I had a ton of fun coming up with each of the algorithms for the virtual memory simulator. I added an option in the menu to automatically set the reference string to make it easier for testing. I decided to just leave it in. But as I'm writing this after finishing all the test case screenshots, I realized I should have also had an option to automatically set the string from the handout.

I used quite a few data structures (Array Lists, Vectors, Linked Lists, and a Hash Table) to make the algorithms work. Each algorithm begins by checking to see if the virtual frame is already in the data structure. If it is, that is a hit. If it is not in the data structure, then add then add it after assessing the current size of the data structure. If the current size of the data structure is equal to the number of physical frames then each algorithm will perform some checks to determine a victim frame. Here is a quick summary of how each of the algorithm works:

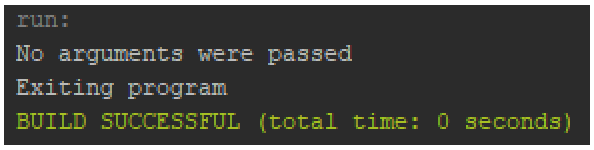
FIFO: If linked list size = frames then boot the first virtual frame in the list using removeFirst().

OPT: If vector size = frames then determine the optimal frames. Iterate through the vector, if the virtual frame appears later down the road then add it to the visited array list. An integer named furthest is used to keep track of which frame appears furthest away. If a virtual frame has been visited, it won't be visited again (keeps track of the closest duplicate). If all frames were visited (visited size = frame size) then boot the frame at index(furthest). If not all frames were visited (visited size < frame size), then boot the first frame in the vector that was not visited (FIFO).

LRU: If vector size = frames then find least recent frame. To do so, visit previous frames in the string. Create a linked list, adding each frame as it appears uniquely (duplicates aren't added again). Once the size of the array list = frames/vector then remove the last element in the linked list using removeLast().

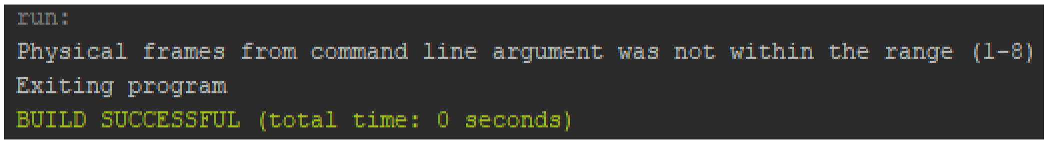
LFU: If vector size = frames then find frame with the lowest frequency. A hashtable is used to keep track of each frame's frequency. The key for the hashtable is the frame itself, the value is the frequency. To determine which value to remove, iterate through all of the current frames to find their frequency. Use an integer called leastFrequent to keep track of lowest frequency. Then, iterate through the frames vector. Upon encountering a frame that matches the lowest frequency, remove it (FIFO).

Test cases of aspects of the program in the form of screenshots begin below:



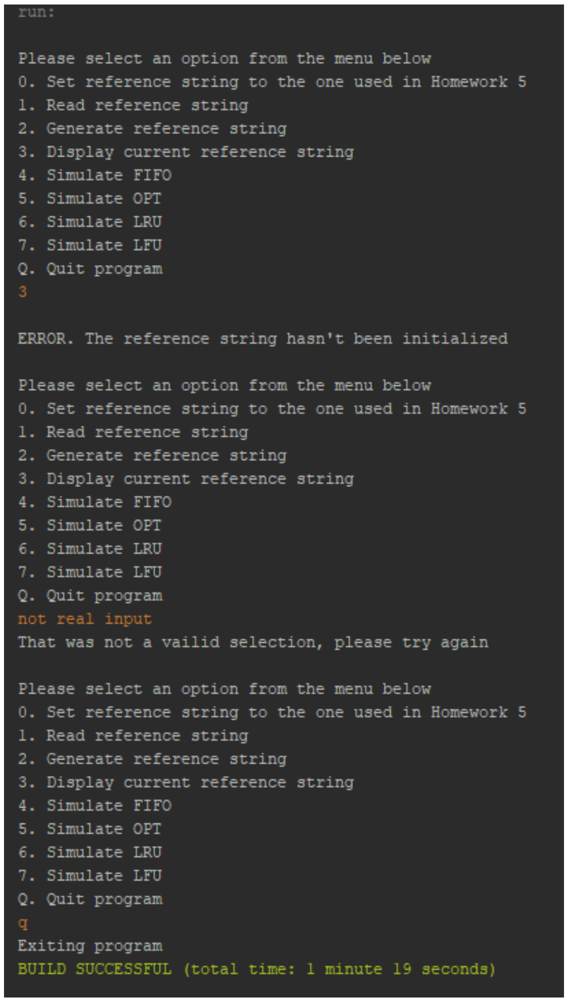
```
run:
No arguments were passed
Exiting program
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 1. Test case #1. Running the program without any arguments.



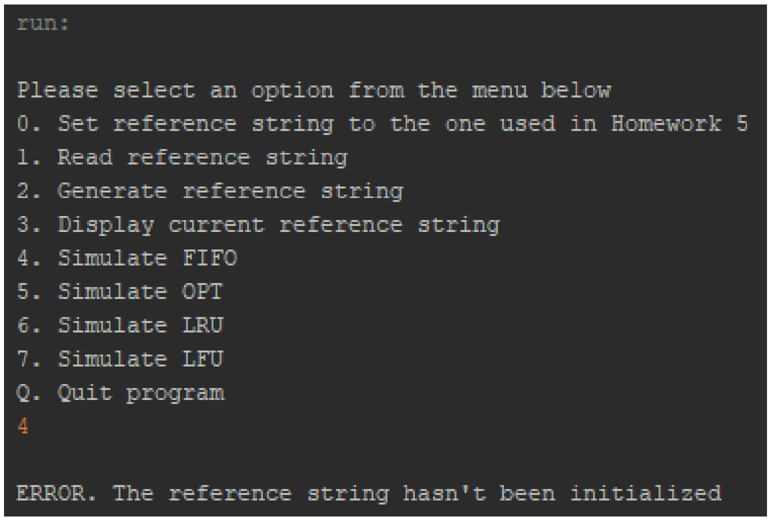
```
run:
Physical frames from command line argument was not within the range (1-8)
Exiting program
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 2. Test case #2. Running the program with arguments outside the range.



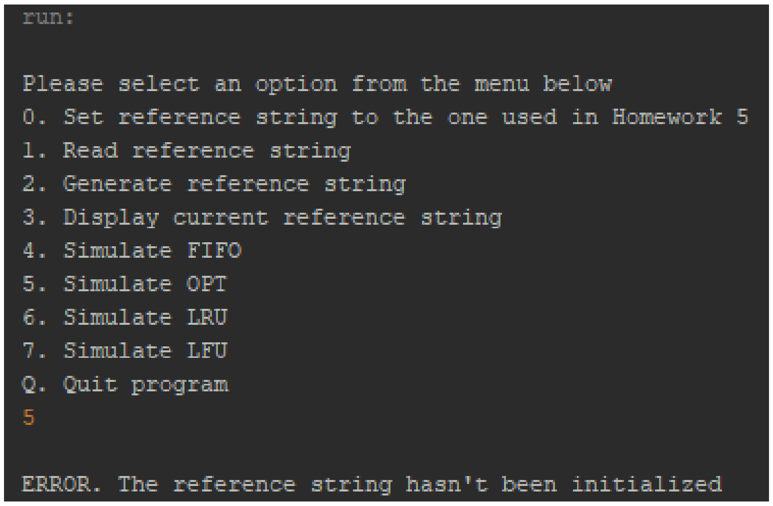
```
run:

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
3

ERROR. The reference string hasn't been initialized

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
not real input
That was not a vailid selection, please try again

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
q
Exiting program
BUILD SUCCESSFUL (total time: 1 minute 19 seconds)
```

Figure 3. Test case #3. Trying to display the reference string without it being initialized. Also showing how invalid selections/constant loops are handled and exit program on q press.



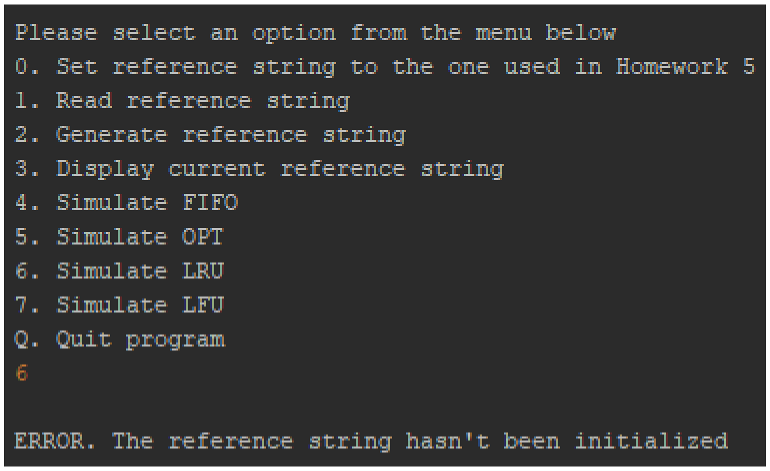
```
run:

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
4

ERROR. The reference string hasn't been initialized
```

Figure 4. Test case #4. Trying to simulate FIFO without a reference string.



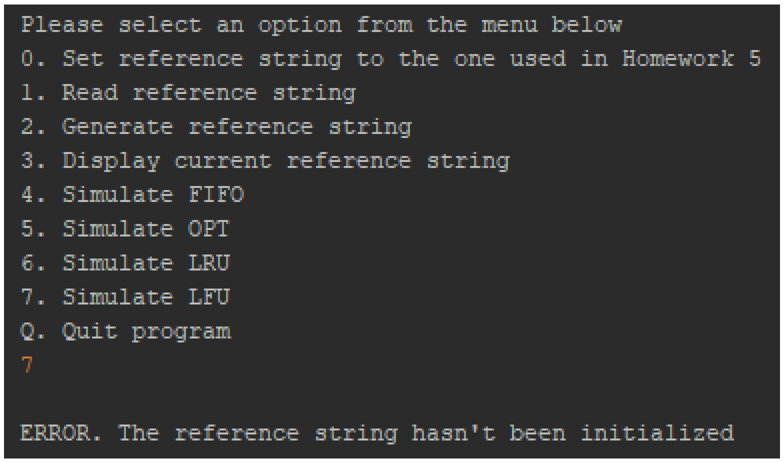
```
run:

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
5

ERROR. The reference string hasn't been initialized
```

Figure 5. Test case #5. Trying to simulate OPT without a reference string.



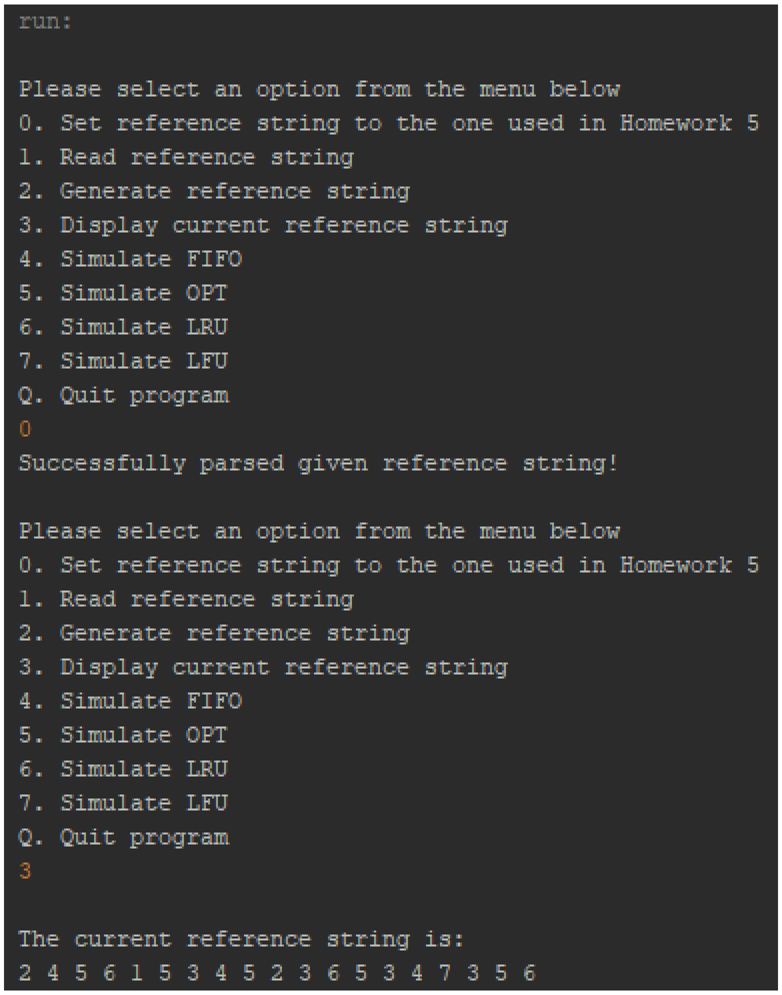
```
Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
6

ERROR. The reference string hasn't been initialized
```

Figure 6. Test case #6. Trying to simulate LRU without a reference string.



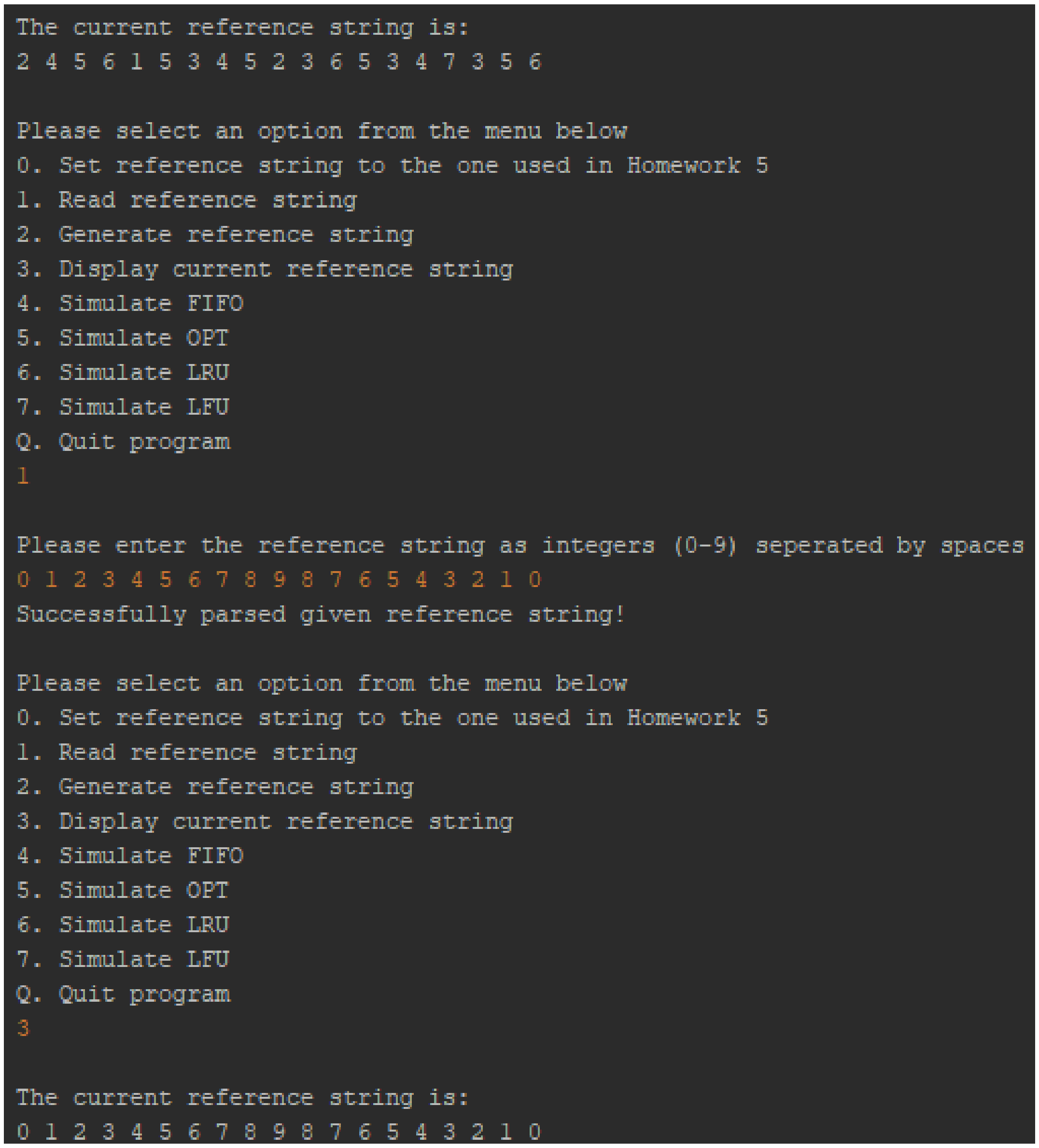
```
Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
7

ERROR. The reference string hasn't been initialized
```

Figure 7. Test case #7. Trying to simulate LFU without a reference string.



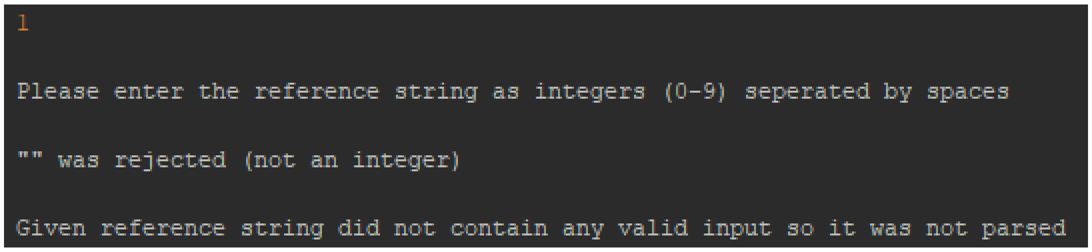
```
run:

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
0
Successfully parsed given reference string!

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
3

The current reference string is:
2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 6
```

Figure 8. Test case #8. Using option 0 to easily select the reference string from homework 5.



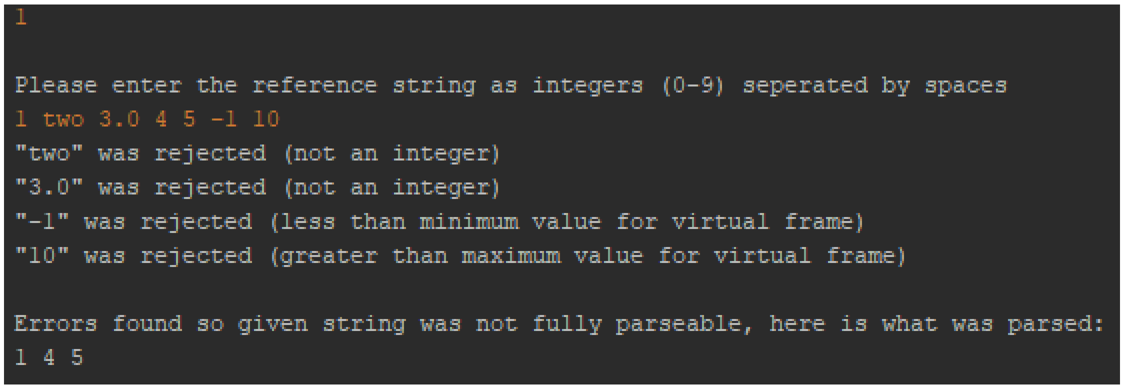
```
The current reference string is:
2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 6

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
1

Please enter the reference string as integers (0-9) seperated by spaces
0 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 0
Successfully parsed given reference string!

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
3

The current reference string is:
0 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 0
```

Figure 9. Test case #9. Reading a reference string from user input and overriding current reference.



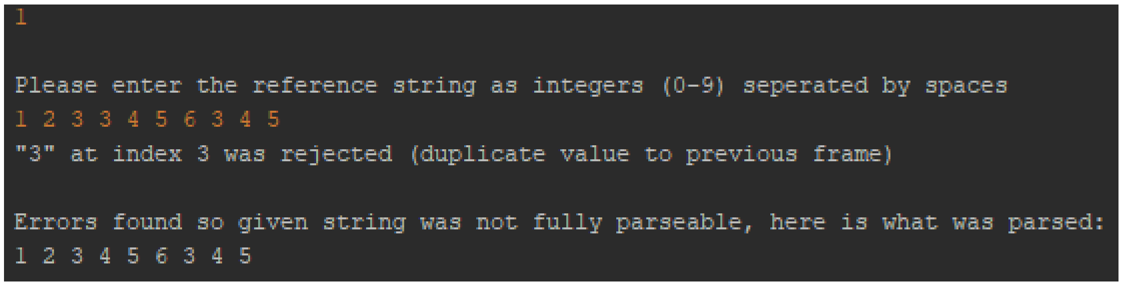
```
1

Please enter the reference string as integers (0-9) seperated by spaces

"" was rejected (not an integer)

Given reference string did not contain any valid input so it was not parsed
```

Figure 10. Test case #10. Using option 1 and giving no input.



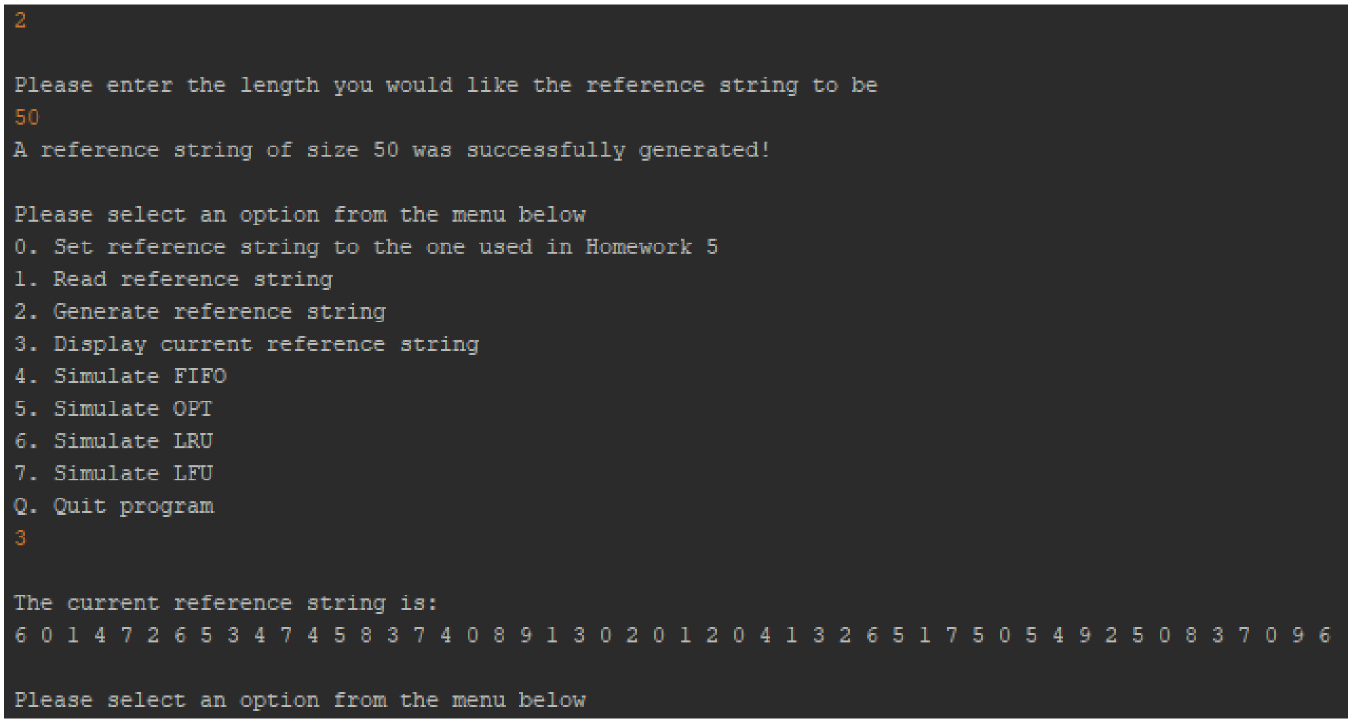
```
1

Please enter the reference string as integers (0-9) seperated by spaces
1 two 3.0 4 5 -1 10
"two" was rejected (not an integer)
"3.0" was rejected (not an integer)
"-1" was rejected (less than minimum value for virtual frame)
"10" was rejected (greater than maximum value for virtual frame)

Errors found so given string was not fully parseable, here is what was parsed:
1 4 5
```

Figure 11. Test case #11. Partial parse of option 1 (rejected string, double, and integers not within range).



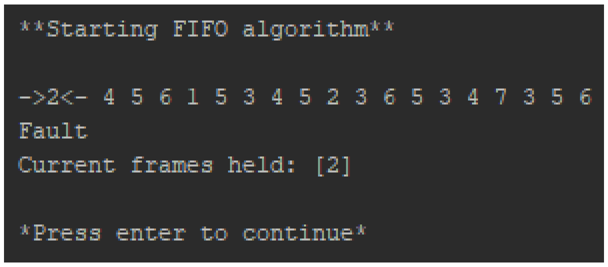
```
1

Please enter the reference string as integers (0-9) seperated by spaces
1 2 3 3 4 5 6 3 4 5
"3" at index 3 was rejected (duplicate value to previous frame)

Errors found so given string was not fully parseable, here is what was parsed:
1 2 3 4 5 6 3 4 5
```

Figure 12. Test case #12. Partial parse of option 1 (rejected neighbor duplicate).



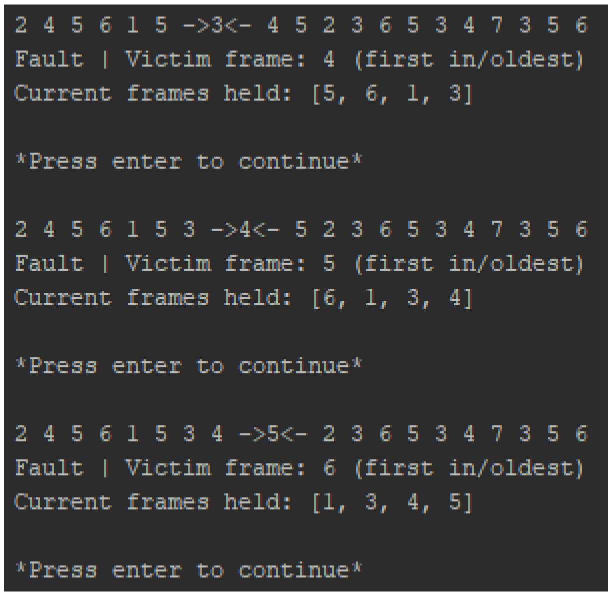
```
2

Please enter the length you would like the reference string to be
50
A reference string of size 50 was successfully generated!

Please select an option from the menu below
0. Set reference string to the one used in Homework 5
1. Read reference string
2. Generate reference string
3. Display current reference string
4. Simulate FIFO
5. Simulate OPT
6. Simulate LRU
7. Simulate LFU
Q. Quit program
3

The current reference string is:
6 0 1 4 7 2 6 5 3 4 7 4 5 8 3 7 4 0 8 9 1 3 0 2 0 1 2 0 4 1 3 2 6 5 1 7 5 0 5 4 9 2 5 0 8 3 7 0 9 6

Please select an option from the menu below
```

Figure 13. Test case #13. Showing a string generated by option 2 and how no duplicate neighbors were generated.



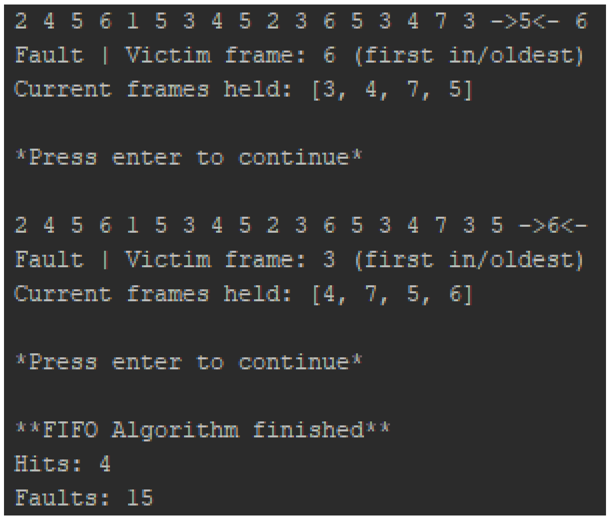
```
**Starting FIFO algorithm**

->2<- 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 6
Fault
Current frames held: [2]

*Press enter to continue*
```

Figure 14. Test case #14. Start of FIFO algorithm on Homework 5’s reference string.



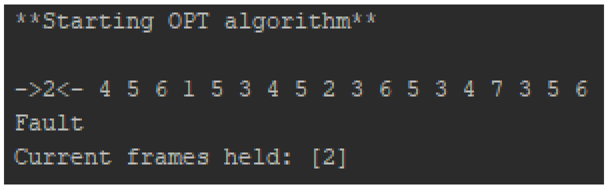
```
2 4 5 6 1 5 ->3<- 4 5 2 3 6 5 3 4 7 3 5 6
Fault | Victim frame: 4 (first in/oldest)
Current frames held: [5, 6, 1, 3]

*Press enter to continue*

2 4 5 6 1 5 3 ->4<- 5 2 3 6 5 3 4 7 3 5 6
Fault | Victim frame: 5 (first in/oldest)
Current frames held: [6, 1, 3, 4]

*Press enter to continue*

2 4 5 6 1 5 3 4 ->5<- 2 3 6 5 3 4 7 3 5 6
Fault | Victim frame: 6 (first in/oldest)
Current frames held: [1, 3, 4, 5]

*Press enter to continue*
```

Figure 15. Test case #15. Midpoint of FIFO algorithm on Homework 5’s reference string.



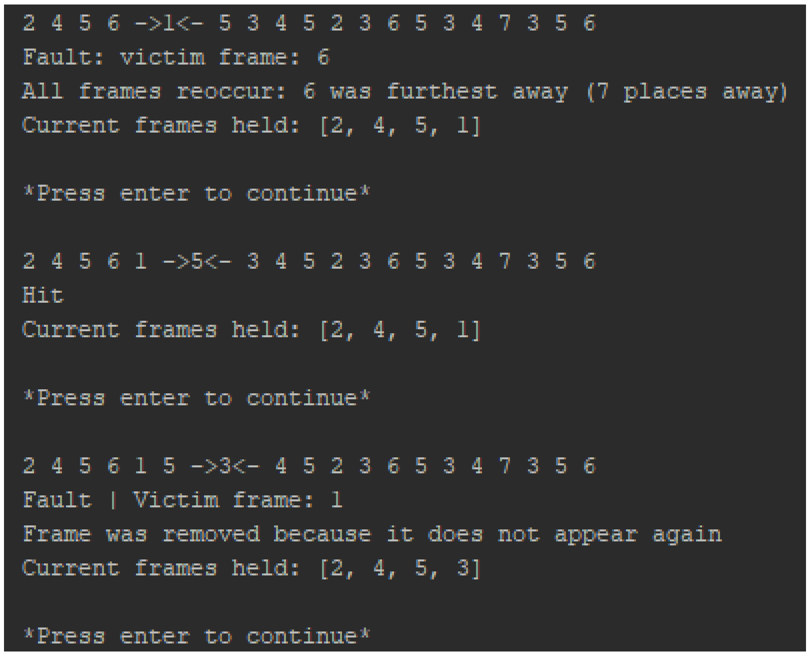
```
2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 ->5<- 6
Fault | Victim frame: 6 (first in/oldest)
Current frames held: [3, 4, 7, 5]

*Press enter to continue*

2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 ->6<-
Fault | Victim frame: 3 (first in/oldest)
Current frames held: [4, 7, 5, 6]

*Press enter to continue*

**FIFO Algorithm finished**
Hits: 4
Faults: 15
```

Figure 16. Test case #16. End of FIFO algorithm on Homework 5’s reference string.



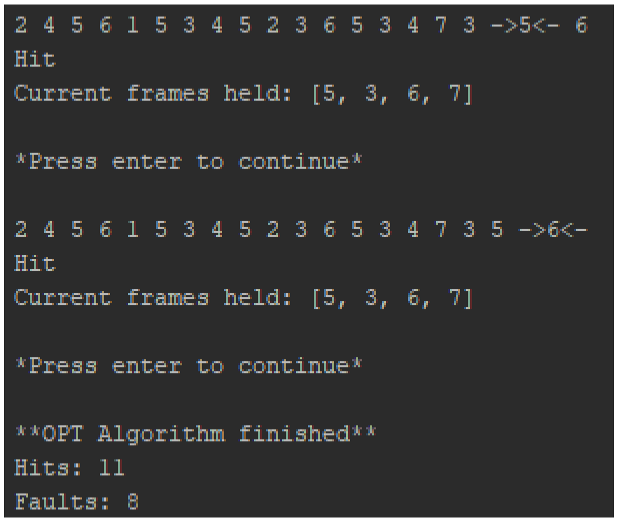
```
**Starting OPT algorithm**

->2<- 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 6
Fault
Current frames held: [2]
```

Figure 17. Test case #17. Start of OPT algorithm on Homework 5’s reference string.



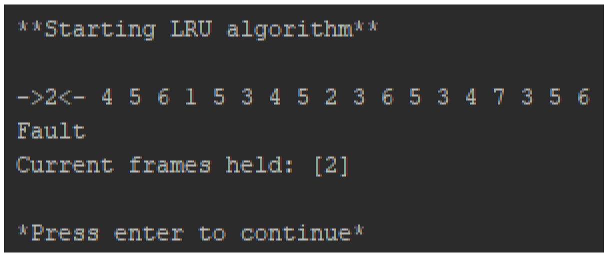
```
2 4 5 6 ->1<- 5 3 4 5 2 3 6 5 3 4 7 3 5 6
Fault: victim frame: 6
All frames reoccur: 6 was furthest away (7 places away)
Current frames held: [2, 4, 5, 1]

*Press enter to continue*

2 4 5 6 1 ->5<- 3 4 5 2 3 6 5 3 4 7 3 5 6
Hit
Current frames held: [2, 4, 5, 1]

*Press enter to continue*

2 4 5 6 1 5 ->3<- 4 5 2 3 6 5 3 4 7 3 5 6
Fault | Victim frame: 1
Frame was removed because it does not appear again
Current frames held: [2, 4, 5, 3]

*Press enter to continue*
```

Figure 18. Test case #18. Midpoint of OPT algorithm on Homework 5’s reference string.



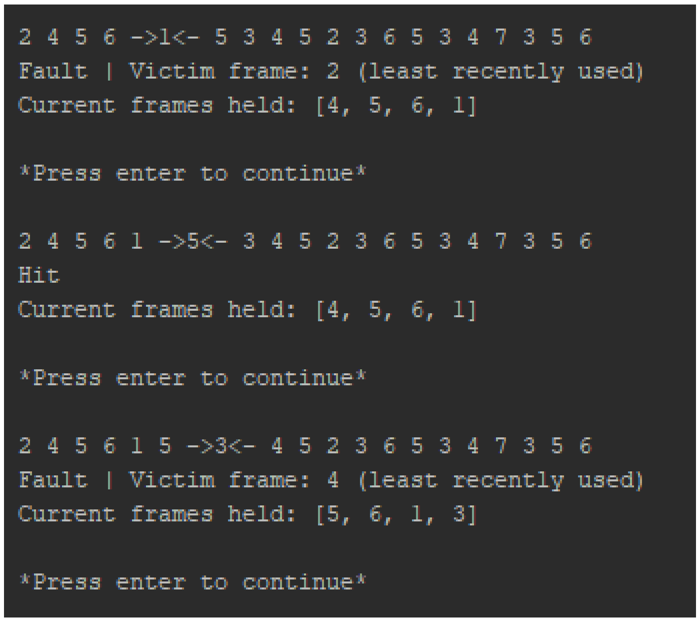
```
2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 ->5<- 6
Hit
Current frames held: [5, 3, 6, 7]

*Press enter to continue*

2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 ->6<-
Hit
Current frames held: [5, 3, 6, 7]

*Press enter to continue*

**OPT Algorithm finished**
Hits: 11
Faults: 8
```

Figure 19. Test case #19. End of OPT algorithm on Homework 5’s reference string.



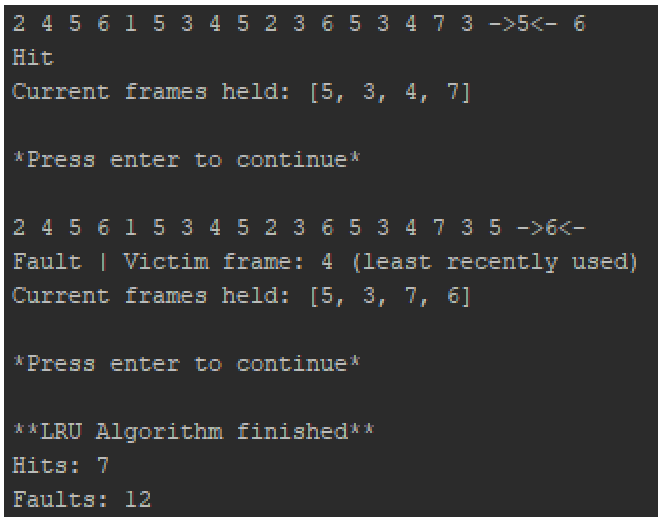
```
**Starting LRU algorithm**

->2<- 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 6
Fault
Current frames held: [2]

*Press enter to continue*
```

Figure 20. Test case #20. Start of LRU algorithm on Homework 5's reference string.



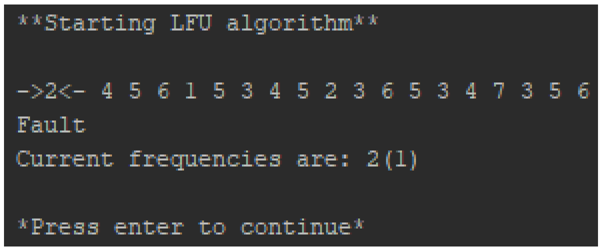
```
2 4 5 6 ->1<- 5 3 4 5 2 3 6 5 3 4 7 3 5 6
Fault | Victim frame: 2 (least recently used)
Current frames held: [4, 5, 6, 1]

*Press enter to continue*

2 4 5 6 1 ->5<- 3 4 5 2 3 6 5 3 4 7 3 5 6
Hit
Current frames held: [4, 5, 6, 1]

*Press enter to continue*

2 4 5 6 1 5 ->3<- 4 5 2 3 6 5 3 4 7 3 5 6
Fault | Victim frame: 4 (least recently used)
Current frames held: [5, 6, 1, 3]

*Press enter to continue*
```

Figure 21. Test case #21. Midpoint of LRU algorithm on Homework 5's reference string.



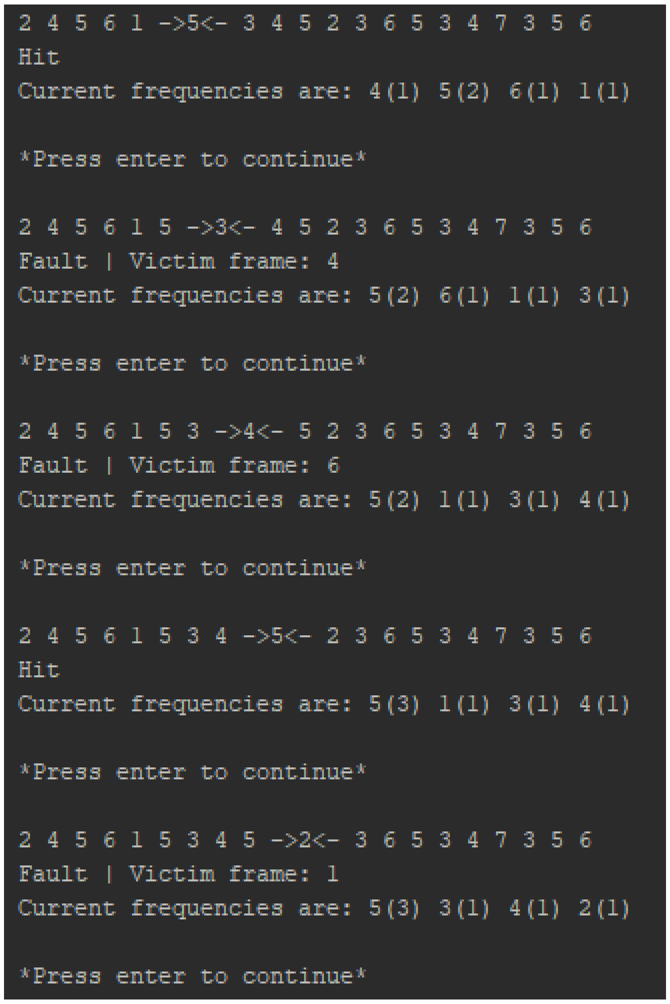
```
2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 ->5<- 6
Hit
Current frames held: [5, 3, 4, 7]

*Press enter to continue*

2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 ->6<-
Fault | Victim frame: 4 (least recently used)
Current frames held: [5, 3, 7, 6]

*Press enter to continue*

**LRU Algorithm finished**
Hits: 7
Faults: 12
```

Figure 22. Test case #22. End of LRU algorithm on Homework 5's reference string.



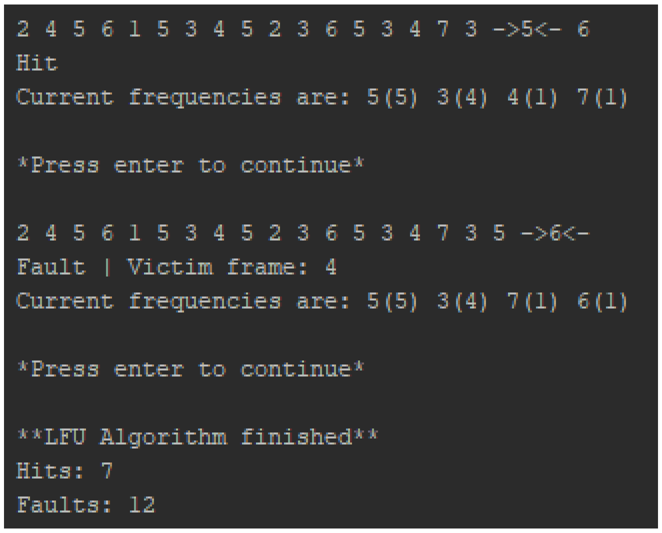
```
**Starting LFU algorithm**

->2<- 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 6
Fault
Current frequencies are: 2(1)

*Press enter to continue*
```

Figure 23. Test case #23. Start of LFU algorithm on Homework 5's reference string.



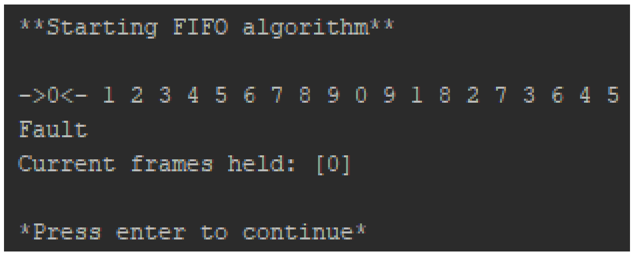
```
2 4 5 6 1 ->5<- 3 4 5 2 3 6 5 3 4 7 3 5 6
Hit
Current frequencies are: 4(1) 5(2) 6(1) 1(1)

*Press enter to continue*

2 4 5 6 1 5 ->3<- 4 5 2 3 6 5 3 4 7 3 5 6
Fault | Victim frame: 4
Current frequencies are: 5(2) 6(1) 1(1) 3(1)

*Press enter to continue*

2 4 5 6 1 5 3 ->4<- 5 2 3 6 5 3 4 7 3 5 6
Fault | Victim frame: 6
Current frequencies are: 5(2) 1(1) 3(1) 4(1)

*Press enter to continue*

2 4 5 6 1 5 3 4 ->5<- 2 3 6 5 3 4 7 3 5 6
Hit
Current frequencies are: 5(3) 1(1) 3(1) 4(1)

*Press enter to continue*

2 4 5 6 1 5 3 4 5 ->2<- 3 6 5 3 4 7 3 5 6
Fault | Victim frame: 1
Current frequencies are: 5(3) 3(1) 4(1) 2(1)

*Press enter to continue*
```

Figure 24. Test case #24. Midpoint of LFU algorithm on Homework 5's reference string.



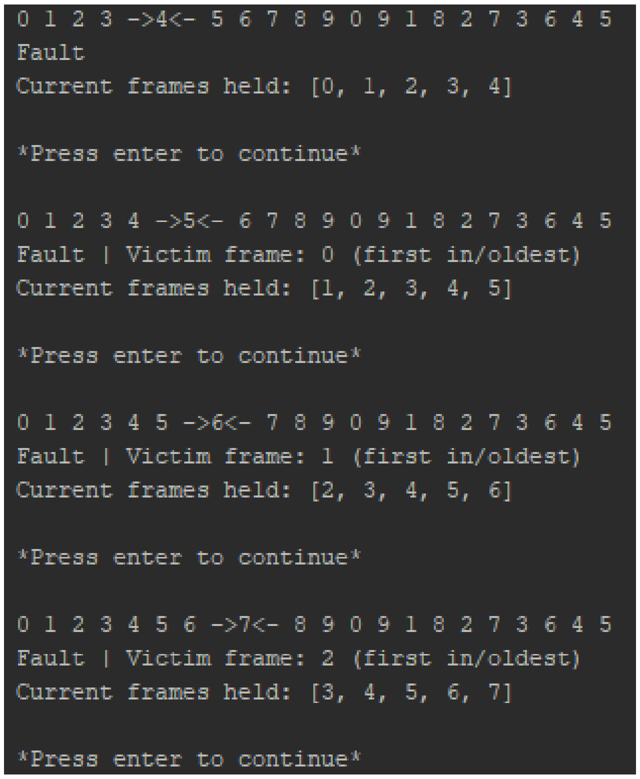
```
2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 ->5<- 6
Hit
Current frequencies are: 5(5) 3(4) 4(1) 7(1)

*Press enter to continue*

2 4 5 6 1 5 3 4 5 2 3 6 5 3 4 7 3 5 ->6<-
Fault | Victim frame: 4
Current frequencies are: 5(5) 3(4) 7(1) 6(1)

*Press enter to continue*

**LFU Algorithm finished**
Hits: 7
Faults: 12
```

Figure 25. Test case #25. End of LFU algorithm on Homework 5’s reference string.



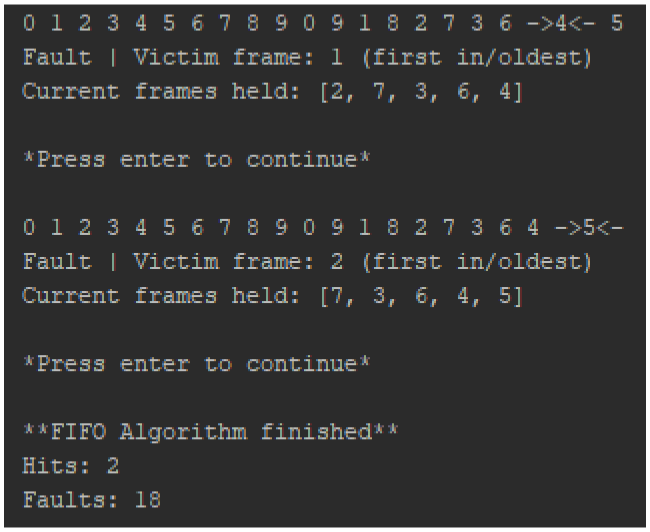
```
**Starting FIFO algorithm**

->0<- 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 4 5
Fault
Current frames held: [0]

*Press enter to continue*
```

Figure 26. Test case #26. Start of FIFO algorithm on the handout’s reference string.



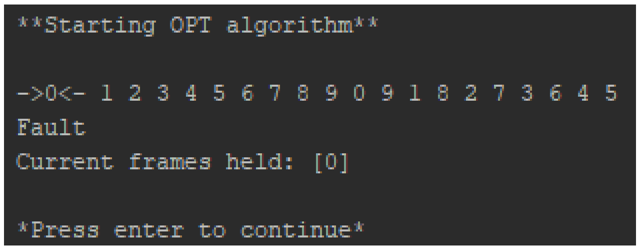
```
0 1 2 3 ->4<- 5 6 7 8 9 0 9 1 8 2 7 3 6 4 5
Fault
Current frames held: [0, 1, 2, 3, 4]

*Press enter to continue*

0 1 2 3 4 ->5<- 6 7 8 9 0 9 1 8 2 7 3 6 4 5
Fault | Victim frame: 0 (first in/oldest)
Current frames held: [1, 2, 3, 4, 5]

*Press enter to continue*

0 1 2 3 4 5 ->6<- 7 8 9 0 9 1 8 2 7 3 6 4 5
Fault | Victim frame: 1 (first in/oldest)
Current frames held: [2, 3, 4, 5, 6]

*Press enter to continue*

0 1 2 3 4 5 6 ->7<- 8 9 0 9 1 8 2 7 3 6 4 5
Fault | Victim frame: 2 (first in/oldest)
Current frames held: [3, 4, 5, 6, 7]

*Press enter to continue*
```

Figure 27. Test case #27. Midpoint of FIFO algorithm on the handout’s reference string.



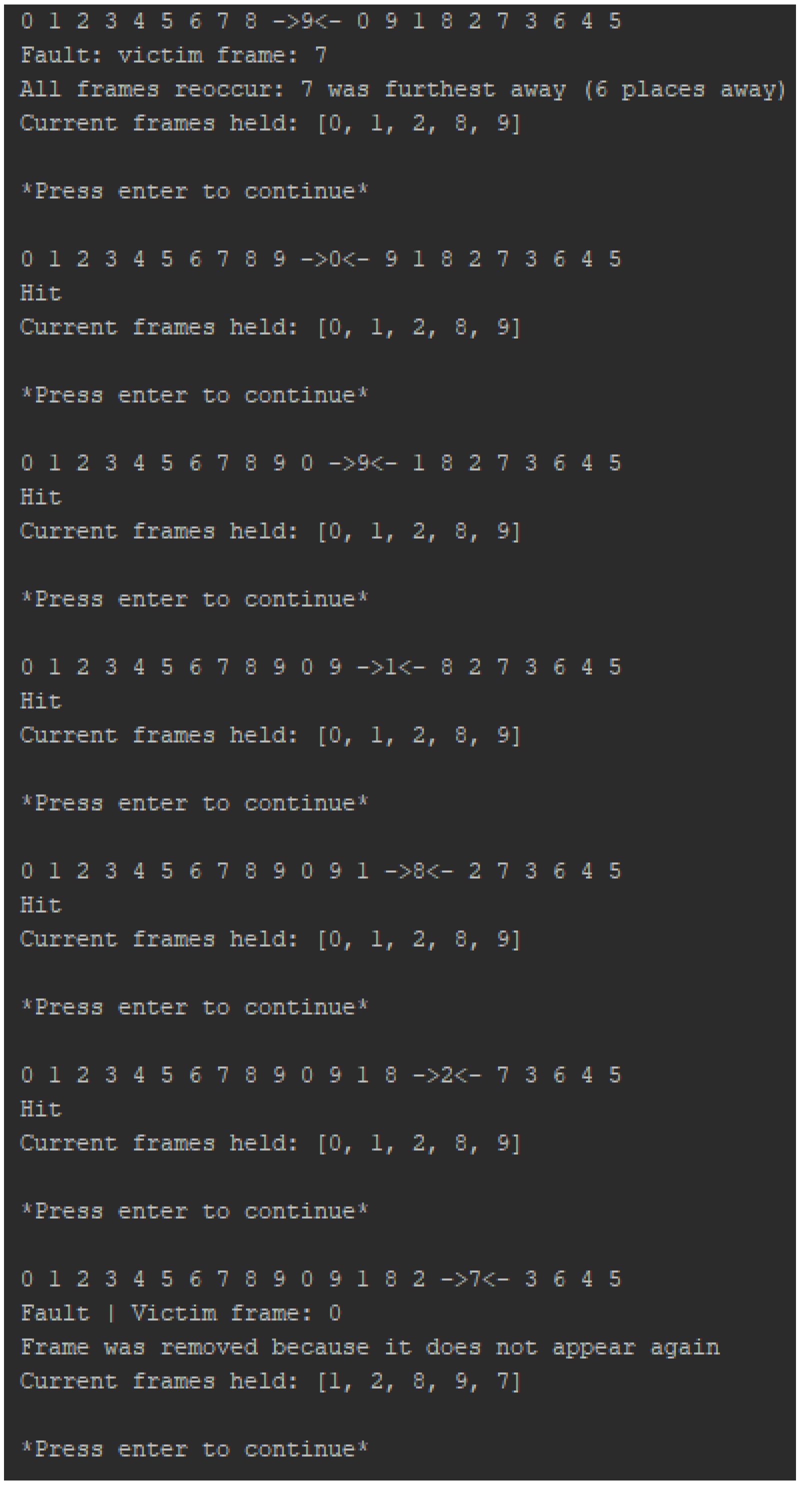
```
0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 ->4<- 5
Fault | Victim frame: 1 (first in/oldest)
Current frames held: [2, 7, 3, 6, 4]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 4 ->5<-
Fault | Victim frame: 2 (first in/oldest)
Current frames held: [7, 3, 6, 4, 5]

*Press enter to continue*

**FIFO Algorithm finished**
Hits: 2
Faults: 18
```

Figure 28. Test case #28. End of FIFO algorithm on the handout’s reference string.



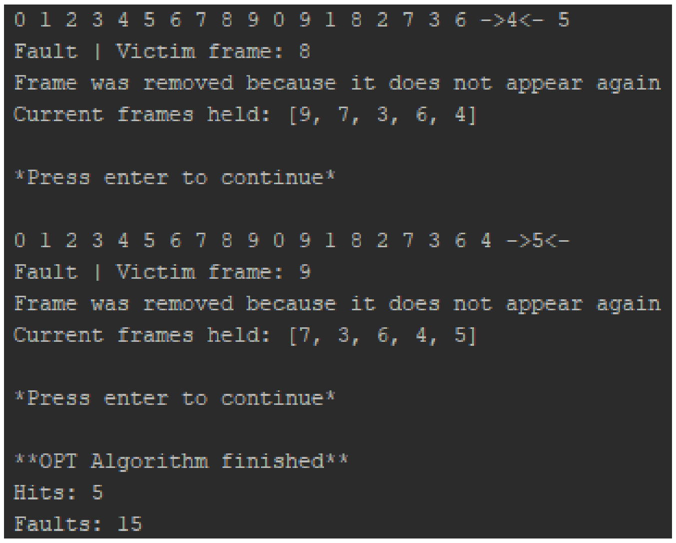
```
**Starting OPT algorithm**

->0<- 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 4 5
Fault
Current frames held: [0]

*Press enter to continue*
```

Figure 29. Test case #29. Start of OPT algorithm on the handout’s reference string.



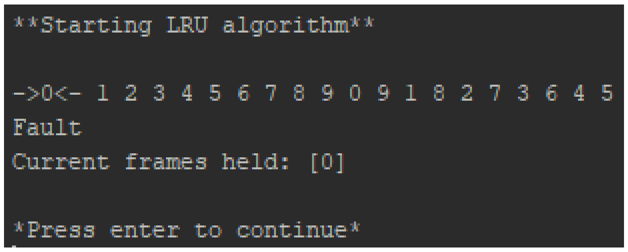
```
0 1 2 3 4 5 6 7 8 ->9<- 0 9 1 8 2 7 3 6 4 5
Fault: victim frame: 7
All frames reoccur: 7 was furthest away (6 places away)
Current frames held: [0, 1, 2, 8, 9]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 ->0<- 9 1 8 2 7 3 6 4 5
Hit
Current frames held: [0, 1, 2, 8, 9]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 ->9<- 1 8 2 7 3 6 4 5
Hit
Current frames held: [0, 1, 2, 8, 9]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 ->1<- 8 2 7 3 6 4 5
Hit
Current frames held: [0, 1, 2, 8, 9]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 1 ->8<- 2 7 3 6 4 5
Hit
Current frames held: [0, 1, 2, 8, 9]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 1 8 ->2<- 7 3 6 4 5
Hit
Current frames held: [0, 1, 2, 8, 9]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 ->7<- 3 6 4 5
Fault | Victim frame: 0
Frame was removed because it does not appear again
Current frames held: [1, 2, 8, 9, 7]

*Press enter to continue*
```

Figure 30. Test case #30. Midpoint of OPT algorithm on the handout’s reference string.



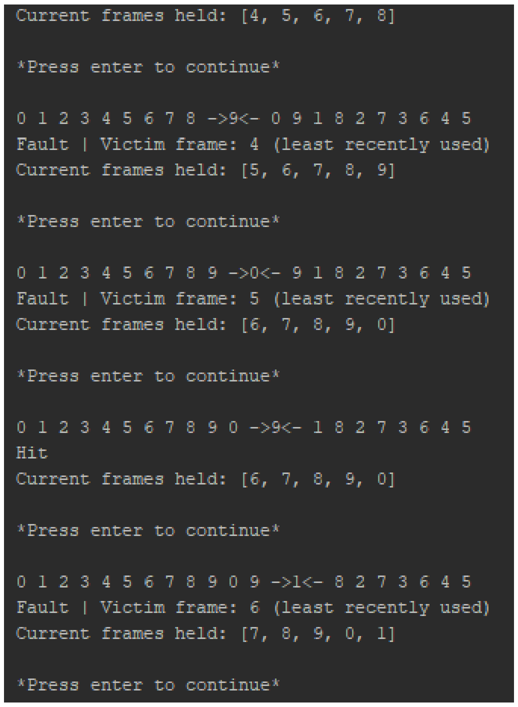
```
0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 ->4<- 5
Fault | Victim frame: 8
Frame was removed because it does not appear again
Current frames held: [9, 7, 3, 6, 4]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 4 ->5<-
Fault | Victim frame: 9
Frame was removed because it does not appear again
Current frames held: [7, 3, 6, 4, 5]

*Press enter to continue*

**OPT Algorithm finished**
Hits: 5
Faults: 15
```

Figure 31. Test case #31. End of OPT algorithm on the handout’s reference string.



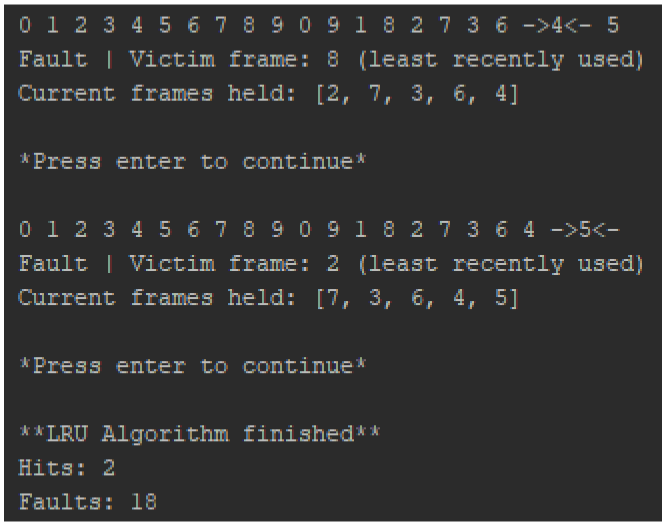
```
**Starting LRU algorithm**

->0<- 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 4 5
Fault
Current frames held: [0]

*Press enter to continue*
```

Figure 32. Test case #32. Start of LRU algorithm on the handout’s reference string.



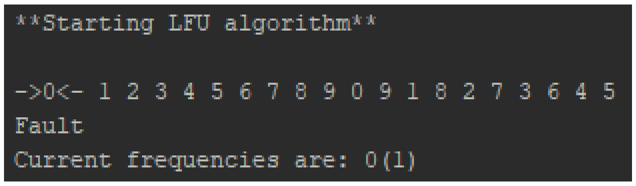
```
Current frames held: [4, 5, 6, 7, 8]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 ->9<- 0 9 1 8 2 7 3 6 4 5
Fault | Victim frame: 4 (least recently used)
Current frames held: [5, 6, 7, 8, 9]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 ->0<- 9 1 8 2 7 3 6 4 5
Fault | Victim frame: 5 (least recently used)
Current frames held: [6, 7, 8, 9, 0]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 ->9<- 1 8 2 7 3 6 4 5
Hit
Current frames held: [6, 7, 8, 9, 0]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 ->1<- 8 2 7 3 6 4 5
Fault | Victim frame: 6 (least recently used)
Current frames held: [7, 8, 9, 0, 1]

*Press enter to continue*
```

Figure 33. Test case #33. Midpoint of LRU algorithm on the handout’s reference string.



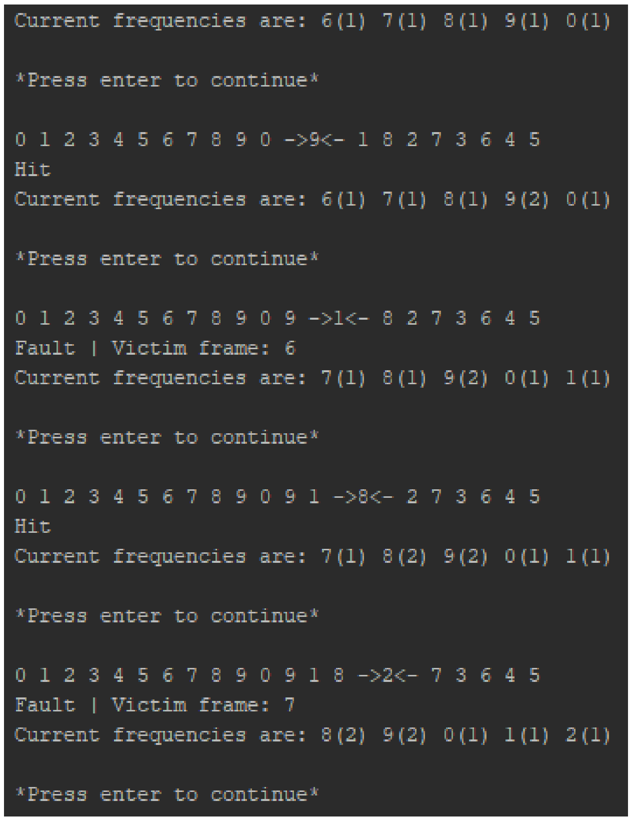
```
0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 ->4<- 5
Fault | Victim frame: 8 (least recently used)
Current frames held: [2, 7, 3, 6, 4]

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 4 ->5<-
Fault | Victim frame: 2 (least recently used)
Current frames held: [7, 3, 6, 4, 5]

*Press enter to continue*

**LRU Algorithm finished**
Hits: 2
Faults: 18
```

Figure 34. Test case #34. End of LRU algorithm on the handout's reference string.



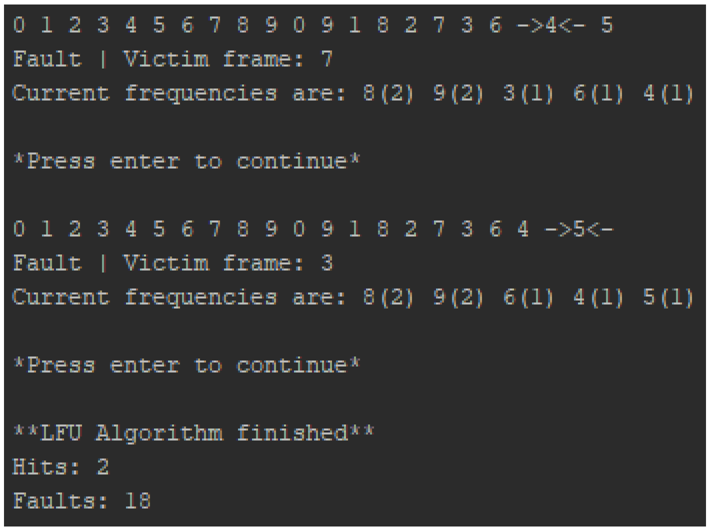
```
**Starting LFU algorithm**

->0<- 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 4 5
Fault
Current frequencies are: 0(1)
```

Figure 35. Test case #35. Start of LFU algorithm on the handout's reference string.



```
Current frequencies are: 6(1) 7(1) 8(1) 9(1) 0(1)

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 ->9<- 1 8 2 7 3 6 4 5
Hit
Current frequencies are: 6(1) 7(1) 8(1) 9(2) 0(1)

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 ->1<- 8 2 7 3 6 4 5
Fault | Victim frame: 6
Current frequencies are: 7(1) 8(1) 9(2) 0(1) 1(1)

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 1 ->8<- 2 7 3 6 4 5
Hit
Current frequencies are: 7(1) 8(2) 9(2) 0(1) 1(1)

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 1 8 ->2<- 7 3 6 4 5
Fault | Victim frame: 7
Current frequencies are: 8(2) 9(2) 0(1) 1(1) 2(1)

*Press enter to continue*
```

Figure 36. Test case #36. Midpoint of LFU algorithm on the handout's reference string.



```
0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 ->4<- 5
Fault | Victim frame: 7
Current frequencies are: 8(2) 9(2) 3(1) 6(1) 4(1)

*Press enter to continue*

0 1 2 3 4 5 6 7 8 9 0 9 1 8 2 7 3 6 4 ->5<-
Fault | Victim frame: 3
Current frequencies are: 8(2) 9(2) 6(1) 4(1) 5(1)

*Press enter to continue*

**LFU Algorithm finished**
Hits: 2
Faults: 18
```

Figure 37. Test case #37. End of LFU algorithm on the handout’s reference string.