Collatz

Due: Thu, 12 Sep 2013, 10pm
70 pts, 7% of total grade.

Specification

Write a program, **individually**, to solve the **The 3n+1 Problem** in **C++**:

- Sphere
- 4073. The 3n plus 1 problem [pdf]
- 4765. The 3n plus 1 problem V2 [pdf] (**5 bonus pts**)
- xkcd

For **all** projects, the **minimum** requirement for getting a **non-zero** grade is to write **standard-compliant C++ (-std=c++0x)** and to satisfy **all** of the **requirements** in the **table** below, including the precise **naming** of all the **files**.

For **this** project, the additional **minimum** requirement for getting a **non-zero** grade is that **Sphere** accept your solution to **4073 (C++ 4.3.2)** and that you design and implement some form of **cache** to make it efficient.

You can earn **5 bonus pts**, if **Sphere** accepts your solution to **4765 (C++ 4.3.2)**.

Bonus pts will **not** increase the **total score** beyond the **max** score.

You **may not** use **new**, **delete**, **malloc()** or **free()**. You **may** use the **STL**.

Analysis

These are additional descriptions of the underlying math:

- MathWorld: Collatz Problem
- Wikipedia: Collatz Conjecture

Tools

- Doxygen
- Git
- GitHub
- Google Test (1.6.0)
- Valgrind

Guides

- Git Cheat Sheet
- Git Guide
- Git Immersion

- Git Reference
- Google C++ Style Guide
- Try GitHub

Requirements

	Points	Description	Files	Submission
1	5 pts	**Git Repository** Set up a **private Git repository** at **GitHub**, named **cs371p-collatz**. Invite the grader to your repository. Commit **at least 5 times**. Commit once for each **bug** or **feature**. If you cannot describe your changes in a sentence, you are not committing often enough. Write meaningful commit messages and identify the corresponding **issue** in the **issue tracker** (below). Create a **tag** for important milestones (e.g. without a cache, with a lazy cache, etc.). Create a **log** of the commits. Push frequently. It is **your** responsibility to protect your code from the rest of the students in the class. If your code gets out, **you** are as guilty as the recipient of **academic dishonesty**.	Collatz.log	GitHub Turnin
2	5 pts	**Issue Tracker** The **GitHub** repository comes with an **issue tracker**. Create an issue for each of the **requirements** in this table. Create an issue for each **bug** or **feature**, both open and closed. Describe and label each issue adequately. Create **at least 10** more issues in addition to the **requirements** in this table.		GitHub
3	15 pts	**Unit Tests** The grader's **GitHub** account will have a public **Git repository** for **unit tests** and **acceptance tests**. It is **critical** that you clone the grader's public repo into a **different** directory than the one you're using for your private repo. Write unit tests **before** you write the code. When you encounter a bug, write a unit test that **fails**, fix the bug, and confirm that the unit test passes. Write **at least an average of 3** unit tests for **each** function. Tests corner cases and failure cases. Name tests logically. Push and pull the unit tests to and from the grader's repository. Prepend **<cs-username>**- to the file names at **GitHub** (i.e. **foo-TestCollatz.c++** and **foo-TestCollatz.out**). Reach consensus on the unit tests. You **must** use **Valgrind**.	TestCollatz.java TestCollatz.c++ TestCollatz.c++.out	GitHub Turnin
4	15 pts	**Acceptance Tests** The grader's **GitHub** account will have a public **Git repository** for **unit tests** and **acceptance tests**. It is **critical** that you clone the grader's public repo into a **different** directory than the one you're using for your private repo. Write acceptance tests **before** your write the code. When you encounter a bug, write an acceptance test that **fails**, fix the bug, and confirm that the acceptance test passes. Write an auxiliary program to **randomly generate** acceptance tests. Create **at least 1000 lines** of acceptance tests. Tests corner cases and failure cases. Push and pull the acceptance tests to and from the grader's repository. Prepend **<cs-username>**- to the file names at **GitHub** (i.e. **foo-RunCollatz.in** and **foo-RunCollatz.out**). Reach consensus on the acceptance tests. You **must** use **Valgrind**.	RunCollatz.java RunCollatz.c++ RunCollatz.in RunCollatz.c++.out	GitHub Turnin
5	15 pts	**Implementation** Use **assert** to check **pre-conditions**, **post-conditions**, **argument validity**, **return-value validity**, and **invariants**. Worry about this **last**, but your program should run as **fast** as possible and use as **little** memory as possible.	Collatz.java Collatz.h Collatz.c++	GitHub Turnin
6	5 pts	**Documentation** Use **Doxygen** to document the **interfaces**. The above documentation only needs to be generated for **Collatz.h**. Comment each function meaningfully. Use comments **only** if you need to explain the **why** of a particular implementation. Choose a coding convention and be consistent. Use good variable names. Write readable code with good indentation, blank lines, and blank spaces.	html/*	Turnin
7	5 pts	**Sphere** Sphere requires a single file to be submitted. Combine **Collatz.h**, **Collatz.c++**, and **RunCollatz.c++**. This is the file that the grader will submit to Sphere to determine a **zero** vs. **non-zero** grade.	SphereCollatz.c++	GitHub Turnin
8	5 pts	**Submission** Rename "**makefile.c++**" to "**makefile**". Fill out the **Google Form** and submit the **ZIP file** to **Turnin**.	makefile.c++ Collatz.zip	Google Turnin

Grader

Name	GitHub ID	GitHub Test Repository	Turnin ID	Turnin Project Folder	Google Form
Chuying Huang	hchuying	cs371p-collatz-tests	hchuying	cs371ppj1	Google Form

Submission

Submit a single **ZIP** file, named **Collatz.zip**, to the grader's **Turnin** account, with the following files:

- makefile
- html/*
- Collatz.c++
- Collatz.h
- Collatz.log
- RunCollatz.c++
- RunCollatz.in
- RunCollatz.out
- SphereCollatz.c++
- TestCollatz.c++
- TestCollatz.out

3130 reads