

PA5 – PCSTree Iterators

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:

Date:

Submission Details

Final **Changelist** number:

Verified build: Yes No

Number Tests Passed:

Required Configurations:

Discussion (What did you learn):

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

Allowed to Add files to this project

- This project will work "as-is" do not add files...

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

- Grading will be on the UnitTestFixture 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 performe 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 a Depth First iterator for a Tree
- Create a Bottom Up iterator for a Tree

Assignments

1. No unit test will be provided to the students
 - a. You need to develop your own testing process
 - b. Instructor provided tests will be evaluated remotely for this project
2. Create a Depth First iterator for a Tree
 - a. Called the ***forward iterator***
 - b. Adjust the forward pointer for each PCSNode
 - c. Make sure you verify inserts and deletes of nodes to the tree
3. Create a Bottom Up iterator for a Tree
 - a. Called the ***reverse iterator***
 - b. Adjust the reverse pointer for each PCSNode
 - c. Make sure you verify inserts and deletes of nodes to the tree
4. You can Create your own unit tests (Hint)
 - a. They do not need to be submitted
 - Just add your code
 - You can use the tests from PA1 as examples
 - b. Do not delete the Keenan_A,B,C,D unit tests
 - Those are stubbed out for grading
5. **Feel free to add Unit Tests (ONLY)**
 - a. **Do not change or add source file or header files to project**
 - **Those files are tied to the project settings**
 - b. **You are ONLY allowed to add unit tests**

Example:

Image you have the original tree loaded nodes Root-W nodes (standard sample tree)

- This should get you started....

Forward iterator sample:

```
PCSTreeForwardIterator pForIter(tree.GetRoot());  
PCNode *pNode = pForIter.First();  
  
CHECK(pNode == &nodeRoot);  
pNode = pForIter.Next();  
  
CHECK(pNode == &nodeA);  
pNode = pForIter.Next();  
  
CHECK(pNode == &nodeD);
```

Reverse iterator sample:

```
PCSTreeReverseIterator pRevIter(tree.GetRoot());  
PCNode *pNode = pRevIter.First();  
  
CHECK(pNode == &nodeQ);  
pNode = pRevIter.Next();  
  
CHECK(pNode == &nodeW);  
pNode = pRevIter.Next();  
  
CHECK(pNode == &nodeV);
```

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 performce?
- Follow the verification process for performce
 - Is all the code there and compiles "as-is"?
 - No extra files
- Is the project leaking memory?

Hints

Most assignments will have hints in a section like this.

- Do this assignment by iterating and slowly growing your project
 - Write your use cases first
 - Diagram, diagram
 - Attack forward iterator first
- Print using Trace is very useful
 - Don't print pointers, using the node names
 - 100x easier to debug

