```
/*_________________Q1_Execute the following SAS macro codes **/

%put DA: &DA;
%put var1: &var1;
%put var2: &var2;
%put t: "&t";
%put t: '&t';

/****or you'd better use the following codes (add . when resolving macro variables)**/
%put DA: &DA.;
%put var1: &var1.;
%put var2: &var2.;
%put t: "&t.";

/*
Comments:

 You'd better use &DA. whenever you resolve the macro variable DA because '.'
indicates the end of macro variable. For example, &DA.ABC-->EmployeeABC, but
&DAABC is incorrect since there is no macro variable  'DAABC' defined above.
*/

/*******SAS codes 1*****************/
proc print data=&DA ;
  var &var1 &var2 salary;
  where salary>&sa;
  title "&t";
run;
%let DA=Manager;
proc print data=&DA ;
  var &var1 &var2 salary;
  where salary>&sa;
  title "&t";
run;

/*******SAS codes 2*****************/
proc print data=Employee ;
  var emp_id dep_id salary ;
  where salary>60000;
title "List of Employees";
run;
proc print data=manager ;
  var emp_id dep_id salary ;
  where salary>60000;
  title "List of Employees";
```

```
run;


/* solution for Q2 (1) */

/*
Comments:

Yes, because two SAS codes are actually the same. The second SAS program is the
resulting codes resolved (or translated) from the first one when you replace the
&macrovariable with the value of macro variable 'macrovariable'.
 */

/* solution for Q2 (2) */
/*
Comments:

The SAS codes 1 is more flexible, because you can simply change the value of the macro
variable  'DA' without changing the main SAS program (PROC Print...) to print different
data set.
*/

/* solution for Q2 (3) */
%let DA=Employee;
proc print data=&DA. ;
  var &var1 &var2 salary;
  where salary>&sa;
  title "List of &DA.s";
run;

%let DA=Manager;
proc print data=&DA. ;
  var &var1 &var2 salary;
  where salary>&sa;
  title "List of &DA.s";
run;

/*
Comments:

&DA.s is resolved as &DA concatenate with 's'
*/

/* solution for Q3 */

 %let memid=E12;
```

```
%let mdepid=D1;

data _null_;
  set WORK.Employee;
  where emp_id="&memid." and dep_id="&mdepid.";
  call symput('m1',tenure);
run;

data _null_;
  set WORK.Manager;
  where emp_id="&memid." and dep_id="&mdepid.";
  call symput('m2',start_since);
run;

%put the tenure of emp_id="&memid." and ‘dep_id="&mdepid." is %trim(&m1.);
%put the start_since of emp_id="&memid." and ‘dep_id="&mdepid." is %trim(&m2.);

/*
 Comments:

Call Symput' will send the value (of a variable for a specified observation) of a
SAS data set to a macro variable. %trim() is a macro function that can cut spaces from
resolved the macto variable
*/

/* solution for Q4  (1) */

  %let mt=1;
  %let ms=0.03;
  %let mo=employee;
  %let newt=newemployee;

 data &newt.;
   set &mo.;
   salary=salary*(1+&ms.);
   tenure=tenure+&mt.;
  run;


/* solution for Q4  (2) */
  %macro getnew(a,b,c,d);
    data &d.;
      set &c.;
      salary=salary*(1+&b.);
      tenure=tenure+&a.;
    run;
```

```
 %mend;

 %getnew(&mt.,&ms.,&mo.,&newt.);


/* solution for Q5 */
 %macro createntables(n,salary_incr_per);
  %do j=1 %to &n.;
   data work.newemployee_&j.;
      set work.employee;
      salary=salary*(1+&salary_incr_per.)**&j.;
      tenure=tenure+&j.;
   run;
  %end;
 %mend;

 %createntables(5,0.03);

 /*  solution for Q6 */
 %macro changebyyear(startyear,endyear,out);
       data temp_1;
          set newemployee_&startyear.;
          rename salary=salary_1 tenure=tenure_1;
       run;

       data temp_2;
          set newemployee_&endyear.;
          rename salary=salary_2 tenure=tenure_2;
       run;

       proc sort data=temp_1;
          by emp_id;
       run;

       proc sort data=temp_2;
          by emp_id;
       run;

      data &out.;
          merge temp_1 temp_2;
          by emp_id;
          salaeary_increase=salary_2-salary_1;
          tenure_increase=tenure_2-tenure_1;
          keep emp_id salaeary_increase tenure_increase;
        run;
 %mend;
```

```
%changebyyear(2,4,diff24);

 /* solution for Q7 */
%macro printdata(D);
  %let starty=%substr(&D.,5,1);
  %let endy=%substr(&D.,6);
  %let yeardif=%eval(&endy.-&starty.);

 proc print data=&D.;
   title "difference after &yeardif. years";
 run;
%mend;

%printdata(diff24);


/* solution for Q8 */
%macro adjust(adjtable);
 %do j=1 %to 5;
   proc sql;
          select tax_rate,  bonus_rate into :mtax, :mbonus
                 from &adjtable. where year=&j.;
        quit;

        %let adjrate=%sysevalf(&mtax.-&mbonus.);
        %put &adjrate.;

        data newemployee_&j.;
           set newemployee_&j.;
           income=salary*(1-&adjrate.);
        run;
        proc print data=newemployee_&j.;
           title  "Table &j.: Total Rate Adjusted: &adjrate. ";
        run;
  %end;
%mend;

%adjust(Adj_rate);


/* solution for Q8: or you can use the following call execute method */
%macro adjust1(taxr,bonusr,tbnum);
        %let adjrate=%sysevalf(&taxr.-&bonusr.);
        %put &adjrate.;
```

```
        data newemployee_&tbnum.;
            set newemployee_&tbnum.;
            income=salary*(1-&adjrate.);
        run;
        proc print data=newemployee_&tbnum.;
           title  "Table &tbnum.: Total Rate Adjusted: &adjrate. ";
        run;

%mend;

data _null_;
  set Adj_rate;
  code=compress('%adjust1('||tax_rate||','||bonus_rate||','||year||')');
  call execute(code);
run;


/* solution for Q9 */
%macro createnew(agelist, inputd, outputd);
  proc sql;
        select code into :mcode separated by ';' from sas_code;
  quit;
  data &outputd.;
        set &inputd.;
        &mcode.;
        where age in (&agelist.);
   run;
%mend;
%let a=%str(30,40);
%let b=coffee_new;
%let c=coffee_result;
%createnew(&a., &b., &c.);


/*
 Comments:

  The macro program will be resolved into the following SAS codes:

  data coffee_result;
       set coffee_new;
       Cups_Per_Week_ms=(Cups_Per_Week=.);
       income_ms=(income=.);
       Own_Home_ms=(Own_Home='');
       where age in (30,40);
run;
```

The %str() will mask the comma in '30,40' because the comma will be confused with the parameter delimiter (which is also the comma) in SAS macro program.
*/