ESS201 – C++ Programming
International Institute of Information Technology Bangalore

**A4: Inheritance**
Announcement on Oct 28, 2020 (Wednesday), at 1:30 pm IST
**Submission by Nov 06, 2020 (Friday), at 11:59 pm IST, on Domjudge and LMS**

---

Highlight:
Build from A3:

- Construct a BinomialDistribution class with data members corresponding to the parameters n and p.
  - BinomialDistribution inherits from Histogram in A3.
  - Histogram must have function for finding differences between 2 histograms if their number of bins are the same – use the functions used in A1.
- Input values for n (#trials), and 2 different p (probability of success) values for binomial distributions.
- Print out the distances between 2 histograms of the input binomial distributions computed using measures used in A1.

Details:
**Input:**
n p1 p2
-- n is the number of trials for generating the binomial distributions, p1 is the probability of success in the first distribution, and p2 is the same for the second distribution.

**Output:** The output must provide the following distance measures between the distributions, in the following order:
Manhattan distance , Euclidean distance, Chebyshev distance, Kullback-Leibler (KL) divergence, Jensen-Shannon (JS) distance (i.e. the square root of JS divergence).
KL and JS divergences must be computed using natural logarithms. KL divergence is asymmetric, and can be symmetrised by taking the sum in either direction, i.e. KL(a,b)+KL(b,a).

The distances can be rounded off to 4 decimal points of precision, and must be printed out with comma separation.

Formula for computing the distribution is that of the PMF given in
https://en.wikipedia.org/wiki/Binomial_distribution
$f(k,n,p) = Pr(k; n, p) = Pr(X=k) = {}^{n}C_{k}.\ p^{k}.\ (1-p)^{(n-k)}$,
where k is #successes, n is #trials, and p is the probability of success.
The PMF for the probability distributions would be:

- H1 = [f(0,n,p1), f(1,n,p1), ..., f(n,n,p1)] where k= 0, 1, ..., n
- H2 = [f(0,n,p2), f(1,n,p2), ..., f(n,n,p2)] where k=0,1, ..., n

H1 and H2 can be considered as histograms, and also, as vectors seen in A1.

**Notes**:

- n in the input for #trials must also be considered for determining number of bins. The random variable X takes value from 0 to n, thus having (n+1) bins.
- The histogram must be considered as a visual representation of the probability mass function.

- The distances to be computed between H1 and H2 are Manhattan distance , Euclidean distance, Chebyshev distance, Kullback-Leibler (KL) divergence, and Jensen-Shannon (JS) distance (i.e. the square root of JS divergence), in this specific order.
- The following are invalid inputs: (n <= 0), (p1 <= 0 or p1 >= 1), and (p2 <= 0 or p2 >= 1), the input not having 3 numbers, n is not an integer.
- In case of invalid inputs, the output must be -1, as done in A1.
- Output can be printed out upto 4 decimal places of precision.
- It's important that the distance operation is a friend function in Histogram class, which has to check if the number of bins are the same for the 2 Histogram objects. If they are not then through an error message using std::cerr, and return -1.
- DataSequence class developed in A2 and A3 is not required for this assignment.

**Sample**:
Input:
15 0.5 0.7
Output:
~~1.1415 0.3829 0.1770 2.5419 0.4950~~
~~1.1414 0.3829 0.1769 2.5418 0.4949~~
1.1414 0.3829 0.1769 2.5419 0.4949

Input:
100 0.5 0.7
Output:
~~1.9228 0.3409 0.0867 16.9460 0.8008~~
~~1.9228 0.3409 0.0867 16.9459 0.8007~~
1.9228 0.3409 0.0867 16.9459 0.8008

Input:
0 -1.0 1.2
Output:
-1

Python code for reference:

```
## for the binomial distributions , a and b, represented as numpy arrays of
## size (n+1) for n trials

from scipy.linalg import norm
from scipy.spatial import distance
from scipy.stats import entropy

## Output : ordered as Manhattan distance , Euclidean distance, Chebyshev distance,
## Kullback-Leibler (KL) divergence,
## Jensen-Shannon (JS) distance (i.e. the square root of JS divergence).

dman = norm((a-b),ord=1)
deuc = norm((a-b),ord=2)
dche = norm((a-b),ord=np.inf)
dkl = (entropy(a,b)+entropy(b,a))
djs = distance.jensenshannon(a,b)

print(dman,deuc,dche,dkl,djs)
```