PA1 – PCS Tree

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: (•) Yes ( ) No

Name: Rachel Micolichek

Date: 04/08/23

Submission Details

Final ***Changelist*** number: 181061

Verified build: (•) Yes ( ) No

Number Tests Passed: 80

Required Configurations: 2: { Debug/Release | x86 }

Discussion (What did you learn):

I learned about the importance of planning for edge cases. It's easy when the unit tests are provided because it tells you if you missed one. Without the unit tests, I need to make sure I'm thinking through possible edge cases. Additionally when the unit tests are provided, I need to make sure I understand what is causing the edge case.

Verify Builds

- Follow the Piazza procedure on submission
  - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
  - No – Generated files
    - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db, *.user
    - Anything that is generated by the compiler should not be included
  - No – Generated directories
    - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
  - *.sln, *.cpp, *.h
  - *.vcxproj, *.vcxproj.filters, CleanMe.bat

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Points will be deducted if minimum is not reached

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn't mean full credit

Submission Report

- Fill out the submission Report
  - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
  - Warning level ALL …
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
  - Code needs to work "as-is".
    - No modifications to files or deleting files necessary to compile or run.
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason…
  - It will not be graded and you get a 0

No Containers

- NO STL allowed {Vector, Lists, Sets, etc...}
  - No automatic containers or arrays
  - You need to do this the old fashion way - ***YOU EARNED IT***

Leave Project Settings

- Do NOT change the project or warning level
  - Any changing of level or suppression of warnings is an integrity issue

Simple C++

- No modern C++
  - No Lambdas, Autos, templates, etc…
  - No Boost
- NO Streams
  - Used fopen, fread, fwrite…
- No code in MACROS
  - Code needs to be in cpp files to see and debug it easy
- ***Exception:***
  - implicit problem needs templates

Leaking Memory

- If the program leaks memory
  - There is a deduction of 20% of grade
- If a class creates an object using new/malloc
  - It is responsible for its deletion
- Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
  - Leaking is ***HORRIBLE***, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
  - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
  - All files must be active to get credit.
  - Better to lose points for unit tests than to disable and lose all points

No Adding files to this project

- This project will work “as-is” do not add files…
- Grading system will overwrite project settings and will ignore any student’s added files and will returned program to the original state

UnitTestConfiguration file (if provided) needs to be set by user

- Grading will be on the UnitTestConfiguration settings
  - Please explicitly set which tests you want graded… no regrading if set incorrectly

Due Dates

- See Piazza for due date and time
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** and commit to perforce
    - ***ONLY*** use Adobe Reader to fill out form, all others will be rejected.
    - Fill out the form and discussion for full credit.

Goals

- Create the Parent-Child-Sibling tree
    - Learn a useful data structure that is used in real-time data structures.
- Understand trees and recursion

Assignments

1. ***Create the PSCTree/PCSNode program in C++***
    a. Document the code
    b. Code should be Warning Level Wall free
    c. For all configurations { DEBUG/RELEASE, x86 }

2. ***Program should be able to create and modify a PSC tree in the following ways***
    a. Tree functions
        1. Create / Destroy PSC nodes
        2. Create / Destroy PSC tree
        3. Insert / remove nodes to the tree
        4. Get the Root node of the tree
        5. Get the info on the tree stats
    b. Print functionality (also called dump)
        1. PrintTree complete hierarchy
        2. PrintChildren for a particular node
        3. PrintSibling for a particular node

c. Individual Node functions within Tree
    1. Get / Set the name of a PSC node
    2. Print the name of a PSC node
    3. Get parent node
    4. Get child node
    5. Get number of siblings
    6. Get next sibling

3. Development – there are Unit tests for grading.

Here is a list of demos, that might get you up and going.
Goal here is twofold

- Learn how to Architect / Design Software
- Learn the API and Uses of this new data structure

These demos will not count, only the unit test will, but can get you started now.

PCS Demo 1: Create the sample tree

- Implement the sample tree given with your code using your interface
- Insert one node at a time to match the supplied tree

PCS Demo 2: Dump the tree

- Print (dumps) all the nodes from sample tree

PCS Demo 3: Delete the whole tree

- All the nodes and links
- Print the whole tree, should only contain the root node

PCS Demo 4: Delete any node and its respective children

- For every node in the tree,
    1. Deletes the node P and their respective sub-nodes
    2. Dumps the compete tree after the deletion of node P
- Delete the whole tree
- Recreate the sample tree
    1. Effectively resetting the given tree sample
- Repeat until you've tested every node

Validation

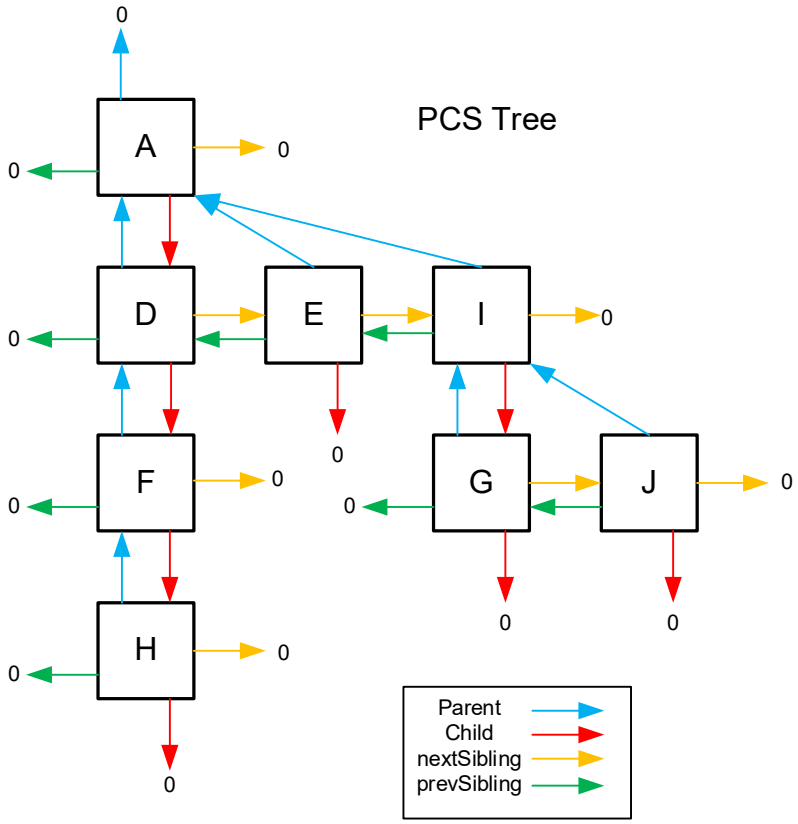
Simple checklist to make sure that everything is submitted correctly

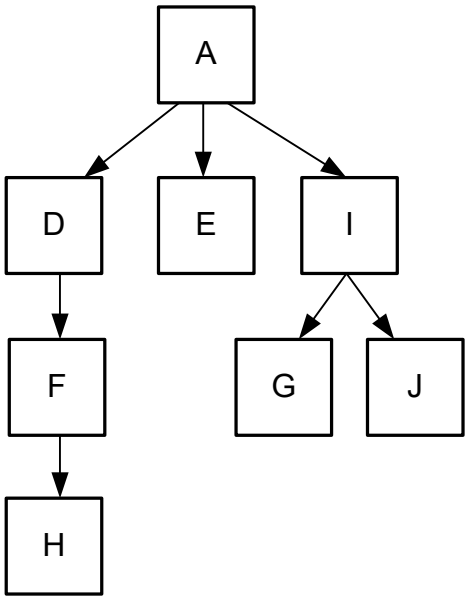
- Is the project compiling and running without any errors or warnings?
- Does the project run ***ALL*** the unit tests execute without crashing?
- Is the submission report filled in and submitted to perforce?
- Follow the verification process for perforce
    - Is all the code there and compiles "as-is"?
    - No extra files
- Is the project leaking memory?
- Did you build for x86 for both DEBUG and RELEASE?

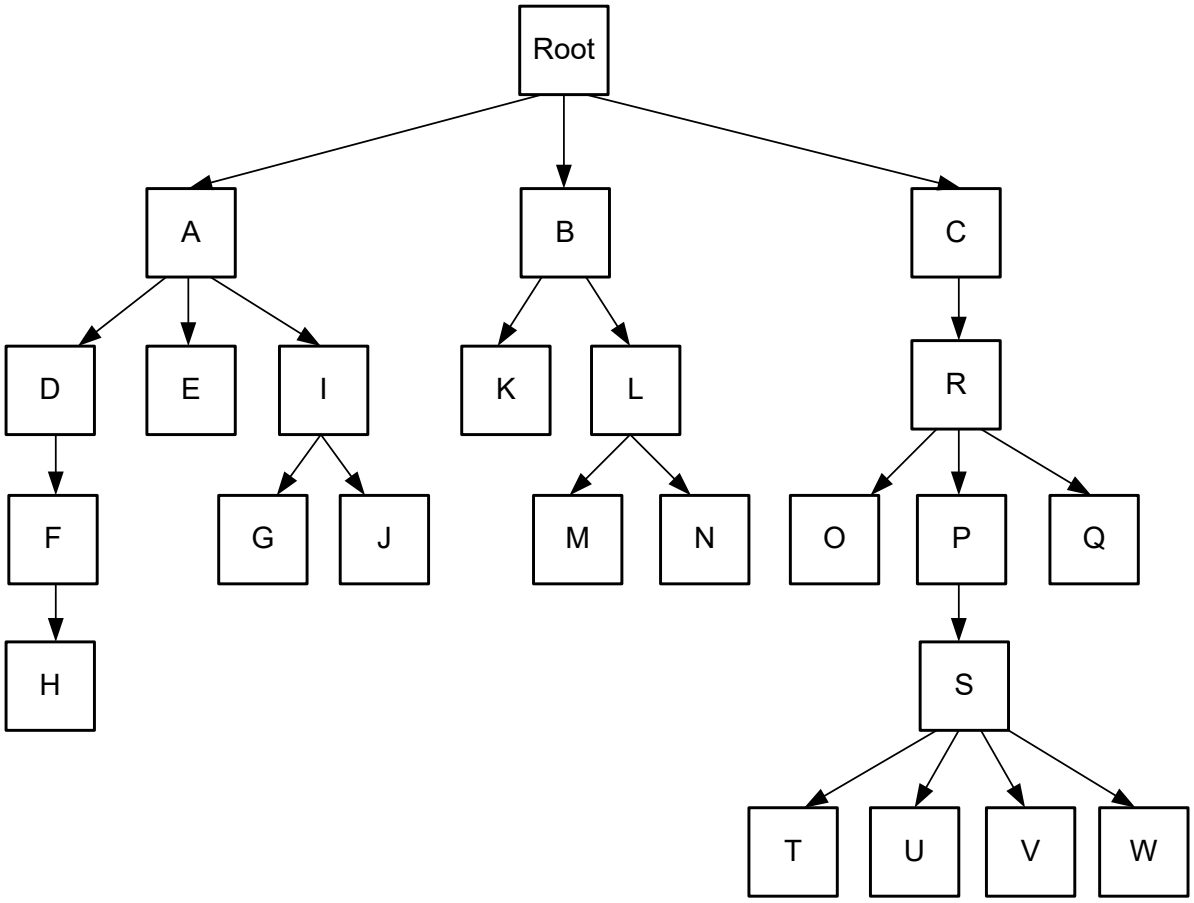
Hints

Most assignments will have hints in a section like this.

- Do this assignment by iterating and slowly growing your project
- Debugging is the key to this program
    - Learn how to debug, add functions to make it easy to debug and track
- ***IMPORTANT:*** Start out having each node containing a string
    - Add a method to print the string name, pointers to the surrounding nodes
        - Don't print the pointer address literally
            - Print the string name of the node the address points to
            - Then it's symbolic and doesn't change at every compile due to different memory addresses
    - The logging and printing functionality is directly affects how easy this assignment will be to design and implement.
- You will need recursion - either depth first or breadth first for Printing and other methods.
    - Practice with many simple programs before you do the project for real.
    - Pointers - References
- I hope that you understand them - this class and this assignment is 100% unforgiving if your pointers addresses are off. Do them right.
- Print, print, print
    - Draw diagrams to help you understand


PCS Tree
A
D
E
I
F
G
J
H
0
Parent
Child
nextSibling
prevSibling


A
D
E
I
F
G
J
H


Root
A
B
C
D
E
I
K
L
R
F
G
J
M
N
O
P
Q
H
S
T
U
V
W