Lecture 06b
Midterm Review
Sept. 28th, 2017

Announcements
- PA4 is due one week from Tuesday
- Quiz 5 is due before class this Tuesday

Midterm #1 : Raw Scores

<table>
<thead>
<tr>
<th>Raw Score</th>
<th>Curved Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>57</td>
</tr>
<tr>
<td>40</td>
<td>64.5</td>
</tr>
<tr>
<td>50</td>
<td>72.8</td>
</tr>
<tr>
<td>55</td>
<td>77</td>
</tr>
</tbody>
</table>

The Curve

\[ C = 77 + \frac{10 \times (R - 55)}{12} \]

<table>
<thead>
<tr>
<th>Raw Score</th>
<th>Curved Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>57</td>
</tr>
<tr>
<td>40</td>
<td>64.5</td>
</tr>
<tr>
<td>50</td>
<td>72.8</td>
</tr>
<tr>
<td>55</td>
<td>77</td>
</tr>
</tbody>
</table>

Make Questions (1-4)

CFLAGS = -Wall -l. -g -c

Will the systems compile if:
1. -Wall is removed? Yes (89.7%)
2. -l. No (78.6%)
3. -g. Yes (73.5%)
4. -c. No (69.2%)

Case Alpha: Output (5-7)

```c
case 'a': {
    Reader r("foo bar foo bar toaster");
    cout << r.Comparer("foo bar toaster") << endl;
    cout << r.Count() << endl;
    cout << r.Size() << endl;
    return 0;
}
```

5. Output #1: 2 (82.9%)
6. Output #2: 2 (73.5%)
7. Output #2: 99 (85.4%)
Where in memory [stack/heap/other] is… (8-10)

- 8. Where is r: stack (it's a local variable) 88.0%
- 9. Where r.ht: stack (it's part of – inside – r) 64.1%
- 10. Where is r.ht.table: stack (same explanation) 21.4%

Note: table is a pointer.

Constructor/Destructor Questions

11. How many string constructors are called within the HashTable constructor?

Table = new string[size] // size = 99
Therefore 99 string constructors are called. (28.2%)

12. How much memory (if any) is leaked?
None: delete [] table (in ~HashTable()) (76.1%)

How Many Bytes Does argv Point To?

13. Command line: HashTable alpha

<table>
<thead>
<tr>
<th>Command</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>HashTable</td>
<td>9+1 = 10 Bytes</td>
</tr>
<tr>
<td>alpha</td>
<td>5+1 = 6 Bytes</td>
</tr>
<tr>
<td>Total:</td>
<td>16 + 10 + 6 = 32 Bytes (5.1%)</td>
</tr>
</tbody>
</table>

Buggy Assignment Operator

HashTable& HashTable::operator = (const HashTable&) src
{
    table = new string[size];
    for(unsigned int index = 0; index < size; index++) {
        table[index] = src.table[index];
    }
    size = src.size;
    factor = src.factor;
    return *this;
}

1) Fails to delete old data (pointed to by table)
2) Allocates new table using old value of size

Case Beta (14-16)

14. What is the first line of output? 17 (set in = operator) (76%)
15. How many HashTable constructors are called? 2 (45.3%)
16. How many HashTable copy constructors? 0 (53.8%)

Case Beta (17-19)

17. How many HashTable destructors are called 2 (75.2%)
18. How many string constructors are called? 17 + 1 + 1 = 19 (8.5%)
19. How many string destructors? 17 + 1 = 18 (11.1%)
Case Beta (20)

- How much memory does the program leak?

  One string. Why?
  Constructors and destructors match
  But = operator fails to delete prior to copy
  One string (allocated for ht2) is lost (33.3%)
General Questions (36-40)

36. Does HashTable define the methods of a header class? Yes: the “Big 3” (1) Copy Constructor, (2) Destructor & (3) Assignment operator.

37. Which method should have been an accessor? Contains

38. If #define HASH_TABLE_H is removed, will system compile? No – double definitions (main loads HashTable & Reader, which load HashTable)

39. If #define READER_H is removed, will system compile? Yes – Reader is never double loaded

40. Which of the four cases crash? Gamma (infinite recursion -> stack overflow), Delta (delete stack variable -> seg fault)