You have unverified email(s). Please click on your name in the top right corner and browse to your profile to send another verification email.

Students:

This content is controlled by your instructor, and is not zyBooks content. Direct ques
content to your instructor. If you have any technical issues with the zyLab submissic
button at the bottom of the lab.

Students:

Section 1.1 is a part of 1 assignment: **Final Exam** Requirements:

Entire class due:

What is a zyBook?

New to zyBooks? Check out a short video to learn how zyBooks uses concise writin
activities, and research-backed approaches to help students learn.

1.1 Question 0

You will be implementing a Breadth-First Search (BFS) and a Depth-First Search (DFS) a
adjacency list. The `AdjacencyList` class inherits from the `Graph` class shown below

```
class Graph {
    protected:
        vector<int> _distances;
        vector<int> _previous;
    public:
        Graph() { }
        virtual int vertices() const = 0; // Return the number
        virtual int edges() const = 0; // Return the number of
        virtual int distance(int) const = 0; // Return the dis
to the vertex passed in
        virtual void bfs(int) = 0;
        virtual void dfs(int) = 0;
};
```

It is up to you how you would like to store the data internally, however, the `AdjacencyL`
adjacency list as discussed in class.

The input file is formatted with the first line being the number of vertices in the graph (la
lines being the edges in a directed graph with the first integer being the source vertex ar
sink vertex.

```
3
0 1
1 2
2 1
2 0
```

Would be a graph with 3 vertices and the edges `{ (0->1), (1->2), (2->1), (2-`
directed edge both from vertex 1 to 2 and vertex 2 back to 1.

It is recommended that you use something similar to the following in the constructor fo

```
    // Read in number of vertices.
    for (unsigned i = 0;i < vertices;++i) {
        // Initialize vertices
    }

    int source, sink;
    while(/* Can still read edges in */) {
        // Read in an edge
        // Add edge to adjacency list (push-back)
    }
```

The `string path(int sink)` function will print out the path from the source to the
format for this is `{source->next->next->sink}` with no whitespace. For example,
to sink `1` would be output as:

```
{0->2->1}
```

For the `dfs()` and `bfs()` functions in your `.cpp` files, annotate the functions with thei
the function definition specify the runtime and space complexity of your implementatio
put the run time of that line of code in your implementation. For helper function calls yo
function definitions.

```
// Overall runtime complexity: O(?)
// Overall space complexity: O(?)
void foo() {
    int x = 5; // O(?)
    int y = bar; // O(?)
}
int bar() {
    return 5; // No annotations necessary
}
```

LAB ACTIVITY | 1.1.1: Question 0

Submission Instructions

Downloadable files

`main.cpp` , `Graph.h` , `AdjacencyList.h` , and `input.tx`

Compile command

```
g++ main.cpp AdjacencyList.cpp -Wall -o a.out
```

We will use this com

Upload your files below by dragging and dropping into the area or choosing

AdjacencyList.h
✓ File added
Remove

AdjacencyList.cpp
✓ File added
Remove

- Graph.h is read-only and has already been provided for you.
- input.txt is not an expected file; check file name and extension. File names are
- main.cpp is read-only and has already been provided for you.

Submit for grading

Latest submission - 2:28 AM on 03/19/20 | Submission passed all tests ✓

☐ Only show failing tests **Downloa**