Lab 10: Graphs

You may work on this lab with another student.

Folder name: A200_L10_YourLastName_YourFirstName ➔If you worked with another student, turn in **ONLY ONE COPY** of the project, named **A250_L10_Yourlastname_Yourfirstname_Otherstudentlastname_Otherstudentfirstname**

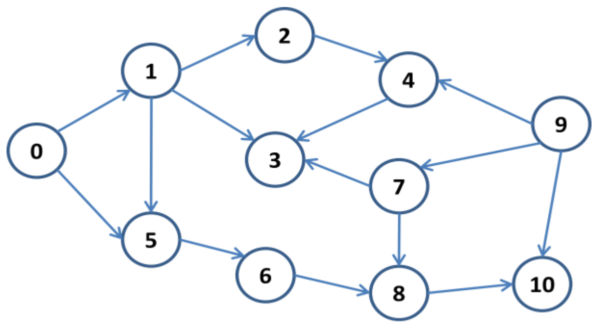
The project contains **four (4) classes**:

- **Node** and **LinkedListType** (in the same file)
  - **Node**
    - Creates nodes for **singly-linked lists**
    - A node contains two elements:
      - **info:** data stored in the node
      - **link:** a pointer to next node
  - **LinkedListType**
    - Creates objects that contain **three** elements:
      - **count**: the number of elements in the list
      - **first**: a pointer to the first node in the list
      - **last**: a pointer to the last node in the list
- **GraphType**
  - Creates graphs implemented as **adjacency lists**. The objects created contain the following:
    - An **int maxSize** storing the **maximum** number of vertices allowed
    - An **int gSize** storing the **current** number of vertices
    - A **pointer graph** to an **array of objects** of type **LinkedListGraph** to create an **adjacency list.**
- **LinkedListGraph**
  - Inherits from the **LinkedListType** class and contains **only** one function
    - **getAdjacentVertices** – Retrieves vertices adjacent to a given vertex

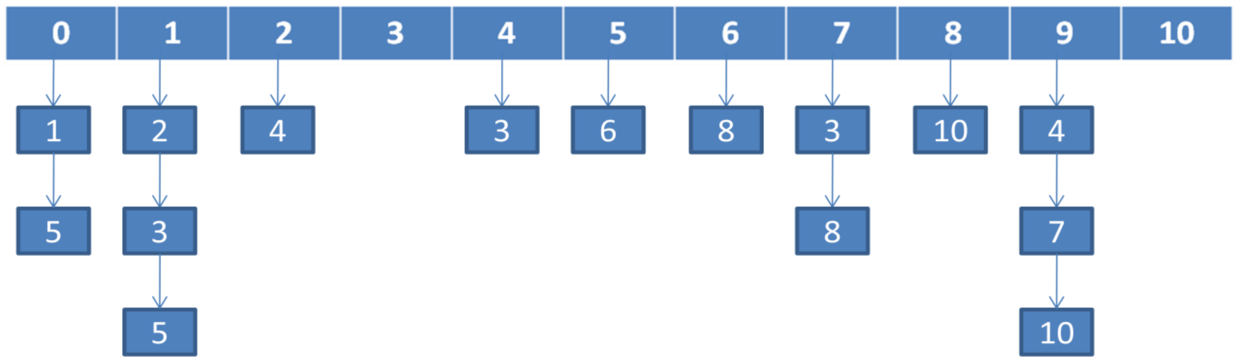
In addition to the classes listed above, the project contains the file **graph_data.txt** which provides the following information, where the first number is the number of vertices in the graph; each subsequent line shows the vertex and it successors. For simplicity, the vertex are labeled 0 to 10.

```
11
0 1 5 -999
1 2 3 5 -999
2 4 -999
3 -999
4 3 -999
5 6 -999
6 8 -999
7 3 8 -999
8 10 -999
9 4 7 10 -999
10 -999
```

This information will allow the program to create the following **graph**:



The graph will be implemented as an **adjacency list**, where successors are inserted in ascending order:



Your job is to implement the following **four (4)** functions in the **GraphType** class and **one (1)** function in the **LinkeListGraph** class. Implement the functions in this order (it will be easier to understand what you need to do):

- Function **getAdjacentVertices**
  - Implement this function in the **LinkedListGraph** class. This class inherits from the **LinkedListType** class.
    NOTE that the member variables of the **LInkedListType** are protected.
  - **Parameters:**
    - An empty **array of integers**
    - An **integer** passed by reference that will store the number of elements in the array.
  - The purpose of this function is to copy all the elements in the list (the adjacent vertices) and insert them in the array that is passed as a parameter. The numbers of elements in the array will be stored in the integer passed by reference.
  - This function will be called by the functions below.

- Function **numberOfSuccessors**
  - Implement the function declaration in the **GraphType.h** class and the function definition in the **Functions.cpp** file, where indicated.
  - The function calls the **getAdjacentVertices** to return the number of successors. NOTE: You will need to create an array of capacity gSize to use as a parameter for the function **getAdjacentVertices**.
  - **Parameter:** an integer storing the index of a vertex.

- Function **printSuccessors**
  - Implement the function declaration in the **GraphType.h** class and the function definition in the **Functions.cpp** file, where indicated.

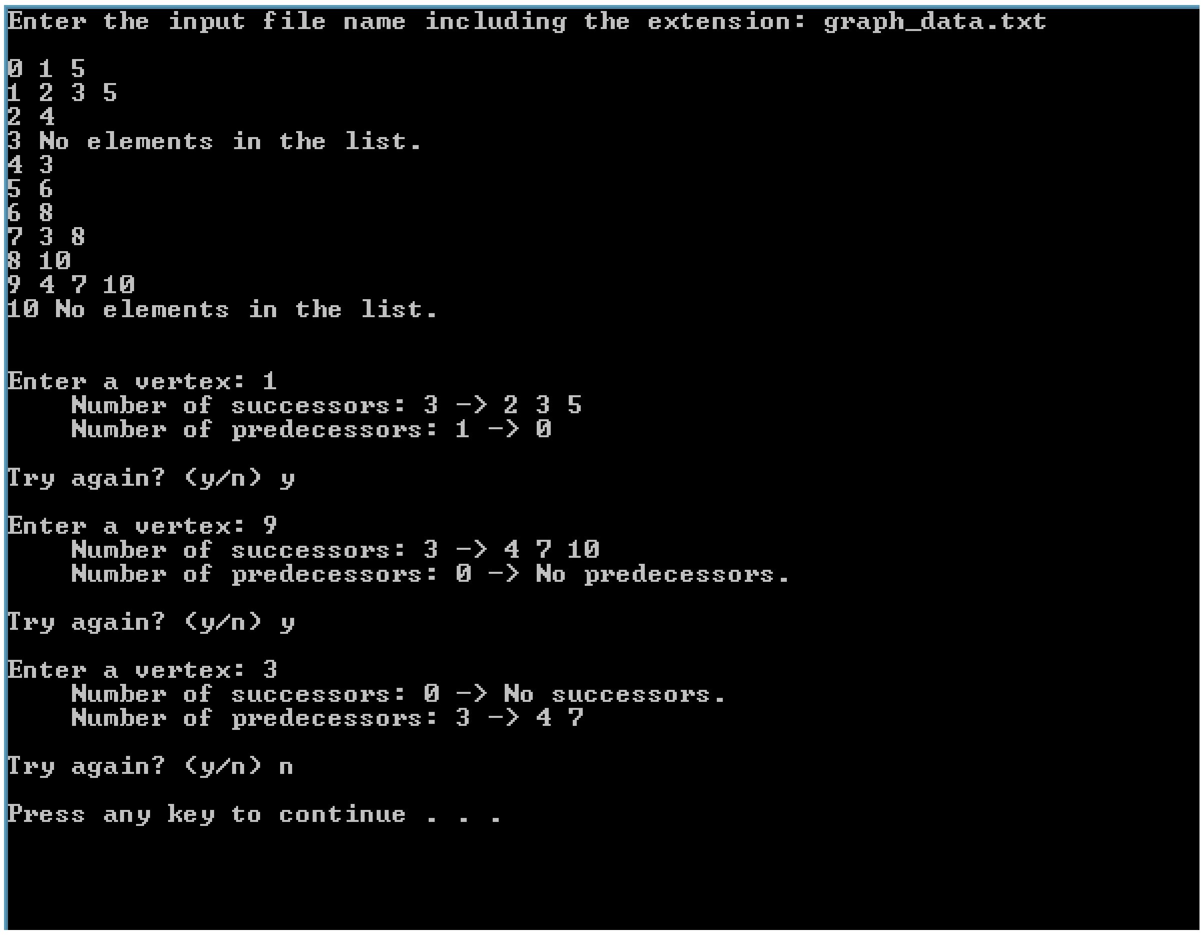
    - The function prints the successors of a given vertex.
    - **Parameter:** an integer storing the index of a vertex.
    - Consider the case when there are no successors and output the message, “No successors.”

- Function **numberOfPredecessors**
    - Implement the **function declaration** in the **GraphType.h** class and the **function definition** in the **Functions.cpp** file, where indicated.
    - The function calls the **getAdjacentVertices** to find all the predecessors of a given vertex. **NOTE:** You will need to create an array of capacity gSize to use as a parameter for the function **getAdjacentVertices**.
    - **Parameter:** an integer storing the index of a vertex.

- Function **printPredecessors**
    - Implement the **function declaration** in the **GraphType.h** class and the **function definition** in the **Functions.cpp** file, where indicated.
    - The function calls **getAdjacentVertices** to retrieve all predecessors of a given vertex and prints the predecessors of a given vertex. **NOTE:** You will need to create an array of capacity gSize to use as a parameter for the function **getAdjacentVertices**.
    - **Parameter:** an integer storing the index of a vertex.
    - Consider the case when there are no predecessors and output the message, “No predecessors.”

The **Main.cpp** file contains implementation to test your functions.

EXPECTED OUTPUT



```
Enter the input file name including the extension: graph_data.txt

0 1 5
1 2 3 5
2 4
3 No elements in the list.
4 3
5 6
6 8
7 3 8
8 10
9 4 7 10
10 No elements in the list.


Enter a vertex: 1
    Number of successors: 3 -> 2 3 5
    Number of predecessors: 1 -> 0

Try again? (y/n) y

Enter a vertex: 9
    Number of successors: 3 -> 4 7 10
    Number of predecessors: 0 -> No predecessors.

Try again? (y/n) y

Enter a vertex: 3
    Number of successors: 0 -> No successors.
    Number of predecessors: 3 -> 4 7

Try again? (y/n) n

Press any key to continue . . .
```