General Overview:

This project is done in a few major steps: parsing the xml file, organizing the parsed data into files, sorting and converting those files to indexes, parsing user queries, and executing those queries on the indexes.

Parsing the xml file is fairly simple given the patterns that xml exhibits, all the program does is split the input string into parts of a list whenever a ‘<’ or ‘>’ is encountered and look through the resulting list for keywords such as ‘aid’.
Everything connected to a keyword is then appended to a list correlating to that keyword, and then the lists are written to the appropriate txt file.
With the files filled out the program uses linux sort on them and then uses the provided Perl script and db_load to convert them to indexes.
With that done, the user is prompted for the query they want to perform.
Once the user enter their query the program searches the indexes for the appropriate answers, making itself a list of ad ids.
The user is then allowed to choose their print out format, the answers are printed out, and the user is allowed to enter another query or quit.

User's Guide:

The program is a bit finicky to use, but fairly straight forward once you have the correct input format. Simply run p2.py < file.txt to create the indexes, completing phase 1 and phase 2 automatically, then run final.py to begin phase 3. In phase 3 the user is prompted to enter their query, and assuming it is valid they will then be asked which printout type they want and the answers will be given in the appropriate format. The user will then have the option of entering another query or ending the program.

Description of Algorithm

So, small confession, we had a hard time figuring out how to set a range to explore within the hash and B-Tree. Instead the entire index file is searched for valid answers.

Testing Strategy

We started by hard-coding in our queries and where the program would look for answers, and we did this with multiple queries for each type of query. Once we were confident that the program was behaving as expected in that way we un-hard-coded our queries and made sure that our parsing was reacting to the different query types properly, again using multiple queries of each type for testing. We only entered valid queries, and we corrected errors as we encountered them until we could enter a range of queries for each query type and the program would give the correct result.

Group Work Break-Down

Liam McDonald
Hours spent on project: 14
Contributions:

- Coded all of phase 1
- Assisted in coding of phase 2
- Coded parts of phase 3
- Wrote this report
- Tested the program

Navjit Gill
Hours spent on project: 17

- Assisted in coding of phase 1
- Coded all of phase 2
- Coded parts of phase 3
- Tested the program
- Organized code for final submission

Jujhar Brar
Hours spent on project: 0
Contributions:

- N/A