```
(* BIOEN 3070/6070: Introduction to Statistics for Bioengineers *)
```



```
(* In-Class Project 1: Traveling Salesman Problem (TSP) Utah Style *)
```

```
(* General Commands *)
```

```
Clear["Global`*"]
```

```
(* Pascal's Triangle *)
```

```
? Binomial
```

Binomial[n, m] gives the binomial coefficient $\binom{n}{m}$. ≫

```
Column[Table[Binomial[n, k], {n, 0, 5}, {k, 0, n}], Center]
```

```
        {1}
       {1, 1}
      {1, 2, 1}
     {1, 3, 3, 1}
   {1, 4, 6, 4, 1}
{1, 5, 10, 10, 5, 1}
```

```
(* 1. Explain each step in the code that produces Pascal's Triangle. *)
```

```
(* 2. Modify the code to generate Pascal's Triangle of 8 and 80 rows. *)
```

```
(* Read the Dataset "Utah_Driving_Distances.txt" *)

stream =
  "http://www.alterlab.org/teaching/BIOEN3070/assignments/Utah_Driving_Distances.txt";
matrix = Import[stream, "Table"];

(* 3. What do the commands "Dimensions", "TableForm" and "Drop" do? *)

? Dimensions
? TableForm
```

Dimensions[*expr*] gives a list of the dimensions of *expr*.
Dimensions[*expr*, *n*] gives a list of the dimensions of *expr* down to level *n*. ≫

TableForm[*list*] prints with the elements of *list* arranged in an array of rectangular cells. ≫

```
Dimensions[matrix]
TableForm[matrix]
```

```
{10, 10}
```

Name	Name/Number	Arches	Bryce_Canyon	Capitol_Reef	Dinosaur	Koda
Name/Number	Number	1	2	3	4	5
Arches	1	0	277	156	216	265
Bryce_Canyon	2	277	0	120	403	22
Capitol_Reef	3	156	120	0	298	108
Dinosaur	4	216	403	298	0	406
Kodachrome	5	265	22	108	406	0
Monument_Valley	6	157	249	195	309	261
Salt_Lake_City	7	197	281	221	232	293
Zion	8	392	91	201	448	103

```
? Drop
```

Drop[*list*, *n*] gives *list* with its first *n* elements dropped.
Drop[*list*, –*n*] gives *list* with its last *n* elements dropped.
Drop[*list*, {*n*}] gives *list* with its n^{th} element dropped.
Drop[*list*, {*m*, *n*}] gives *list* with elements *m* through *n* dropped.
Drop[*list*, {*m*, *n*, *s*}] gives *list* with elements *m* through *n* in steps of *s* dropped.
Drop[*list*, seq_1, seq_2, …] gives a nested list in which elements specified by seq_i have been dropped at level *i* in *list*. ≫

```
TableForm[Drop[matrix, {1, 2}]]
names = Drop[matrix[[1]], {1, 2}];
TableForm[{names}]
numbers = Drop[matrix[[2]], {1, 2}];
TableForm[{numbers}]
```

Arches	1	0	277	156	216	265	157	197	392
Bryce_Canyon	2	277	0	120	403	22	249	281	91
Capitol_Reef	3	156	120	0	298	108	195	221	201
Dinosaur	4	216	403	298	0	406	309	232	448
Kodachrome	5	265	22	108	406	0	261	293	103
Monument_Valley	6	157	249	195	309	261	0	397	218
Salt_Lake_City	7	197	281	221	232	293	397	0	326
Zion	8	392	91	201	448	103	218	326	0

```
Arches    Bryce_Canyon    Capitol_Reef    Dinosaur    Kodachrome    Monument_Valley    Salt_

1    2    3    4    5    6    7    8
```

```
(* 4. Use the command "Transpose" to test whether
 the dataset "Utah_Driving_Distances.txt" is symmetric. *)
```

```
? Transpose
```

Transpose[*list*] transposes the first two levels in *list*.
Transpose[*list*, {n_1, n_2, ...}] transposes *list* so that the k^{th} level in *list* is the n_k^{th} level in the result. ≫

```
(* 5. How many places are listed in the dataset? *)
```

```
(* 6. How many different possibilities are there
 to visit each place on the list once and only once? *)
```

```
(* 7. Create a list of these different possibilities that calls each place
 by name. Test whether the expected number of possibilities is listed. *)
```

```
? Permutations
```

Permutations[*list*] generates a list of all possible permutations of the elements in *list*.
Permutations[*list*, *n*] gives all permutations containing at most *n* elements.
Permutations[*list*, {*n*}] gives all permutations containing exactly *n* elements. ≫

```
(* 8. Create a list of these different possibilities that calls each place
 by number. Test whether the expected number of possibilities is listed. *)
```

```
(* 9. What is the distance between the 2nd and 7th places? *)
```

```
distances = Drop[Transpose[Drop[matrix, {1, 2}]], {1, 2}];
distances[[2, 7]]
```

```
281
```

```
(* 10. What is the total distance of traveling
 from the first to the last place sequentially? *)
```

```
Sum[distances[[a, a + 1]], {a, 1, 7}]
```

```
2085
```

```
Sum[distances[[
   numbers[[a]],
   numbers[[a + 1]]]], {a, 1, 7}]
```

```
2085
```

```
(* 11. What is the total distance of the first permutation of the places? *)

Permutations[numbers]
Permutations[numbers][[1]]
Sum[distances[[
    Permutations[numbers][[1]][[a]],
    Permutations[numbers][[1]][[a + 1]]]], {a, 1, 7}]
```

A very large output was generated. Here is a sample of it:

```
{{1, 2, 3, 4, 5, 6, 7, 8}, {1, 2, 3, 4, 5, 6, 8, 7}, {1, 2, 3, 4, 5, 7, 6, 8},
 {1, 2, 3, 4, 5, 7, 8, 6}, {1, 2, 3, 4, 5, 8, 6, 7}, {1, 2, 3, 4, 5, 8, 7, 6},
 <<40 308>>, {8, 7, 6, 5, 4, 1, 2, 3}, {8, 7, 6, 5, 4, 1, 3, 2}, {8, 7, 6, 5, 4, 2, 1, 3},
 {8, 7, 6, 5, 4, 2, 3, 1}, {8, 7, 6, 5, 4, 3, 1, 2}, {8, 7, 6, 5, 4, 3, 2, 1}}
```

Show Less | Show More | Show Full Output | Set Size Limit...

```
{1, 2, 3, 4, 5, 6, 7, 8}

2085

permutations = Permutations[numbers];
Sum[distances[[
    permutations[[1]][[a]],
    permutations[[1]][[a + 1]]]], {a, 1, 7}]
2085

(* 12. Compute the total distances of all permutations. *)

permutations = Permutations[numbers];
allTotalDistances = Table[
    Sum[distances[[
        permutations[[b]][[a]],
        permutations[[b]][[a + 1]]]], {a, 1, 7}],
    {b, 1, 8!}];

(* 13. What is the shortest distance? *)

? Min
```

Min[x_1, x_2, ...] yields the numerically smallest of the x_i.
Min[{x_1, x_2, ...}, {y_1, ...}, ...] yields the smallest element of any of the lists. ≫

```
(* 14. Which permutation does it correspond to? *)

? Position
```

Position[*expr*, *pattern*] gives a list of the positions at which objects matching *pattern* appear in *expr*.
Position[*expr*, *pattern*, *levelspec*] finds only objects that appear on levels specified by *levelspec*.
Position[*expr*, *pattern*, *levelspec*, *n*] gives the positions of the first *n* objects found. ≫

```
(* Access Geographical Data in Mathematica *)

(* 15. Select a set of Utah cities. Tabulate their longitudes
 and latitudes. Compute the distances between each pair of cities in
 the set. Find the shortest route among the cities in the set and the
 permutation of the cities that the shortest route corresponds to. *)
```

```
? CityData
```

CityData["*name*", "*property*"] gives the value of the specified property for the city with the specified name.
CityData["*name*"] gives a list of the full specifications of cities whose names are consistent with *name*. ≫

```
cities = CityData[{All, "Utah"}];
cities[[1]]
CityData[cities[[1]], "Coordinates"]
```

```
{SaltLakeCity, Utah, UnitedStates}
```

```
{40.7785, -111.931}
```

```
? GeoDistance
```

GeoDistance[{lat_1, $long_1$}, {lat_2, $long_2$}] gives the geodesic distance in meters between latitude–longitude positions on the Earth.
GeoDistance[pos_1, pos_2] gives the distance between positions specified by position objects. ≫

```
(* Symbolic Computing in Mathematica *)

(* 16. Repeat the TSP solution with the input being a list of variable,
instead of cities and distances. *)
```