```
/******SAS code solution for Q1 – 1 and 2******/
libname C '.';

data C.account;
 infile 'C:\coursedec\course_1\hkdata\Account.txt';
 input Account_ID $ Credit_Limit Account_Balance Open_Date: date9.;
 format Open_Date date9.;
RUN;

/******SAS code solution for Q1-- 3******/
data UT_SAS;
 set C.account;
 file '.\UT_export.txt';
 utilization=Account_Balance/Credit_Limit;
 put Account_ID Credit_Limit Account_Balance Open_Date utilization;
RUN;

/*
Comments:

The statement libname C '.' means you create a library that uses the current folder
as the physical location. The current folder is selected from the status bar in SAS
window. The 'infile/input' statement is used for reading data, and the 'file/put'
statement is used for writing data.

*/

/******SAS code solution for Q2******/
data measure_1;
 infile 'C:\coursedec\course_1\hkdata\measure_body_1.txt' DSD;
 input ID HEIGHT WEIGHT GENDER $ AGE;
RUN;

/*or use**/
data measure_1;
 infile 'C:\coursedec\course_1\hkdata\measure_body_1.txt' dlm=',';
 input ID HEIGHT WEIGHT GENDER $ AGE;
RUN;

/*
Comments:

You can use either DSD or DLM=',' option in this case. But DSD has the following functions
(1) Mask quotation marks when reading the raw data into a SAS data set
(2) Can jump to read the next column if there is no values between two consecutive delimiters,
and treat the current field as missing value
*/
```

```
/******SAS code solution for Q3******/
data measure_2;
 infile 'C:\coursedec\course_1\hkdata\measure_body_2.txt' DLM='|' DSD ;
 length ID 8. name $20. HEIGHT 8.  WEIGHT 8. GENDER $2. AGE 8.;
 input ID name $ HEIGHT WEIGHT GENDER $ AGE;
RUN;


/*
Comments:

You must use DLM and DSD in this case. DLM is for specify the delimiter and DSD is for
correctly reading missing values if there is no values between two consecutive '|'. Also, use length
statement to set the maximum length for the 'name' column because the default length is 8, which
is less than the length of actual data

*/

/******SAS code solution for Q4******/
data measure_3;
 infile 'C:\coursedec\course_1\hkdata\measure_body_3.txt' missover truncover;
 length ID 8. LASTNAME $16. DOB 8.  HEIGHT 8. WEIGHT 8. GENDER $10. AGE 8.;
 input ID  LASTNAME $ DOB: mmddyy8.  HEIGHT WEIGHT GENDER $ AGE;
 format DOB mmddyy10.;
RUN;

/*or use**/

data measure_3;
 infile 'C:\coursedec\course_1\hkdata\measure_body_3.txt' DLM=' ' DSD;
 length ID 8. LASTNAME $16. DOB 8.  HEIGHT 8. WEIGHT 8 GENDER $10. AGE 8.;
 input ID  LASTNAME $ DOB: mmddyy8.  HEIGHT WEIGHT GENDER $ AGE;
 format DOB mmddyy10.;
RUN;

/*
Comments:

For the first solution
You must use DLM=' ' and DSD in this case. DLM is for specify the delimiter and DSD is for
correctly reading missing values if there is no values between two consecutive ' '. Also, use length
statement to set the maximum length for the 'name' column because the default length is 8, which
is less than the length of actual data.

For the second solution
You can also use missover and truncover options without DLM and DSD. This means you use the
sequential reading method that use ' ' as the
default delimiter.

*/
```

```
/******SAS code solution for Q5******/
data callcenter_data;
   infile 'C:\coursedec\course_1\hkdata\call_information.txt'
           DSD missover truncover firstobs=3 obs=max;
   length Date Time ANI MDN ClientNum Ans1 Ans2  Ans3  Ans4
           MsgInd  Start_time  End_time  Router_Call_Key $20. ;
    input Date $ Time $ ANI $ MDN $ ClientNum $ Ans1 $ Ans2 $
           Ans3 $ Ans4 $ MsgInd $ Start_time $ End_time $ Router_Call_Key $ ;
RUN;


/*
Comments:

DSD option here can realize both DLM=' ' and correctly reading
missing values if there is no values between two consecutive ' '.

The option firstobs=3 read from the third record and obs=max (default) reads until EOF.

Also, use length statement to set the maximum
length for the some columns because the default length is 8, which is less than the length of actual
data.


*/



/******SAS code solution for Q6******/
data books;
  infile 'C:\coursedec\course_1\hkdata\books.txt';
  input book $1-9 sales 12-16 type $18-30 salesdate: mmddyy10.;
  format salesdate mmddyy10.;
RUN;


/******or use******/
data book1s;
 infile 'C:\coursedec\course_1\hkdata\books.txt';
 input @1 book $9. @12 sales 5. @18 type $13. salesdate: mmddyy10.;
 format salesdate mmddyy10.;
RUN;
```

```
/******SAS code solution for Q7******/
data info;
  input id sex $ @@;
  point=3;
datalines;
1 M 2 M 3 . 4 F 5 F 6 . 7 F
8 M 9 .
10 M 11 F 12 F 13 . 14 . 15 M
;
run;

/*
Comments:

We should use the @@ (strongly hold the row)
Since these data are on different lines without specific rule, we must use the @@ to strongly hold the line, force SAS to read until no value can be read
*/


/******Solution for Q8******/

/*
Comments:

This is an example of using '@' in data step. The symbol '@' stands for 'weakly hold the row' in data step programming. You can use '@' to conditionally read raw data based on the values of the field that is first read (rather than read in the order of fields in raw data).

In this example you first read the column 'type' (this field is at the tail of each row, so you use @ to hold the row to avoid changing line by SAS. After the field 'type' is read, you judge its value to determine if other columns should be read, i.e. if type=1, then do not read others, otherwise read other fields 'id','age' and 'weight' according to their positions in the raw data.
*/

/******SAS code solution for Q9******/
DATA SHORTWAY;
   INPUT ID 1-3 @4 (Q1-Q5)(1.) @9 (Q6-Q10 HEIGHT AGE)(2.);
DATALINES;
1011132410161415156823
1021433212121413167221
1032334214141212106628
1041553216161314126622
;
RUN;

PROC PRINT DATA=SHORTWAY;
  TITLE 'Example';
RUN;
```

```
/*
Comments:

In data step the statement V1-V7) (1.) expresses that v1, v2.. V7 are read in the same way
(length=1).

*/



/******SAS code solution for Q10******/
data address;
  infile 'C:\coursedec\course_1\hkdata\rent_people.txt';
  length cstnumb cstname pstnumb pstname $20.;
  input cstnumb $ cstname & $  pstnumb $ pstname & $ ;
RUN;


/*
Comments:

 The & (ampersand) is used as format modifier in the input statement to read character values
containing embedded blanks in list input approach. This is to tell SAS to read data until it meets
two consecutive spaces/blanks, the defined length of the variable, or the end of the input line.

*/.

/******SAS code solution for Q11******/
data emp_info;
  infile  'C:\coursedec\course_1\hkdata\businessacc.CSV' DSD firstobs=2 obs=max missover
          truncover;
  length id $20. num Bureau $20. tradeline $20. open_date 8. acc_num 8. limited 8.
          Acc_banlance 8. Am_pastdue 8. monthly_pay 8. acc_name $20. acc_type $20.;

  Input  id $ num Bureau $ tradeline $ open_date: mmddyy10. acc_num limited
         Acc_banlance Am_pastdue monthly_pay acc_name $ acc_type $;
  format open_date mmddyy10.;
RUN;
```