Part 1

translateAddress starts by taking the command line input and converts it to an integer using atoi. Then it finds the page number by dividing this address by 4096. Then the offset is found by taking the address and subtracting the page number found in the previous step.

```
ALEXANDERs-MacBook-Air:hw6 alexanderryner$ gcc
translateAddress.c
ALEXANDERs-MacBook-Air:hw6 alexanderryner$ ./a.out 2050000
page number: 500
offset: 2000
ALEXANDERs-MacBook-Air:hw6 alexanderryner$ ./a.out 1000000
page number: 244
offset: 576
ALEXANDERs-MacBook-Air:hw6 alexanderryner$ ./a.out 10000
page number: 2
offset: 1808
ALEXANDERs-MacBook-Air:hw6 alexanderryner$ ./a.out 30
page number: 0
offset: 30
ALEXANDERs-MacBook-Air:hw6 alexanderryner$ ./a.out 23523623652
page number: 500192
offset: 740
ALEXANDERs-MacBook-Air:hw6 alexanderryner$
```

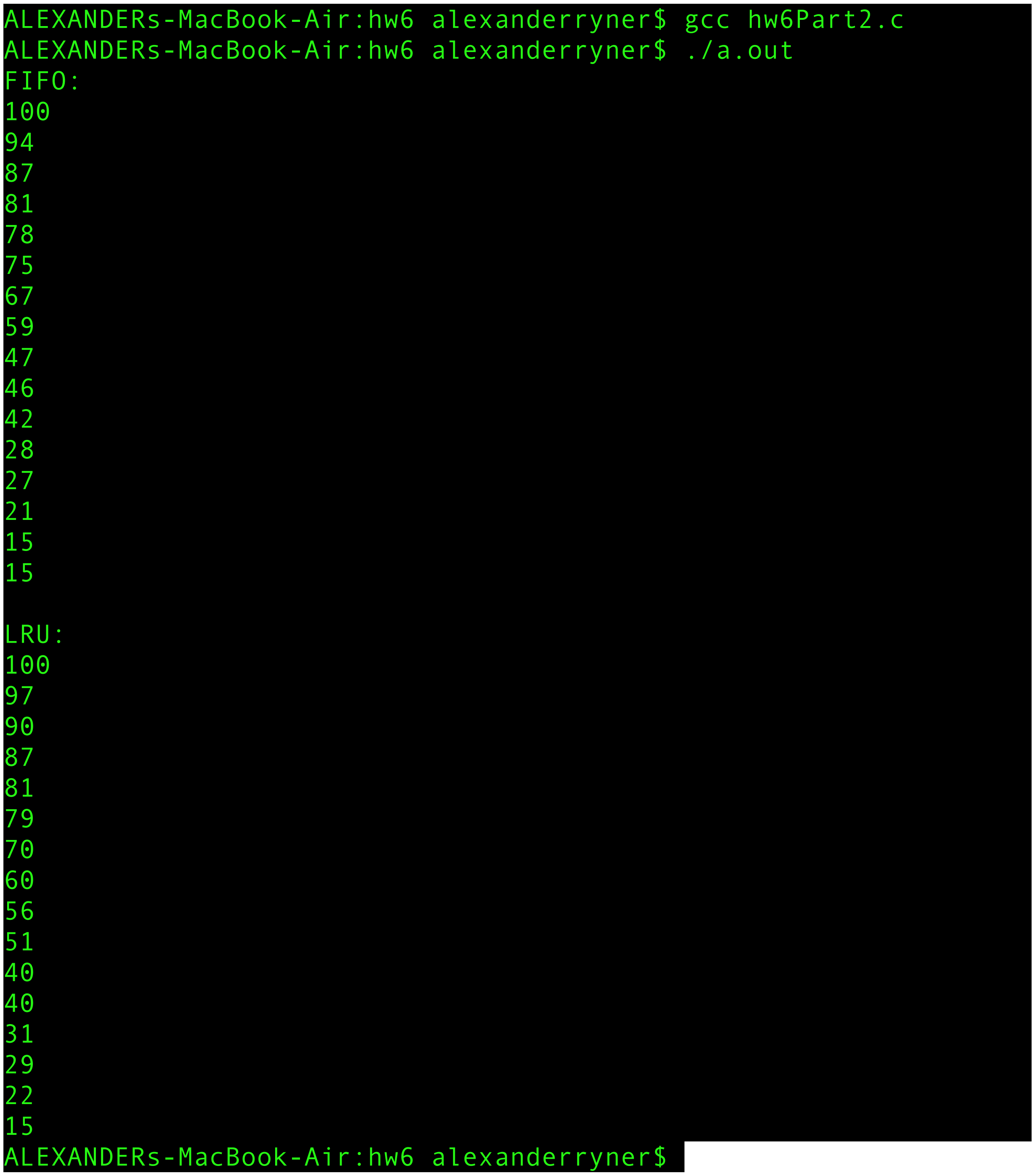
Part 2

For part two I created an array of 100 integers and populated each cell with a random int from 0-15, not letting consecutively indexed ints be the same. I then created an array of 16 ints to be the page table and send it to the zeroTable function I created which actually makes each element -1 not 0. Then I use two for loops to call the functions fifo and LRU 16 times each specifying 1-16 frames.

The fifo function takes a page table(array of ints), the number of frames, reference string and the size of the reference string as arguments. Then we iterate through the reference string checking if the element we are on is in the page table, if it is we move on to the next element. If not we increment the fault variable and set the oldest element in the page table equal to the element of the reference string we are currently on.

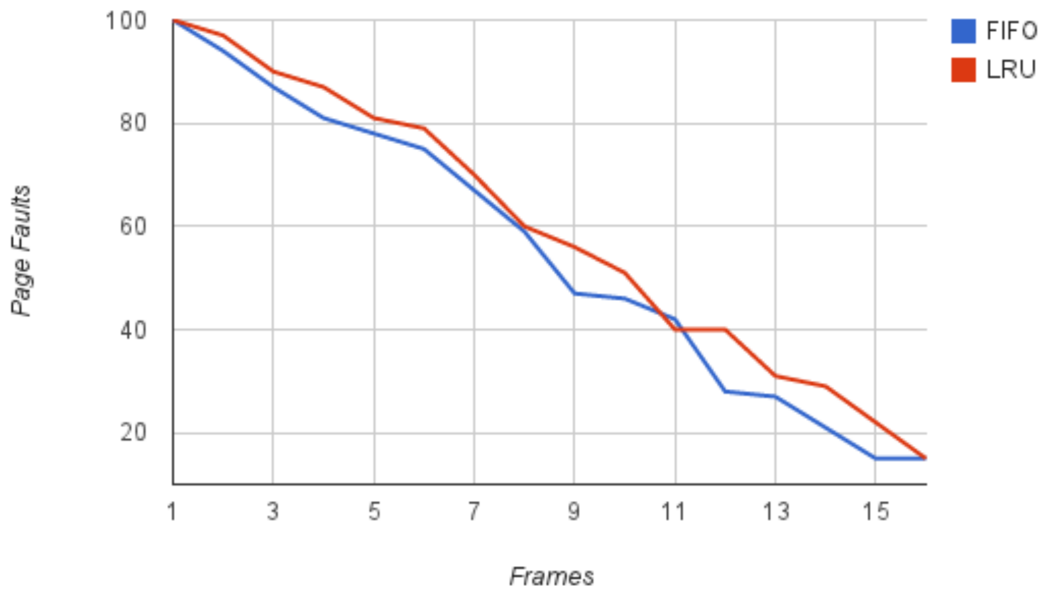
LRU works in a similar way except that when they number of faults are less than the number of frames, each fault simply populates the next unpopulated element of the page table with the element of the reference string we are on and populates and lru array with the index of the page table just populated. When the number of page faults is greater than the number of frames we set table element whose index is the first element of lru to the reference string element we are on. Then we set the last element of the lru array to value of the first element of the lru array after having shifted the rest of the elements one index down. Whenever there is no page fault we move the value in lru that is associated with the index of the page table that had the reference string element in it to the back of lru, shifting the rest of the elements down to make room.

After running the program fifo seems to be a bit better. I did not see Beladys anomaly in my results, the number of frames always either decreased or did not change the number of page faults.



```
ALEXANDERs-MacBook-Air:hw6 alexanderryner$ gcc hw6Part2.c
ALEXANDERs-MacBook-Air:hw6 alexanderryner$ ./a.out
FIFO:
100
94
87
81
78
75
67
59
47
46
42
28
27
21
15
15

LRU:
100
97
90
87
81
79
70
60
56
51
40
40
31
29
22
15
ALEXANDERs-MacBook-Air:hw6 alexanderryner$
```



Page Faults
100
80
60
40
20
1
3
5
7
9
11
13
15
Frames
FIFO
LRU