Project 6

Published | Edit | ⋮

Download this file first: project06_materials.zip.

Complete the provided partial C++ program that will implement a ***Graph ADT Class*** in which an **Adjacency List** is used as the internal representation of the graph vertices and edges.

The lecture notes and textbook provide the source code for basic graph methods and the depth-first search algorithm but you will need to modify this code to complete this assignment.

Required modifications include

– conversion to use the alternate representation of an adjacency list (Method 2 from notes)

– addition of the specified exception detection/handling

– conversion to use the stack and queue containers from the Standard Template Library.

Additional details may be found on subsequent pages of the handout.

<Project06 Files Provided – DO NOT MODIFY OR SUBMIT THESE FILES ** >

graph.h – declares the ***Graph ADT Template*** implemented as linked dynamically allocated nodes

main.cpp – includes the **main()** function test driver for the ***Graph*** class.

makefile – includes all commands required to compile and link your project on ***blackhawk***

Do not modify or submit any of the provided files named above** !!!

****Failure to satisfy these requirements will result in zero credit (0 points) on this assignment.***

All output to the monitor *(stdout) will be performed by the code provided.*

<Project06 FILE YOU MUST WRITE AND SUBMIT ** >

graph.cpp – add your code here to implement Graph functionality (note that the code in this file must not include any INPUT or OUTPUT statements. A PRINT function has already been provided for you (implemented in graph.h) for debugging purposes).

===> Submit only the SOURCE CODE in graph.cpp for grading. <===**

****Failure to satisfy these requirements will result in zero credit (0 points) on this assignment.**

<Project06 Sample Solution>

Use the sample solution and preview script to help you debug your program – your goal is to match the output of the sample solution.

Run the sample solution by typing the following in a **blackhawk** terminal window **:**

/home/work/cpe212/project06/p06 <NameOfSomeTestFile>

You may also use the preview06 script to examine the differences between the output of your project06 executable and that of the sample solution. You must have used make on blackhawk to compile your code first and your current working directory must be the directory where your project06 executable is located. If that is the case, then type the following at the Linux prompt to execute the preview06 script.

/home/work/cpe212data/project06/preview06.bash

<Project06 Include File Constraints **>

All allowed include files within graph.cpp appear below:

#include "graph.h"

Use of the Standard Template Library containers or container adapters (other than stack and queue) is not allowed. Includes for stack and queue appear within graph.h

****** Failure to follow these directions will result in zero (0) credit on this assignment.***

<Project06 Submission Instructions>

Once you are satisfied with your solution, submit the ONE file named below via this Canvas assignment. ***Submisssions by email receive ZERO CREDIT (0 points)!!!***

graph.cpp

NOTE: Your project submission will be automatically graded so it is EXTREMELY IMPORTANT that you READ and FOLLOW the project directions.

Failure to follow directions may result in ZERO CREDIT (0 points) on this assignment.

<Project06 Compilation Instructions>

This project consists of multiple files (**.h** and **.cpp** files) provided by the instructor. You have been provided with a ***makefile*** that contains all commands required to compile and link your project on ***blackhawk***. So, for this assignment, all you must do to compile the program is to use the following command

make

which will follow the ***makefile*** instructions to create an executable named ***Project06*** from the various project ***.h*** and ***.cpp*** files.

If your program compiled successfully, just type **./project06 InputFile** to execute your program.

<Project06 Hints>

You will be better prepared for the Final Exam if you can write the **Graph ADT** code on your own rather than just copying code from the textbook or lecture notes since the written exams are CLOSED BOOK and CLOSED NOTES.

When adapting the code provided during lecture, you will need to make modifications to convert the code work with the Adjacency List representation of a graph. The **WhereIs** method takes the place of the **IndexIs** method since there is **NO ARRAY IN THIS PROJECT**!!!

For help with the **stack** and **queue** containers from the Standard Template Library, see the lecture notes.

<Project06 Adjacency List Representation>

VertexNode and **EdgeNode** are structs used to represent vertices and edges of the graph in the adjacency list representation. In this version of the Adjacency List, there is **NO ARRAY**!!!

Within the constructor, you will need to initialize the pointer to the list of vertices to NULL.

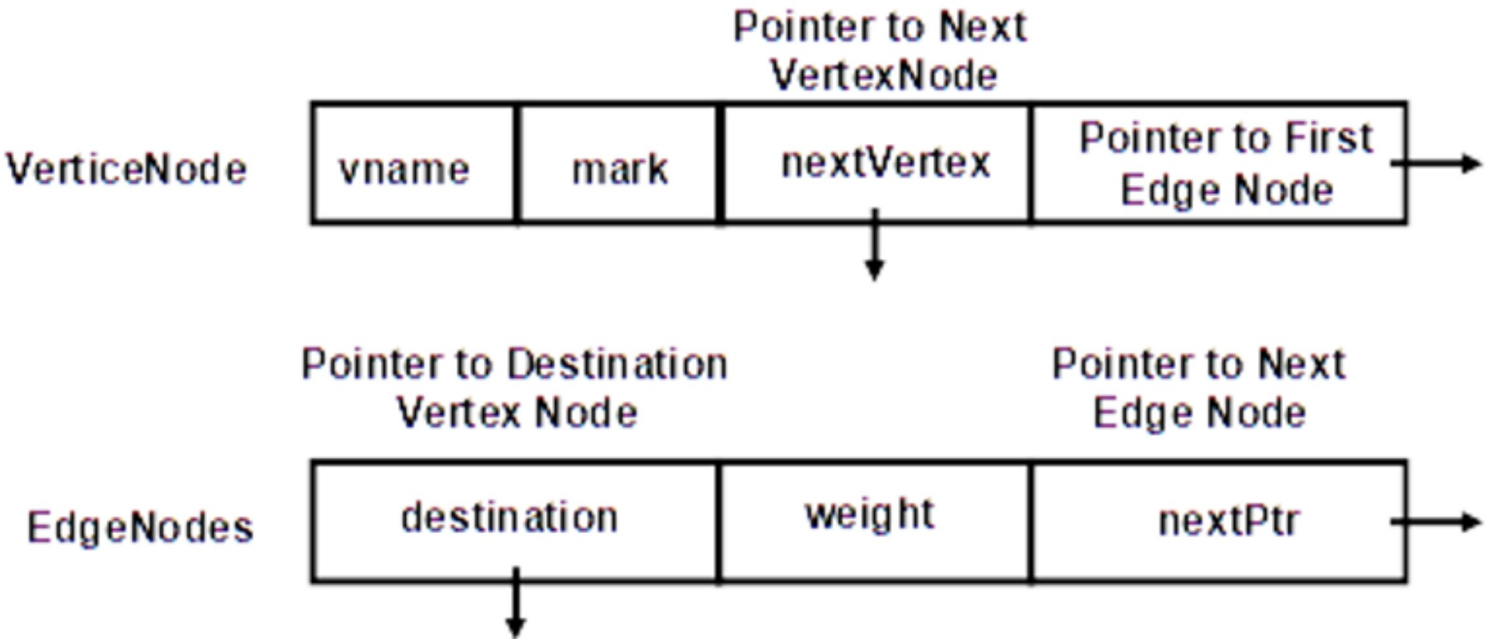

Figure 1: VerticeNode and EdgeNodes Structure Diagram

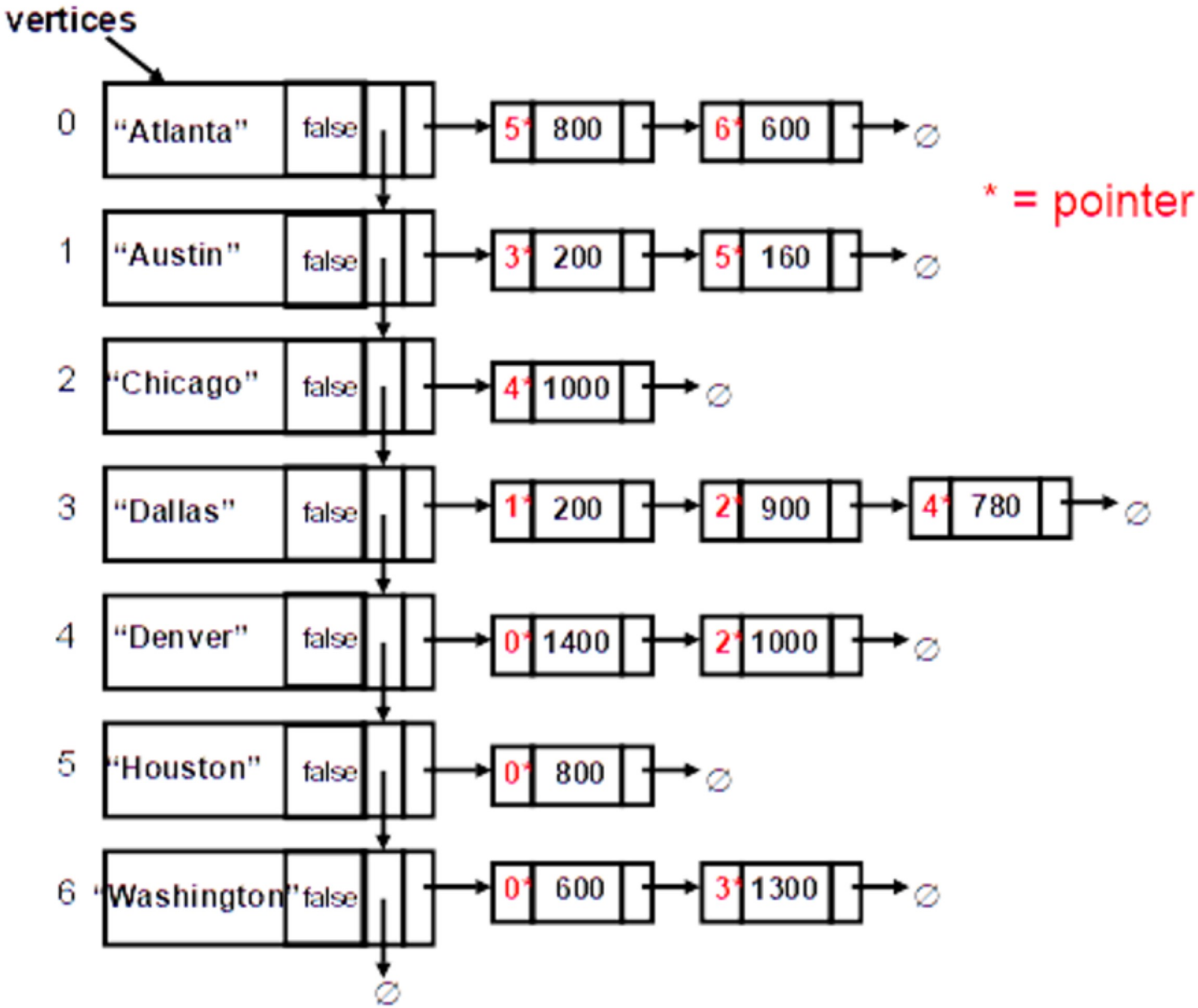

Figure 2: Vertices Linked List Diagram

Points	100
Submitting	a file upload
File Types	cpp

Due	For	Available from	Until
Dec 10	Everyone	Nov 30 at 12am	Dec 12 at 11:59pm

+ Rubric