The Tasks

- In this assignment, you need to implement a class for a graph with adjacency-list representation. Name your class **`MyGraph`**.
- You also need to define a class for its vertices. Name this class **`MyGraphVertex`**. Each vertex should have a data member for its adjacency list.
  - Using a **multilist** might be better for the actual task, but this is not required. If you use a multilist, you need to define the class of its nodes.
- The graph class has an array containing its vertices. The vertices are indexed from **`0`** to **`N-1`**, where **`N`** is the number of vertices.
- The graph is unweighted and undirected.

The Tasks

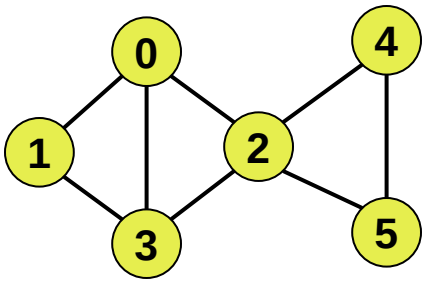
- The graph is initialized with a binary array, which represents an adjacency matrix. You need to convert it to adjacency-list representation in your constructor.
  - The constructor of the graph class takes two inputs: The number of vertices, **`N`**, and a 1-D array (type **`int`**) of size **`N*N`** for the adjacency matrix.
  - You can assume that the input matrix is symmetric.
- Example construction:

```
int x[] = {0,0,1,0,0,0,0,1,1,0,0,0,0,1,0,0};
MyGraph G(4, x); // a graph with 4 vertices
```

The Tasks

- Here is the task: To find a one-stroke-plot path through all the edges.
  - Name this function `OneStroke` (no input argument). It should print out the order of vertices visited in the path.
  - The path should go through each edge exactly once.
  - Use a return value (type `int`) to indicate whether such a path exists: 1 for success and 0 otherwise.
  - Using a **multilist** to represent the graph (optional) helps you identify whether an edge is already in the current path.
  - Use rules that you already know (for doing this by hand) to speed up the process.

Example



Initialization:

{ 0, 1, 1, 1, 0, 0,
1, 0, 0, 1, 0, 0,
1, 0, 0, 1, 0, 0,
1, 1, 1, 0, 0, 0,
0, 0, 1, 0, 0, 1,
0, 0, 1, 0, 1, 0 }

A possible output (not unique): 0 1 3 2 5 4 2 0 3

The Guidelines (Programming Part)

- Allowed programming environment: VS2015 only.
- For simplicity of submission, put the whole class, including the implementation, in **a single header file**.
- You need to write your own `main` function to test your code. You do not need to include this `main` function in your submission.
- STL class templates: The following class templates are allowed: `list` and `deque`.
- Include documentation; this will be part of your grade.
- Demo: Only a randomly selected subset of students; the list will be announced separately after the due date.

Submission

- Use E3 only.
- Name your code **`P4_xxxxxx.h`**, where **`xxxxxx`** is your ID. **Do not** submit your **`main`** function or any file that is not your code (such as the *.sln file). No compressed file (*.zip, *.rar, etc.) accepted.
- Due date: **12/20/2016**. There's a grace period of 3 days with 10% deduction per day. (The deduction kicks in only when you have accumulated more than three days of delay during the semester.)