CSCI 360 **Assignment 9 – Subprograms and Linkage** **Summer 2021**
200 points

This assignment adds four external subprograms to your Assignment 8 program. Each subprogram will require at least one parameter passed to it.

To start, copy your `ASSIGN8` PDSE member and name the new PDSE member `ASSIGN9`.

`PAYROLL` will "drive", or control, the processing of the payroll information to create the ***exact same report*** you created in Assignment 8.

The input data set is

```
DSN=KC02322.CSCI360.DATASU21(DATA9),DISP=SHR
```

Note that ALL records must be read in the subprogram `BUILDTBL` described below.

External Subprograms

Implement each of the following subprograms ONE AT A TIME! Get it working before moving to the next.

- `BUILDTBL`

  Parameter List:

  ```
  BTPARMS  DC    A(PFWHPCT,PSWHPCT,EMPTBL,PEMPCTR)
  ```

  Functionality:

  This subprogram will read the input file and, for the first record, will pack the federal and state withholding percentages into the storage of `PAYROLL` to be used later. To do this, both `PFWHPCT` and `PSWHPCT` must be passed to `BUILDTBL` as parameters. For the remaining records, place the character (employee name) and packed decimal fields (employee ID, hourly pay rate, hours worked, deductions and bonus) for each employee into a table defined in the main program's storage, named `EMPTBL` with 120 entries. Each entry of the table should only be as long as absolutely necessary to store an employee's data. The table should be defined in the storage of the main program ***immediately following*** the 18-fullword register save area and, like `PFWHPCT` and `PSWHPCT`, must be passed into `BUILDTBL` as a parameter.

- `PROCTBL`

  Parameter List:

  ```
  PTPARMS  DC    A(PFWHPCT,PSWHPCT,EMPTBL,PEMPCTR)
  ```

  Functionality:

  This subprogram will be the most complex. It will process the employee data ***stored in the table*** in a loop creating the exact same report as created in Assignment 8. It will call `CALCNPAY` when necessary

for each employee entry in the table.

Keep running totals in this subprogram and, when the table and all employees have been detailed, print the totals page and, when ready, call CALCAVG to calculate each of the averages in turn. Before calling the subprogram, move the total that needs averaging to a generic 7-byte packed decimal field named PTOTAL. When the subprogram CALCAVG completes its work and returns, the average should be stored in the generic 6-byte packed decimal field named PAVG.

Note that, if a variable is declared in a program (for example, PFEDWITH in PAYROLL) that calls a subprogram (PROCTBL) that needs that variable as a parameter and then THAT subprogram (PROCTBL) calls another subprogram (CALCNPAY) that needs the same variable as a parameter, you need to do something special. First, dereference that parameter in the first subprogram (PROCTBL) using the standard LM instruction. After doing this, the address of that variable (PFEDWITH) is in one of the registers. To prepare to pass that same address to the subprogram (CALCNPAY) of the subprogram (PROCTBL), store the pointer register's contents (the one that points to PFEDWITH, for example) in the parameter list defined in the subprogram's (PROCTBL's) storage that will be used to call the subprogram (CALCNPAY).

In the first subprogram's (PROCTBL's) storage, declare:

```
CPPARMS  DC    A(PEMPGPAY)  <- declared in PROCTBL
         DC    A(PEMPNPAY)  <- declared in PROCTBL
         DC    A(0)         <- declared in PAYROLL and passed into PROCTBL
         DC    A(PFEDWITH)  <- declared in PROCTBL
         DC    A(0)         <- declared in PAYROLL and passed into PROCTBL
         DC    A(PSTWITH)   <- declared in PROCTBL
```

Then, when you enter PROCTBL and do the LM, store the register that points at PFWHPCT at CPPARMS+8 and the register that points at PSWHPCT at CPPARMS+16 before calling CALCNPAY.

You will have to do something similar with PEMPCTR in the parameter list named CAPARMS to prepare to call CALCAVG.

Additionally, after you ED the PEMPCTR into the print line after the loop in PROCTBL, shift it 2 decimal digits to the left to add two decimal places so that it and the total you're going to divide by it in CALCAVG will both start out with the same number of decimal places BEFORE calling CALCAVG. Of course, only shift it only once and then use it to call CALCAVG as many times as necessary.

- CALCNPAY

  Parameter List:

  ```
  CPPARMS  DC    A(PEMPGPAY,PEMPNPAY,PFWHPCT,PFEDWITH,PSWHPCT,PSTWITH)
  ```

  Functionality:

  This external subprogram will calculate the employee's two withholding amounts and the employee's net pay amount.

- CALCAVG

  Parameter List:

```
CAPARMS  DC    A(PTOTAL,PEMPCTR,PAVG)
```

  Functionality:

  This external subprogram will calculate a ***single*** average. It must be called each time an average needs to be calculated.

Notes

- PAYROLL calls BUILDTBL first and then PROCTBL.

- PROCTBL calls subprogram CALCNPAY when necessary and, when it is done processing all of the employees and is ready to calculate and print averages on the totals page, calls CALCAVGS as necessary.

- Use a DSECT for both the input record and the table entry. A DSECT should only be declared above the CSECT in which it is used. The names of a DSECT and all of its fields must begin with a dollar sign.

- Be sure that you use the variable names provided for the previous assignment. For the fields' corresponding input fields, replace the beginning P with I, for their corresponding output fields, replace the beginning P with O, for their corresponding tablized fields, replace the beginning P with T, and, for their corresponding DSECT fields, replace the beginning P with a dollar sign ($).

- Note that you will not need all of the provided variable storage in the main program and in every subprogram. Decide which need to be declared in which program or subprogram. For example, the input record need only be declared in BUILDTBL. Variables named PEMPLID, PHRPAY, PHOURS, PDEDUCT and PBONUS will now all be packed into and stored in an entry of the table. A DSECT should be defined for a single table entry.

- Once again, pay close attention to the exact output provided and what it looks like to help you build your program.

- Note that you ***will*** get addressability errors in PAYROLL when you declare your table. Be sure your 18-fullword save area to the top of storage (right after the ORG and DC statements that immediately follow the LTORG) and add the following immediately after standard entry linkage for PAYROLL only:

```
LA     11,4095(,12)      POINT R11 4095 BYTES BEYOND R12
LA     11,1(,11)         POINT R11 4096 BYTES BEYOND R12
USING  PAYROLL+4096,11  ESTABLISH R11 AS SECOND BASE REG
LA     10,4095(,11)      POINT R10 4095 BYTES BEYOND R11
LA     10,1(,10)         POINT R10 4096 BYTES BEYOND R11
USING  PAYROLL+8192,10  ESTABLISH R10 AS THIRD BASE REG
```

  This establishes register 11 as a second base register that takes over from register 12 when your program is longer than 4095 bytes and then establishes register 10 as a third base register that takes over from register 11 when your program is longer than 8192 bytes. Because of this, make sure you don't use registers 10 or 11 in your code. All of this is because your table is 6,360 bytes long!

- As you were taught, you are ***not allowed*** to reference a variable that is defined in a different `CSECT` by name! ASSIST might not flag this as an error but it is. You may only use parameters and registers as pointers to reference variables in other `CSECT`s.

- The output from this program will match that from your Assignment 8 program exactly!

- Put your `CSECT`s and their subprograms in the following order in your file:

```
PAYROLL
BUILDTBL
PROCTBL
CALCNPAY
CALCAVG
```

Document your program completely according to the ***CSCI 360 Documentation and Coding Guidelines*** in Course Documents on Blackboard and submit your single .txt file on Blackboard as before.