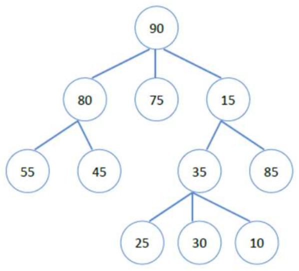
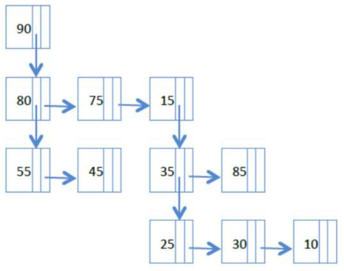
CS 2413 – Data Structures – Spring 2020 – Project Four
Due: 11:59 PM, April 8, 2020

Description: This project is an extension of Project 3. In the class ArrayGLL you will add the following methods:

a) insertAChild (DT& parent, DT& child): You will do a find on parent and add the child to the generalized list pointed by the _Down. To do this, you need to find the location of the parent in the array structure. Assume that you need to add a child 100 to the node 80. The find method for ArrayGLL will return 4. Next, you will execute the findFree method (described below). In the example below, the findFree will return 8 and set the firstFree = 0. The new child 100 is placed in position 8. Set a tempDown variable to be the _Down in array location 4. The new _Down for array location 4 will be whatever the findFree returns which in this case is 8. The _Next for location 8 will be set to tempDown and _Down for location will be set to -1.

b) removeANode (DT& node): First execute find (node) and get the location L where node is found. If it is a leaf node, that is, _Down at location L is -1. Copy the contents of _Next for location L, if _Next at that location is -1, then simply perform delete _Info. If the node is not a leaf node, then replace the node with the left most child in the subtree. That is keep keeping going down the tree until _Down is -1. That is your left most leaf node in the tree. For example, in the example below for node 15, node 25 is the left most leaf node. You need to replace 15 with 25 and remove 25.





	_Info	_Next	_Down
0	999	12	-1
1	75	9	-1
2	90	-1	4
3	30	13	-1
4	80	1	6
5	85	-1	-1
6	55	7	-1
7	45	-1	-1
8	999	0	-1
9	15	-1	10
10	35	5	11
11	25	3	-1
12	999	-1	-1
13	10	-1	-1

firstElement = 2; firstFree = 8;

Where firstElement is the first location in the array that contains the first element. In this case it is the root of the tree that has a value 90.

The free locations in the array are linked by the index positions in the array as a linked list. Follow the linked list starting at index position starting at location 8 and using the _Next index position. For **ease** I have put the value 999 in the _Info position where the index position is empty.

You will input will be something like this (I have comments on each line with a //, this will not be in the input file)
// Need to assume that the first free element is zero and build the GLL
14 //Indicating the size of the array
I -1 90 //insert 90. The -1 indicating that it is the root of the tree.
I 90 15
I 15 85
I 15 35
I 90 75
I 90 80
I 80 45
I 35 10
I 80 55
I 35 30
I 35 25
I 90 200
I 200 300
D //call the display method on the data structure
F 80 //call the find method
F 200 //call the find method
P 200 //display information on the parent of 200; You will call ParentPos.
R 15 //execute the remove method and remove node that contains 15
R 80
R 200
R 90
D
I 45 800
D

Your main program will have the following structure (changes may be required).

```
#include <iostream>
using namespace std;
//define all your classes here (as given above)
//write the methods after the class definition (not inside the class
//definition); You will have this main program to start. I HAVE NOT SYNTAX
CHECKED THIS CODE AND IT IS YOUR REPOSIBILITY.

int main () {
      ArrayGLL<int>* firstGLL;
      int noElements;
      char command;

      cin >> noElements;

      firstGLL = new ArrayGLL<int>(noElements);
      while (!cin.eof()) {
            cin >> command;
            switch (command) {
                  case 'I' : {\\perform the insert; break;}
                  . . . //other cases
            }
      }
      delete firstGLL;

      return 0;
}
```

Redirected Input: Redirected input provides you a way to send a file to the standard input of a program without typing it using the keyboard. To use redirected input in Visual Studio environment, follow these steps: After you have opened or created a new project, on the menu go to project, project properties, expand configuration properties until you see Debugging, on the right you will see a set of options, and in the command arguments type "< **input filename**". The < sign is for redirected input and the **input filename** is the name of the input file (including the path if not in the working directory). A simple program that reads a matrix can be found below.

```
#include <iostream>

using namespace std;

int main () {

      int r,c,cv,nsv;
      int val;

      cin >> r >> c >> cv >> nsv;
      for (int i=0; i < r; i++) {
         for (int j=0; j < c; j++) {
            cin >> value;
            cout << value << " ";
         }
         endl;
      }
      return 0;
}
```

Constraints

1. In this project, the only header you will use is #include <iostream> and using namespace std.
2. None of the projects is a group project. Consulting with other members of this class our seeking coding solutions from other sources including the web on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.

<u>Rules for Gradescope (Project 4):</u>

1. Students have to access GradeScope through Canvas using the GradeScope tab on the left, or by clicking on the Project 2 assignment submission button.
2. Students should upload their program as a single cpp file and cannot have any header files. If, there are header files(for classes) you need to combine them to a single cpp file and upload to GradeScope.
3. Students have to name their single cpp file as 'project4.cpp'. All lower case. The autograder will grade this only if your program is saved as this name.
4. Sample input files and output files are given. Your output should EXACTLY match the sample output file given. Please check the spaces and new lines before you email us telling that the 'output exactly matches but not passing the test cases. Suggest using some type of text comparison to check your output with the expected.
5. Students need to have only one header file(iostream) while uploading to GradeScope. You cannot have 'pch.h' or 'stdafx.h'.
6. Students cannot have 'system pause' at the very end of the cpp file.
7. A few test cases will be released during the course of the project and the remaining test cases will be released after all the students have submitted that will add up to the allocated autograder points on GradeScope.
8. Manual grading involves implementation points and documentation points that will be graded after every student submits.