CSCI 360-1 **Assignment 10 – Packed Decimal** **Fall 2017**

Final Exam Project
200 exam points

MORE INFORMATION AND EXAMPLE OUTPUT TO COME! WATCH FOR ANNOUNCEMENTS!

This program, your final exam project, will give you experience in manipulating numbers in packed decimal format. You will NOT be using XDECI or XDECO in this assignment. ALL numbers being manipulated or used in arithmetic must be handled using packed decimal.

As a starting point, copy your ASSIGN9 program and rename your PDSE member ASSIGN10. The first CSECT in this program will be named ACCOUNTS. The one external subroutine will be named PROCACCT.

Input Data

An unknown number of 80-byte records in data set: KC02322.CSCI360.DATAFA17(DATA10)

Input Format

Each 80-byte record will hold the account holder's name followed by two numbers in zoned decimal format (EBCDIC). An example record in the data set could look as follows:

```
MUNSTER, HERMAN          0003233750525
```

The first 25 bytes is the account holder's, or customer's, name in EBCDIC. Following the name is the account balance in dollars with a range of –$9,999,999.99 to $9,999,999.99. Following the account balance is the interest rate percentage in the range of 0% to 30.00%. The two number fields are in zoned decimal format. The remaining 42 bytes of each 80-byte record are spaces.

Processing

Inside the read loop, add 1 to the customer counter and immediately call an external subroutine named PROCACCT to process the customer account. Pass the external subroutine PROCACCT the following parameters:

1) input buffer
2) packed decimal field into which you will sum the new account balances you calculate for each customer

After the loop ends, all customer records have been processed and a detail line has been printed for each, print a line displaying the number of customers, a second line displaying the sum of the new account balances you have calculated for each customer, and, finally, a third line displaying the average new account balance. The last two lines are dollar amounts and should be fully edited as such.

Calculations

1. Equation to calculate interest amount:

```
    if original account balance > $0.00
```

```
        interest amount = original account balance * interest rate
    else
        interest amount = $0.00
    end-if
```

2. Equation to calculate penalty fee:

```
      if original account balance < $0.00
        fee = $35.00
    else
        fee = $0.00
    end-if
```

3. Equation to calculate new account balance:

```
      new account balance = original account balance + interest amount + fee
```

   In the above calculation, you will either be adding 0 for the fee or -35.00 for the fee. (See hint #6 below).

Notes for Assignment 10

1. Define your detail line used for each customer in the subroutine's storage.
2. Fully edit each of the four dollar amounts, i.e., old account balance, interest amount, fee, and new account balance, as shown in the sample output.
3. Spread the five items to be printed across the entire 133-byte detail line.
4. XPRNT the detail line, double-spaced, in the subroutine just before you return to the caller.
5. ALL arithmetic must be done using packed decimal arithmetic, including the counter.
6. Round all calculations to two decimal places.
7. Print $0.00 for any dollar amounts that are zero.
8. To compare two packed decimal fields, use CP, compare packed. With this you can use the extended mnemonics BE, BL, BNL, BH, and BNH.

Hints

1. Never MVC packed decimal numbers; only use ZAP.
2. To add 1 to a packed decimal field, use a literal: =PL1'1'
3. Never let lengths default using packed decimal instructions.
4. Consider the interest rate percentage as having four decimal points when doing the MP to calculate the interest dollar amount. For example: 12.05% is actually .1205
5. For the fee, you want to print $35.00 but be sure to SUBTRACT it when computing the new account balance. For example, if a customer's balance is -$45,055.21, their new balance would be -$45,090.21.
6. Also for the fee, it might be smart to have a packed decimal field to hold the fee in storage. If the account balance is positive, ZAP packed 0 into it. If not, ZAP packed -35.00 into it.

General Structure of PROCACCT

Immediately upon entering PROCACCT, but after entry linkage, de-reference the two parameters passed in.

Define a 7-byte packed field in PROCACCT's storage perhaps named PNEWBAL. This field should be reset to 0 each time you enter the subroutine.
After you have packed the original account balance into a 5-byte packed field named perhaps named PACCTBAL, you can immediately edit it into the print line field. Because this could be a negative amount, you need to add an extra byte in the output field for the negative sign.

The largest number that can fit in a 5-byte packed field is 99 99 99 99 9F, or $9,999,999.99 with full editing and a "floating" dollar sign. That is 13 bytes! Add one more byte for the negative sign in case it IS a very large negative number. That is 14 bytes! By the way, only two fields in your output line will potentially require a negative sign, the current account balance (the first numeric output field following the customer's name) and the new account balance (the fourth numeric output field on the far right).

When editing it, use the edit pattern: =X'4040206B2020206B2021204B2020'

The "extra" X'40' at the beginning of this edit pattern will be for the negative sign IF the original account balance is more negative than -$999,999.99.

Next, immediately determine if the original account balance is negative or positive using Compare Packed (CP). In this instruction, you can use a packed decimal literal of =PL1'0'.

If it is negative, immediately decrement register 1 by 1 (BCTR 1,0) once again and place your negative sign using MVI. Second, ZAP -35.00 to a 3-byte packed fee amount field perhaps named PFEEAMT in PROCACCT's storage, and, third, move 0 to PINTAMT (do not worry about calculating interest for a negative balance!).

If it is positive, ZAP 0 to PFEEAMT a packed fee amount field and calculate PINTAMT using what was shown to you in class.

After this if-else conditional structure is done, fully edit both PFEEAMT and PINTAMT into their respective output fields in the print line. Both will require a floating dollar sign and a decimal point, of course. The second will require not only a floating dollar sign and a decimal point but also commas. Note that neither will require a negative sign.

Next, add the packed original account balance, fee amount, and interest amount into PNEWBAL in PROCACCT's storage. As stated above in the second paragraph of this section, be sure PNEWBAL is reset to 0 before you do any adding to it.

```
         AP    PNEWBAL(7),PACCTBAL(5)  ADD ORIGINAL ACCT BAL TO NEW ACCT BAL
         AP    PNEWBAL(7),PFEEAMT(3)   ADD FEE AMT (THIS WILL ADD EITHER 0     *
OR -35.00)
         AP    PNEWBAL(7),PINTAMT(5)   ADD INT AMT (THIS WILL ADD EITHER 0 OR
*                                      THE CALCULATED INT AMT)
```

Now edit PNEWBAL into the print line. If it is a negative amount, be sure to decrement register 1 again and place your negative sign. Its edit pattern will also require two X'40' at the beginning but note that PNEWBAL is declared as a 7-byte packed field and not 5-bytes as the original account balance field was.

Now, you are ready to XPRNT your detail line.

Finally, add PNEWBAL to your summation field, pointed at by register 3 if you passed in this sum field as the second parameter.

Now fly home!

As before, submit you single .txt file on Blackboard for grading.