./Pranav Yalamala

“I pledge my honor that I have abided by the Stevens Honor System.”

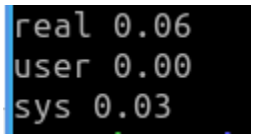
Text2bin.c:

Approach:

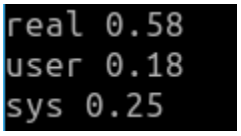
For this program I used the fgets() method to go through the input file line by line. In each line I use sscanf to look for a specific format of inputs. This is then used to load the value of each parameter (user_id, item_id, rating, and timestamp) into the proper variable. Each vaiable is then written to the output binary file using fwrite(). Each variable is specified to the proper number of bytes using fwrite, so user_id gets 4, item_id gets 2, ratng gets 1, and timestamp gets 8. After this the files are closed.

Timings:

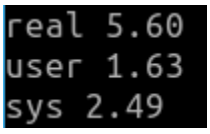
100,000 ratings:



1 million ratings:



10 million ratings:



Every time the data set is increased by a factor of 10, the time the program takes is also roughly increased by a factor of 10.

Bin2text.c:

Approach:

For this program I assume the given binary input is in the same format as the binary file made through text2bin.c. First fseek() is used to go to the end of the file, to call ftell() to get the total size of the input file. This is used as an upper limit for the while loop to know when to stop writing to the output file. Rewind() is used to move the file pointer back to the beginning of the file. Then a while loop starts that continues until a the counter i is greater than or equal to the size of the file. Every iteration of the while loop progress I by the the number of bytes that have

been written. Fread() is used to load the values from the binary file, in the specified order, to their respective variables. This works because the binary file is stored with a set number of bytes for each parameter and in s specified order. After these values are read into their variables, the are written to a file using fprintf() to write them into a line in the output file in the proper format. This is repeated line by line until the counter (i) is greater than the previously specified size of the input file. After this the files are closed.

Timings:

100,000 ratings:

```
real 0.06
user 0.01
sys 0.02
```

1 million ratings:

```
real 0.50
user 0.16
sys 0.17
```

10 million ratings:

```
real 5.05
user 1.86
sys 1.65
```

Every time the data set is increased by a factor of 10, the time it takes is also roughly increased by a factor of 10.

Bin2indexed.c:

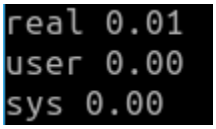
Approach:

This program combines the ideas used in the previous two programs. Firstly, the size of the second input file(the .item file), has it's size read to use later for the while loop. I looked at the the max item_id in the item file to use with malloc() to dynamically allocate memory for the array that stores all the offsets. Then, fgets is run to iterate through the item file line-by-line. Each iteration of the loop, uses sscanf to get the item_if from the beginning of the line and store it in the variable item. We then make offsets[item] equal to ftell() from the line before. This sets each index to the proper offset. After the offsets array is created, the while loop begins with the limit of the previously taken size. And a counter that iterates by the number of bytes read from the binary file from each iteration. During each iteration of the while loop. fread() is used in a similar way as in bin2text to get the values of each parameter and store them into the proper parameter. Then, these read in variables are used to with fwrite() to be written to the output file. The only difference is that instead of writing item_id to the fiel after user_id, instead we get the offset using offsets[item_id], to get the offset of the item_id of the item_id that has just

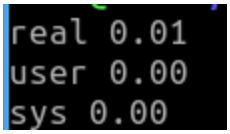
been read. This is written in as an 8 byte integer unlike, the 2 byte integer that was previously used for text2bin. The rating and timestamp are written in as normal after this. This loop continues until the counter is greater than the specified size. After this the files are closed and the offsets array is freed.

Timings:

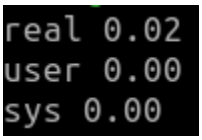
100,000 ratings:



1 million ratings:



10 million ratings:



The times reported are all very low and roughly close together. The 100,000 ratings and 1 million ratings report the same time in fact. However, the time slightly goes up for the 10 miilion ratings. Therefore, I would say that the times scales linearly with slight increase to the time as the data set increases.