Voting

Due: Thu, 26 Sep 2013, 10pm
70 pts, 7% of total grade.

Specification

Write a program to solve the **Australian Voting** problem in **C++**:

- UVa
- UVa Toolkit
- 10142. Australian Voting [pdf]

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 **UVa** accept your solution to **10142 (C++ 4.3.2)** and that when reassigning the ballots of a **loser**, you do **not** iterate over **all** the ballots.

For this project, yet another additional **minimum** requirement for getting a **non-zero** grade is that your code **must** successfully pass **five** other students' acceptance tests.

You can earn **5 bonus pts**, if you work with a **partner** using **pair programming** and vouch for the fact that you worked on the project **together** for more than **75%** of the time.

Only **one** solution **must** be turned in for the **pair**. If **two** solutions are turned in, there will be a **10%** penalty, and the **later** one will be graded.

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:

- Wikipedia: Arrow's Theorem
- Wikipedia: Range Voting

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-voting**. 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**.	Voting.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-TestVoting.c++** and **foo-TestVoting.out**). Reach consensus on the unit tests. You **must** use **Valgrind**.	TestVoting.c++ TestVoting.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. In your acceptance tests include **five** other students' acceptance tests. 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-RunVoting.in** and **foo-RunVoting.out**). Reach consensus on the acceptance tests. You **must** use **Valgrind**.	RunVoting.c++ RunVoting.in RunVoting.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.	Voting.h Voting.c++	GitHub Turnin
6	5 pts	**Documentation** Use **Doxygen** to document the **interfaces**. The above documentation only needs to be generated for **Voting.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	**UVa** UVa requires a single file to be submitted. Combine **Voting.c++** and **RunVoting.c++**. This is the file that the grader will submit to UVa to determine a **zero** vs. **non-zero** grade.	UVaVoting.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++ Voting.zip	Google Turnin

Grader

Name	GitHub ID	GitHub Test Repository	Turnin ID	Turnin Project Folder	Google Form
Aizhuldyz Gabbassova	aizhuldyz	cs371p-voting-tests	aizhuli	cs371ppj2	Google Form

Submission

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

- makefile
- html/*
- Voting.c++
- Voting.h
- Voting.log
- RunVoting.c++
- RunVoting.in
- RunVoting.out
- UVaVoting.c++
- TestVoting.c++
- TestVoting.out