

CSCA48 Tutorial 7 - Testing

Tabeeb Yeamin, github.com/tabeebyeamin

March 5, 2020

Agenda

- Administrivia
- Testing
- Testing Practice

- Your midterms are being marked right now, at this very moment
- Expect them done by early next week

- A **test case** is a set of actions executed to verify a particular feature or functionality of your software application
- it contains test steps, test data, precondition(s), postcondition(s) developed for specific test scenario to verify any requirement

Why Test?

- helpful to measure whether expectations of the client/instructor etc. were fulfilled or not
- Helps you design correct code
- Helps massively if you are collaborating with others on a code base
- **Test Driven Development:** Writing tests first, code after. (You'll learn more in CSCB07, CSCC01 etc.)

Design a test-suite for insertMovieReview()

```
/**  
Inserts a new movie review into the linked list,  
if it does not exist already.  
*/  
ReviewNode *insertMovieReview(char *title,  
char *studio, int year, double B0_total,  
int score, ReviewNode *head)
```

- Describe the test in detail, e.g.:
 - Expected input to the function
 - What the expected result should be if correct
 - How to check that the result is as expected (no code is needed, but detail should be enough to implement)

Test-suite for insertMovieReview

- Insertion into an empty list
 - (result is, new node should be at head, content should be correct, 'next' pointer for head should be NULL)
- Insertion into a non-empty list
 - (result is, new node is at head, 'next' pointer points to old head, information is correct in both nodes)
- Trying to insert a duplicate movie
 - (should NOT insert anything)
- Inserting more than 2 movies, checking they are linked in the right order as expected, content is correct

Design a test-suite for deleteMovieReview()

Perform the same exercise for deleteMovieReview

```
/** Removes a review matching the input query  
from the linked list, if such review can be found.  
*/
```

```
ReviewNode *deleteMovieReview(char *title,  
char *studio, int year, ReviewNode *head)
```


Test-suite for deleteMovieReview

- Deletion on an empty list (should not crash)
- Deletion on a list of length 1 (should delete the node and return NULL so head is updated to be an empty list)
- On a list with multiple entries:
 - Deletion at head
 - Deletion at tail
 - Deletion in the middle (for all of these, the linkage needs to be checked and verified correct)

Design a test suite for sortReviewsByTitle()

Perform the same exercise for sortReviewsByTitle

```
/** Sorts the list of movie reviews in  
ascending order of movie title.  
If duplicate movie titles exist,  
the order is arbitrary.  
*/
```

```
ReviewNode *sortReviewsByTitle(ReviewNode *head)
```

Test-suite for sortReviewsByTitle

- Resulting list is actually sorted (how?)
- Resulting list has the exact same reviews as input (how?)
- No dangling pointers or endless loops
- Try multiple different input ordering and verify it works

- None of the tests above ask to check the *output* of the program!
- It is always checking the information stored in the list, and the pointers linking things together!
- We test for correctness, not output.
- If it's correct, it will produce the right output. Not the other way around.