1. [Pascal’s Triangle](https://en.wikipedia.org/wiki/Pascal%27s_triangle)

***1***

***1 1***

***1 2 1***

***1 3 3 1***

***1 4 6 4 1***

***1 5 10 10 5 1***

***1 6 15 20 15 6 1***

***....***

The rth column of nth row of the Pascal's Triangle is *n! / ( r! x (n - r)! )*. Note that the value of *n* starts from *0* and nth row has *(n + 1)* entries, i.e., *0 ≤ r ≤ n*.

Using JavaScript or PHP, complete the function *pascalTriangle*, that has one parameter- an integer *k*. The function should print first *k* rows of Pascal's Triangle, and the entries of a row must be printed separated by space.

**Input Format**  
The input consists of single integer, k.

**Constraints**

*2 ≤****k****≤25*

**Output Format**

Print first ***k*** rows of the Pascal's Triangle (***n****= 0 to****k****- 1*).

**Sample Input**

***4***

**Sample Output**

***1***

***1 1***

***1 2 1***

***1 3 3 1***

***Explanation***

The first row is n = 0 and r = 0.

The second row is n = 1 and r = 0 to 1.

The third row is n = 2 and r = 0 to 2.

The fourth row is n = 3 and r = 0 to 3.

1. Students and Departments

 A university uses *2* data tables, *Students* and *Departments*, to store data about its students and the departments associated with each major. Write a query to print the respective *department name* and *number of students* majoring in each department for *all* departments in the *Departments* table (even ones with no current students). Sort your results by descending *number of students*; if two or more departments have same number of students, then sort those departments alphabetically by *department name*.

**Input Format**

The *Students* and *Departments* tables are described as follows:

| Students | |
| --- | --- |
| *Column Name* | *Type* |
| STUDENT\_ID | Integer |
| STUDENT\_NAME | String |
| GENDER | Character |
| DEPT\_ID | Integer |
|  | |

where *STUDENT\_ID* is the student's ID number, *STUDENT\_NAME* is the student's name, *GENDER* is their gender, and *DEPT\_ID* is the department ID associated with their declared major.

| Departments | |
| --- | --- |
| *Column Name* | *Type* |
| DEPT\_ID | Integer |
| DEPT\_NAME | String |
|  | |

where *DEPT\_ID* is the department's ID number and *DEPT\_NAME* is the department name.

1. Consider the two functions in PHP below. Will they return the same thing?

function foo1()

{

return {

bar: "hello"

};

}

function foo2()

{

return

{

bar: "hello"

};

}

1. What does the function \_\_construct do in PHP?
2. Which Javascript framework do you think is the best? Why? Compare at least 2
3. What is the difference between an Abstract class and an Interface?
4. What is the keyword ‘static’ used for in PHP?
5. Consider the following PHP code:

$str1 = 'yabadabadoo';

$str2 = 'yaba';

if (strpos($str1,$str2)) {

echo "\"" . $str1 . "\" contains \"" . $str2 . "\"";

} else {

echo "\"" . $str1 . "\" does not contain \"" . $str2 . "\"";

}

The output will be:

*"yabadabadoo" does not contain "yaba"*

Why? How can this code be fixed to work correctly?

1. Write a program in whatever language you are comfortable with based on below.

We have a set of three tables (for each one schema is followed by the sample data):

1) **invoiceheader** (

invoicenum varchar,

invoicedate date,

invoiceamount float

)

invoicenum | invoicedate | invoiceamount

-----------+-------------+--------------

00551198 | 1/1/2014 | $150.5

00551199 | 1/2/2014 | $10

2) **invoicedetail** (

invoicenum varchar,

trackingno varchar,

detailamount float

)

invoicenum | trackingno | detailamount

-----------+--------------------+-------------

00551198 | 1Z2F12346861507266 | $50

00551198 | 1Z2F12346861507267 | $80

00551198 | 1Z2F12346861507268 | $20.5

00551199 | 1Z2F12346861503423 | $10.5

3) **invoicecharges** (

invoicenum varchar,

trackingno varchar,

chargetype varchar,

chargeamount float

)

invoicenum | trackingno  | chargetype | chargeamount

-----------+--------------------+------------+-------------

00551198 | 1Z2F12346861507266 | FRT | $40

00551198 | 1Z2F12346861507266 | FUE | $11

00551198 | 1Z2F12346861507267 | FRT | $65

00551198 | 1Z2F12346861507267 | FUE | $17

00551198 | 1Z2F12346861507268 | FRT | $10

00551198 | 1Z2F12346861507268 | FUE | $5

00551198 | 1Z2F12346861507268 | HAZ | $5.5

00551199 | 1Z2F12346861503423 | FRT | $8

00551199 | 1Z2F12346861503423 | FUE | $2.5

The relationship between tables: the data set contains invoice with overall invoice amount, list of packages uniquely identified by tracking# in detail table and break down of charges for the tracking#. I.e. Freight, Fuel.

Using the data from the tables above program should generate the following reports:

**Report 1.**

Input parameters: from date and to date

For specified time period print all invoices

for each show

invoicenum, invoice date and invoice amount

Print a total line: # of invoices and total amount for the period

**Report 2.**

Same input as above

For specified time period print all invoices where the invoice amount does not match the sum of all detailamount column values for this invoice

for each show

invoicenum, invoice date and invoice amount, detailamount total and the discrepancy amount

**Report 3.**

Same input as above

For specified time period print all tracking#s  where the detailamount does not match the sum of all chargemount column values for this invoice and trackingno

for each show

invoicenum, invoice date, trackingno, detailamount, chargeamount total and the discrepancy amount