Research – Parallelization

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: Joey Domino

Date: 11/28/2022

Submission Details

Final ***Changelist*** number: 163498

Verified build: (•) Yes ( ) No

Required Configurations: 1: { Debug | x86 }

Discussion (What did you learn):

IM GONNA TRY DAMMIT!

HOLY CRAP I GOT IT DONE!

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
    - 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, *.suo,
  - *.vcxproj, *.vcxproj.filters, *.vcxproj.user
  - *.cpp, *.h
  - 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
    - Seriously, I'm checking

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

Leave Project Settings

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

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
- Disable your debug printing otherwise you will lose points

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

- Learn
  - To parallelize code…
    - First the code needs to be linear to allow multiple threads to divide work.
  - Research Demo
    - You linearize a non-linear problem for future parallelization

Assignments

1. This is a Research/Demo …
    a. Produce the design/code to do the following
        i. Implement the copy assignment operator for a special linked list
    b. Code up the linked lists… Use the data structure provided
        i. For development… Try approximately 6 nodes to prove the concept
            1. After having it work with 6… return the count to 500 for final testing
    c. **Create a function** to do the deep copy assignment operator
        i. Given the head pointer
        ii. Create a deep copy of the list with the same data, with the same relationships
            1. But every node is a copy.
            2. Example
                - If the original A random points to C
                - The new A random points to the new C
    d. Use Trace::out() or the provided Print() methods to help in debugging
        i. Walk the list printing all the nodes of the new list
    e. Pdf write up with diagrams
        i. 2-3 pages with O() notation
        ii. Explain the complexity of the function what is the Big O notation
        iii. The diagrams to communicate what you are doing in your function

2. The problem:

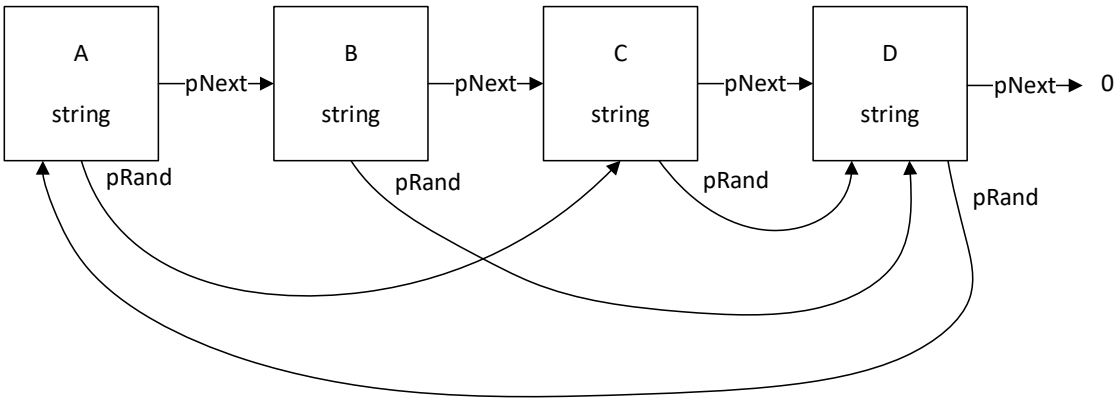
    Thinking in C/C++, assume you have structure:

```
struct Node
{
      std::string element;
      Node *pNext;
      Node *pRand;
};
```

    Implement the copy assignment operator will deep copy the linked list. The function will receive an existing list and it should return the deep copied list.

    The element is a random string for each node.

    The random pointer points to a random element within the node. For example, the first node's random might point to the fifth node, the second node's random might point to the first node, and the third node's random might also point to the fifth node.



When copied, the random pointer must carry to the copy. For example, if the first node's random points to the fifth node, then the copy's first node's random must point to the copy's fifth node.

Solve for performance and optimization. Use as little extra memory as possible. State the big-O for your algorithm when turning it in.

1. ***You need to a copy assignment operator***
    a. Add all your code there
    b. Place your PDF in the same directory

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*** in all configurations 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?

Hints

Most assignments will have hints in a section like this.

- I had to do a lot of marker board… it was easy to figure out.
    - Once you get the diagrams working… the code was trivial.