Assignment 2

Part I: Programming Assignments

1. Create a ‘Part_Structure’ table as follows (key attributes are underlined): (5 points)

Part_Structure Table:

MAJOR_P	MINOR_P	QTY
P1	P3	2
P1	P2	4
P2	P4	1
P2	P3	3
P3	P5	9
P4	P5	8
P5	P6	3

2. Write a Java program (use JDBC to connect to the database) that implements the following function (written in **pseudo** code) that prints out the subparts of a part using a depth-first search, and the branch with a smaller part number will be searched first (where there is more than branch): (20 points)

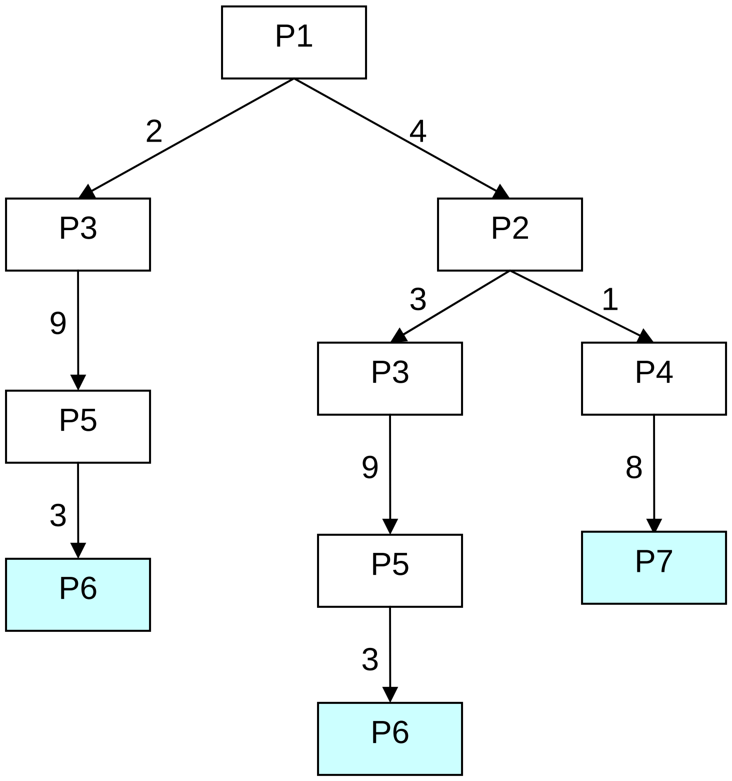
```
CALL RECURSION ( GIVENP# ) ;
RECURSION: PROC ( UPPER_P# ) RECURSIVE ;
        DCL UPPER_P# ... ;
        DCL LOWER_P# ... INITIAL ( '      ' ) ;
        EXEC SQL DECLARE C CURSOR FOR
                SELECT MINOR_P#
                FROM   PART_STRUCTURE
                WHERE  MAJOR_P# = :UPPER_P#
                AND    MINOR_P# > :LOWER_P#
                ORDER  BY MINOR_P# ;
        print UPPER_P# ;
        DO "forever" ;
                EXEC SQL OPEN C ;
                EXEC SQL FETCH C INTO :LOWER_P# ;
                EXEC SQL CLOSE C ;

                IF no "lower P#" retrieved THEN RETURN ;      END IF ;

                IF "lower P#" retrieved THEN
                CALL RECURSION ( LOWER_P# ) ;
                END IF ;
        END DO ;
END PROC ;
```

Given the value of the input parameter ‘P1’, it should print out the following sequence (in the **exact** same order) for the table in Q1: **P1 P2 P3 P5 P6 P4 P5 P6 P3 P5 P6**

3. Please further expand the function you implemented in (2) to allow the printing of the number of each **unique** “leaf part” in a sub-parts-structure tree as the example shown below: (20 points)



In this example, given the part ‘P1’ and its sub-parts-structure tree, there are 2*9*3+4*3*9*3 = 378 ‘P6’ parts and 4*1*8=32 ‘P7’ parts. Therefore, given input value ‘P1’, the function should print out the following (you **need** to create a table with the data above to test your code):

```
P6  378
P7  32
```

4. Implement the above two functions in PL/PGSQL. (22 points)

   See Appendix for PL/PGSQL tutorial.

5. Write a Java Program to simulate an interactive SQL interface like psql. It should allow the user to query and update the tables in the database. Make sure your program prints out the query result (with column headings) or the update status (e.g., the number of rows deleted/updated/inserted) (10 points)

6. (**Bonus**) Write a Java program to simulate a student information query interface where input can be provided for **one or more** of the following fields: student name, **and/or** student number, **and/or** DoB. Assume information about students are stored in a table Students(Snumber, Sname, DoB). The output should be all the qualified student records. A graphical user interface is NOT required. (8 points, **no** partial credits will be awarded. Hint: Dynamic SQL)

You need to:

- ✓ Make sure your source code is compilable and executable.
- ✓ Submit to Canvas a softcopy of the function code with detailed comments, the commands to execute your functions, the execution results (screenshots showing your blazerID), and your DB web user name and password.

Appendix:

1. PL/PGSQL tutorial
https://www.postgresql.org/docs/13/plpgsql.html (42.1--42.7)

2. Read 'SETOF' and 'Return Next' in PL/PGSQL (42.6.1.2) https://www.postgresql.org/docs/13/plpgsql-control-structures.html

3. An example of recursive function in PL/PGSQL

```
CREATE FUNCTION cp_getitemfullname(int8) RETURNS varchar AS '
DECLARE
itemid ALIAS FOR $1;
itemfullname varchar(255);
itemrecord RECORD;
BEGIN
    SELECT s.* INTO itemrecord FROM supplyitem s  where si_id=itemid;
     itemfullname := itemfullname + itemrecord.si_item;
     IF itemrecord.si_parentid IS NOT NULL  THEN
           itemfullname := cp_getitemfullname(itemrecord.si_parentid) + ''->''
+ itemfullname ;
           RETURN itemfullname;
     ELSE
           RETURN itemfullname;
     END IF;
END'  LANGUAGE 'plpgsql'
```