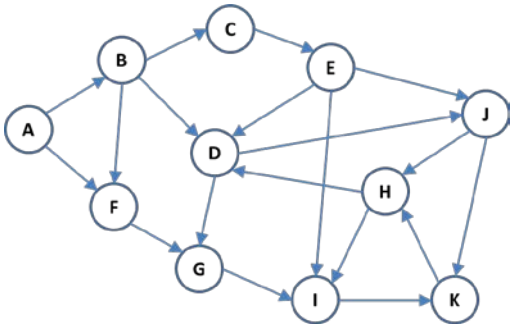
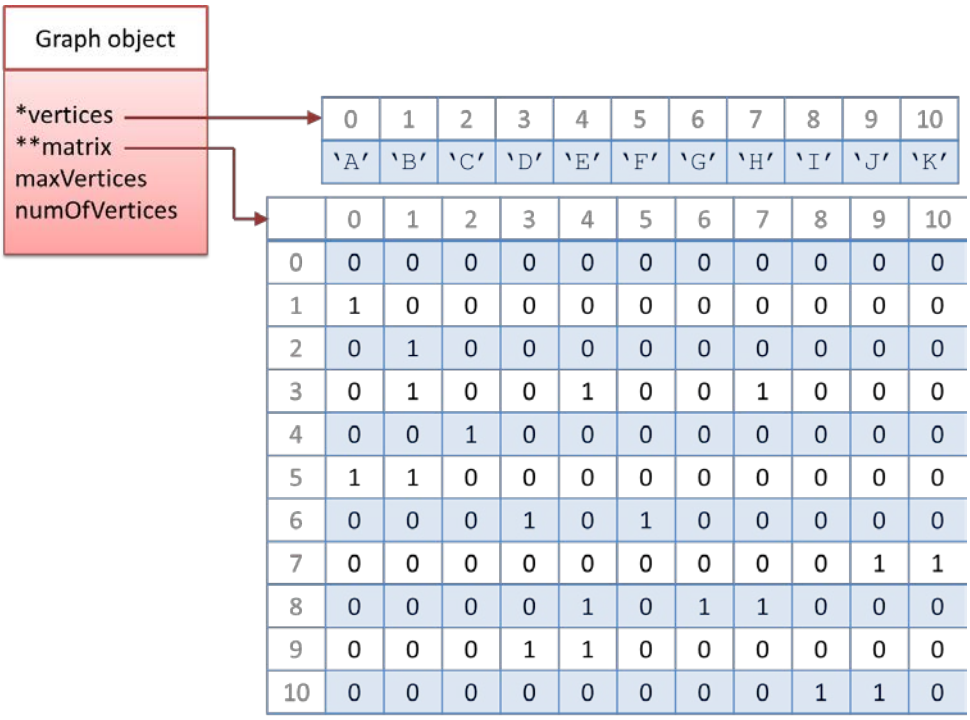
Lab 07 (Part A): Adjacency Matrix

You may work on this lab with other students (**max of 3**). If you do so, make sure you include all names in the header.

For this lab, you need to represent **directed graphs** as **adjacency matrices**. Your **start project** contains the class **Graph**, which creates objects pointing to a one-dimensional dynamic array and a two-dimensional dynamic array. An example is illustrated below.



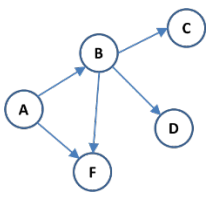
This is how the Graph object represents the graph above:



0	1	2	3	4	5	6	7	8	9	10
'A'	'B'	'C'	'D'	'E'	'F'	'G'	'H'	'I'	'J'	'K'

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0
3	0	1	0	0	1	0	0	1	0	0	0
4	0	0	1	0	0	0	0	0	0	0	0
5	1	1	0	0	0	0	0	0	0	0	0
6	0	0	0	1	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	1
8	0	0	0	0	1	0	1	1	0	0	0
9	0	0	0	1	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	1	0

Example: Vertex B has 3 successors: C, D and F.
Vertex B is at column [1].
If you go down column [1], you will see that rows [2], [3] and [5], which correspondes to vertices C, D, and F, are marked as 1. This means that the **successors** of B are C, D and F.
Let's look at B in row [1]. The only 1 in that row is at column [0], which



means that B has **predecessor** [0], which is vertex A.
Implement the following **member functions** of the **Graph** class as indicated below:

- **Default constructor**
  - Initialize the number of vertices to 0 and the maximum number of vertices (capacity of the array) to MAX_VERTICES.
  - Using pointer **vertices**, declare a dynamic array of characters.
  - How to create a dynamic 2-dimensional array? First, it is important to understand that a 2D array is an array of pointers, where each pointer points to another array, thus creating a table.
    - To start, you need to create the initial array. Since it is a **dynamic** array, you will need to use the **pointer matrix** to create an **array of pointers to arrays of integers**. How to do this? If you are not sure, you need to research it.
    - Then, **for each index** of the dynamic array of pointers you just created, you will need to **initialize a dynamic array of integers**. Use **parenthesis ( )** at the end of your statement so that the array will be automatically populated by 0s.

- **Overloaded constructor**
  - Same implementation as the default constructor, with the difference that the maximum number of vertices is given by the parameter.

- Function **createGraph**
  - This is partially implemented. It reads from the **graph_data.txt** file to retrieve the necessary data to create the graph: the first line shows the number of vertices in the graph, the second line shows the label for each vertex, and the subsequent lines show the successors (or predecessors, depending on whether you are looking at the columns or rows) of each vertex.

```
11
A B C D E F G H I J K
0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0
0 1 0 0 1 0 0 1 0 0 0
0 0 1 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 1 0 1 1 0 0 0
0 0 0 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 0
```

  - You will need to complete the function by implementing **2 FOR loops**, one to populate the **array of characters** with the labels given to the vertices (A, B, C and so on), and the other to populate the **2-dimensional array** containing the 1s and 0s.

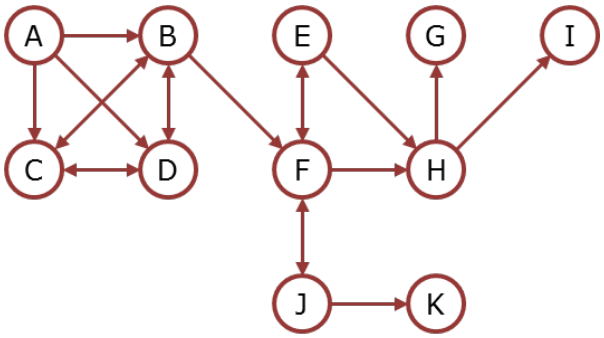
- Function **clearGraph**
  - What does the function need to do? It is not going to destroy the matrix; it only needs to **re-set all the values to 0s**. There is an **efficient way to do this**; you are expected to figure this out.

- **Destructor**
  - You will need to delete all the dynamic data you created

- **Overloaded insertion operator:**
    - The print out of the graph must look as shown in the output below.

The **main** function in the **Main.cpp** file creates a graph and prints it using the insertion operator.

After testing the given graph, create another text file that contains data representing the graph below. Name this file, **graph_data_2.txt.**



Expected output for **graph_data.txt**:

```
  A B C D E F G H I J K
A 0 0 0 0 0 0 0 0 0 0 0
B 1 0 0 0 0 0 0 0 0 0 0
C 0 1 0 0 0 0 0 0 0 0 0
D 0 1 0 0 1 0 0 1 0 0 0
E 0 0 1 0 0 0 0 0 0 0 0
F 1 1 0 0 0 0 0 0 0 0 0
G 0 0 0 1 0 1 0 0 0 0 0
H 0 0 0 0 0 0 0 0 0 1 1
I 0 0 0 0 1 0 1 1 0 0 0
J 0 0 0 1 1 0 0 0 0 0 0
K 0 0 0 0 0 0 0 0 1 1 0


Press any key to continue . . .
```