CMPS 282 Software Engineering — Spring 2010
Section 1
MIDTERM EXAM
50 minutes

May 12, 2010

Please draw a horizontal line across the page between the answers to each question

You may refer to the following during the exam:

- the course textbook
- the course lecture notes
- your homework solutions
- any notes that you have taken in class

You may **not** refer to any other materials. Good luck!

1. (40 points) This question is about the file system example specification. What happens if the operation `moveObj`($String\ p, String\ n$) is invoked with $o \in$ `ancestors`(d), where o is the object in `Cur` with name n?

Discuss in detail and justify your answer. You will receive no credit for guessing.

Answer.
The resulting file system will violate the acyclicity and reachability constraints, since a cycle is created containing o, d, and every directory that is an ancestor of d and also a descendant of o, i.e., all directories on a path from d to o in the "parent" tree.

2. (60 points) This question is about the web search example design. We wish to add the following operaion to `Engine`:

operation `queryMoreDisj`($String\ w$)
CHECKS: `Key` $\neq \emptyset \wedge w \notin$ `NK` $\wedge\ w \notin$ `Key`
EFFECTS: Adds w to `Key` and makes `Match` be the documents that are either already in `Match` **or** that contain w (i.e., disjunction).
Orders `Match` properly by the total occurrence count, as before.

You may assume that `queryMore`(w) and `queryMoreDisj`(w) are not both invoked on the same query, i.e., queries are either "conjunctive" or "disjunctive".

(a) (20 points) Which modules must be modified, e.g., by adding new methods or modifying existing methods?

Answer.
`Query`, `Query.addKeyDisj`(w), `Query.addDoc`($d,\ h$)

(b) (20 points) For each existing method that is modified, give a revised specification, and for each new method that is introduced, give a specification.

Answer.
`Query`: store whether a query is conjunctive or disjunctive. This is determined when either `queryMore`(w) or `queryMoreDisj`(w) is first invoked on the query.

`Query.addKeyDisj`(w): same as `addKey`(w), except now just add documents that match w to the documents currently in the Query.

`Query.addDoc`(d, h): if query is conjunctive, then same as before. If query is disjunctive, then matches are docs that contain at least one keyword. Computing occurrence counts and sorting as before.

(c) (20 points) For each existing method that is modified, give a revised implementation sketch, and for each new method that is introduced, give an implementation sketch.

Answer.

`class Query` {
OVERVIEW: as before
`WordTable` k;
`Vector` *matches*; //Vector of `DocCnt` objects
String[] *keys*; //The keywords used in the current query
char kind; //'c' for conjunctive queries, 'd' for disjunctive

`void addKeyDisj`(*String* w) throws `NotPossibleException`
REQUIRES: w is not `null`
MODIFIES: `this`
EFFECTS: If `this` is empty or $w \in$ `Key` throws `NotPossibleException` else
modifies `this` to contain docs in `this` and those that match w,
HELPS: `Engine.queryMore`(w)
IMPLEMENTATION SKETCH:
store current matches in a vector
add w to *keys*
look up w in the `WordTable`
store the information about w's matches in a hash table
for each current match, look up the document in the hash table and
if it is there, add w's ocurrence count to the total occurrence count and remove the document from the hash table
add the remaining documents in the hash table to the vector
sort the vector using `quickSort`

`void addDoc`(`Doc` d, `Hashtable` h)
REQUIRES: d is not `null` and h maps strings (the interesting words in d)
to integers (the occurrence count of the word in d).
MODIFIES: `this`
EFFECTS: If each keywords of `this` is in h, adds d to the matches of `this`.
HELPS: `Engine.addDocs`(u)
IMPLEMENTATION SKETCH:
use the argument h to get the number of occurrences of each keyword ($\in$ `Key`)
If (query is conjunctive and the document d contains all the keywords) or (query is disjunctive and the document d contains some keyword)
compute the total occurrence count *sum* for all keywords and insert the $<d, sum>$ pair in the vector of matches.