library being the only parameter of

the macro) that sorts all the sets from the library with respect to the key consisting of all the common

variables (i.e. variables that occur in all sets). The variables in the key should appear in the alphabetical

order. */
```

/* 11.9 Write a macro with the parameter lib that creates a data set in which the names of variables are the names

of all the variables that occur in data sets from the library lib. The values of the variables should be the

names of the data sets in which the variables occur. */

* Task 1;

```
data a;
input x$ y z;
cards;
a 1 2
a 2 3
a 3 3
c 10 0
;
run;

data b;
input y x$ p;
cards;
4 a 0
4 b 2
6 b 1
;
run;

data c;
input aa bb;
cards;
```

```
1 2
;
run;

%macro abc(lib, var1, var2);
proc sql noprint;
  create table sets_name_type as
  select distinct memname, name, type
  from dictionary.columns
  where libname=%upcase("&lib");
  ;

  select distinct memname
  into :memname_group separated by ' '
  from sets_name_type
  where (name="&var1" and type="char") or (name="&var2" and type="num")
  ;
quit;
        %put &memname_group;

        data all;
        set &memname_group;
        by &var1;
        run;

        proc means data=all;
        var &var2;
                class &var1;
        run;
%mend;

%abc(work, x, y);
```

```
proc sql;
describe table dictionary.tables;
describe table dictionary.columns;
describe table sashelp.class;
quit;


* Task 2;
data data;
input x$ y;
cards;
A 5
A 2
A 0
B 1
B 8
C 1
C 3
;
run;

data which;
input x$ k;
cards;
A 3
C 2
;
run;

data _null_;
 set which end=done;
 by x;
```

```
 retain avg 0;
 retain count 1;
 retain obs 0;

 if _n_ = 1 then set data nobs=obs;

 do i = 1 to obs;
        set data (rename=(x=x2)) point=i;
        if (x = x2) then do;
                if (count = k) then do;
                        avg = avg + y;
                        count = 1;
                end;
                else count = count + 1;
        end;
end;
if done then do;
        avg = avg/_n_;
        put avg=;
end;
run;



* Task 3;
%macro split(string);
        %let amount = %eval(%sysfunc(countw(&string)));
        %do i = 1 %to &amount;
        %global word&i;
        %let word&i = %scan(&string, &i, ' ');
        %end;
        %put _global_;
%mend;
%split(i like fruit pancakes);
```

```
* Task 4;
proc sql;
create table car_rentals as
select r.id_client, c.make
from exam.rentals as r
join exam.cars as c
on r.id_car = c.id_car
order by r.id_client
;

create table car_num as
select count(distinct make) as count
from exam.cars
;

create table customers_cars as
select distinct make, id_client
from car_rentals
order by id_client
;

select id_client, count(distinct make) as counter
from customers_cars
group by id_client
having counter = (select count from car_num)
; *these are the clients that rented all available makes;
quit;
```

EXAM :

```
/* Task 1 */
```

```
data aa;
set exam.a;
by x;
array y(10) y1-y10;
array ynames(10) $ ('y1' 'y2' 'y3' 'y4' 'y5' 'y6' 'y7' 'y8' 'y9' 'y10');
array moms(10) _temporary_ (10 * 0); * strictly smaller than mean ;
array woms(10) _temporary_ (10 * 0); * strictly greater than the mean ;

m = mean(of y1-y10);
put y1-y10 m=;

do i = 1 to dim(y);
 if y(i) > m then woms(i) = woms(i) + 1;
 else if y(i) < m then moms(i) = moms(i) + 1;
end;

if last.x then do;
        do i = 1 to dim(y);
        zm = ynames(i);
        mom = moms(i);
        wom = woms(i);
        output;
        end;

        do i = 1 to dim(moms);
         moms(i) = 0;
         woms(i) = 0;
        end;
end;

keep x zm mom wom;
run;
```

```
/* Task 2 */
data b;
obs = floor(ranuni(0) * 20 + 5); /* random number of observations, here I gave 5-25 */
put 'Num of observations: ' obs;

do i = 1 to obs;
missing = floor(ranuni(0) * 2 );

if missing = 1 then z = .;
else z = floor(ranuni(0) * 19 + 1);
 output;
end;
keep z;
run;

data bb;
 set b end=done;
 retain counter 0; /* counts the number of consecutive missing values */
 retain previous 0; /* holds the previous value of z */

 if z = . then do;
  previous = .;
  counter = counter + 1;
 end;
 else do;
            if previous = . and counter = 1 then output;

            if previous = . and counter > 1 then do;
                  tmp = z;
                  z = .;
                  output;
                  z = tmp;
                  output;
```

```
        end;

        if previous ne . then output;

        counter = 0;
        previous = z;
end;

if done then do;
        if z = . and previous = . then output;
        else if z ne . and previous = . then do;
                tmp = z;
                z = .;
                output;
                z = tmp;
                output;
        end;
end;
keep z;
run;


/* Task 3 */
data c;
infile '\Users\ashmitha67\Downloads\k1\c.txt';
input c1 $ n1 c2 $ n2 v $20.; /* since the numer v is long we need to force the length */
array c(2) c1-c2;
array n(2) n1-n2;

if c(1) = 'x' then do;
        x = input(substr(v, n(1), 1), 8.);
        y = input(substr(v, n(2), 1), 8.);
end;
```

```
 else do;
        x = input(substr(v, n(2), 1), 8.);
        y = input(substr(v, n(1), 1), 8.);
 end;

 output;
 put _all_;
 keep x y;
run;

/* Task 4 */

/* Is it possible that a student retook an exam */
proc sql;
 select count(distinct r) from exam.D;
 select o, count(distinct r) from exam.D
 group by o, r
 having count(distinct r) > 1;
run; /* the result is empty, so NO */

/*find the number of persons who have undergone the test in the years in which the person
C was not tested*/
proc sql;
 select count(distinct o) from exam.D;
 select distinct(r), count(distinct o) from exam.D
 where o ne 'C'
 group by r, o;
quit;
```