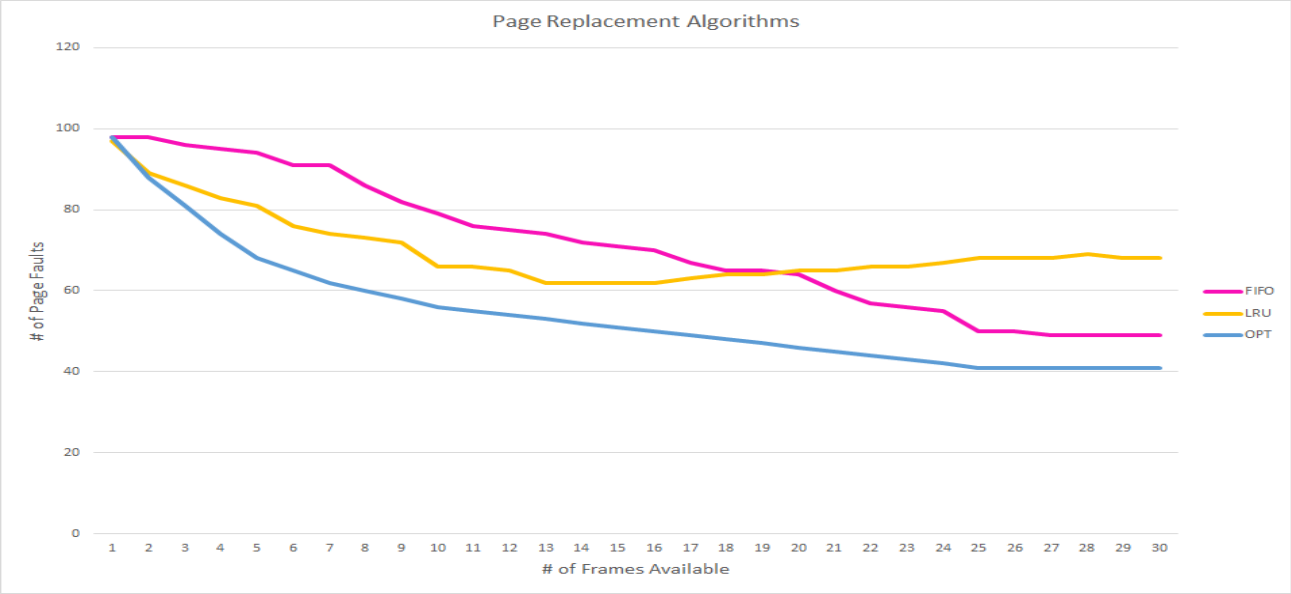
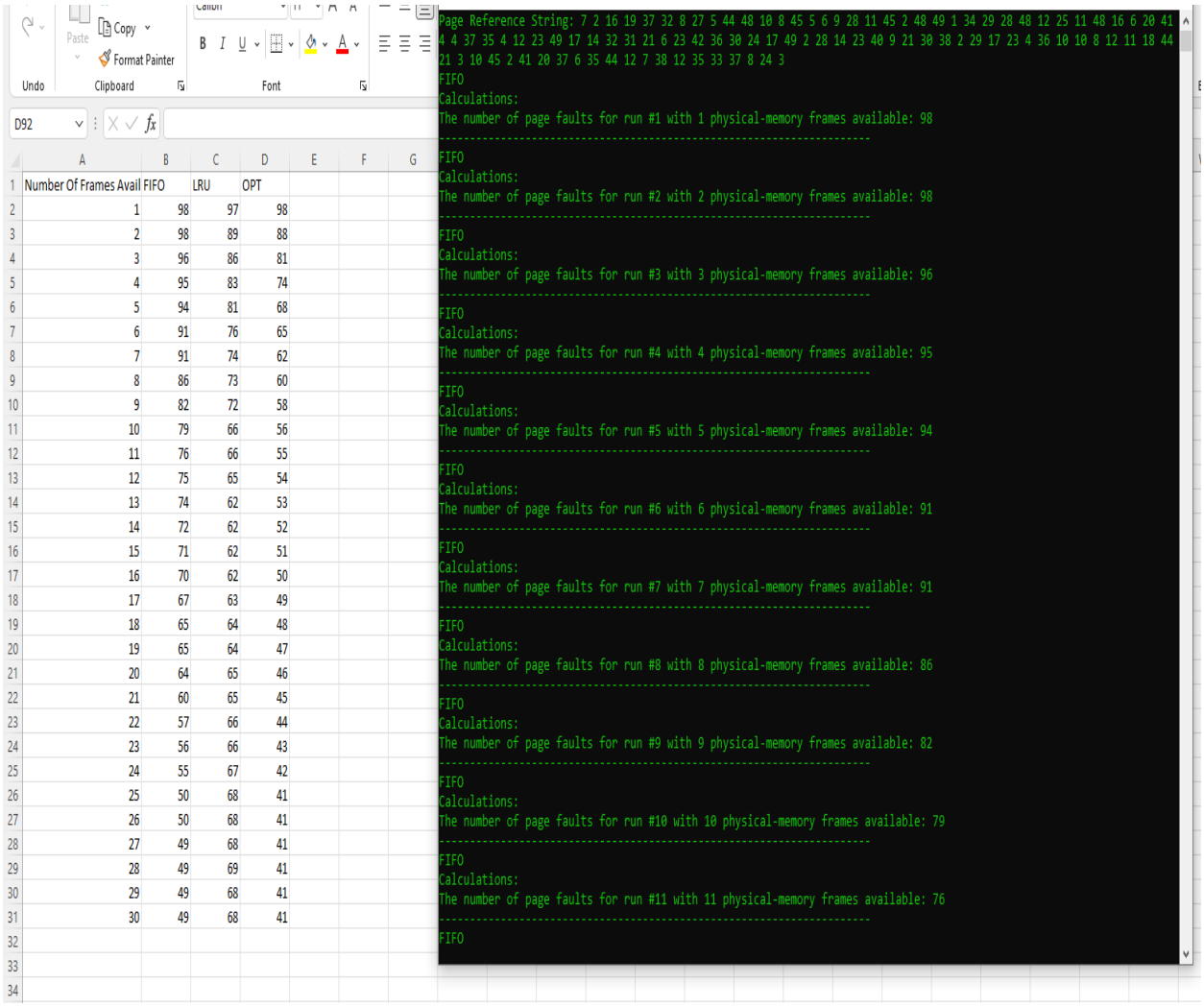
Brief Overview:

This program is a virtual-memory page replacement algorithm simulation that runs different page replacement algorithms and outputs the number of page faults that occur in each run. The program is written in C++. It implements the following page replacement algorithms: First-In First-Out (FIFO), Least Recently Used (LRU), and Optimal (OPT). It computes the number of page faults for varying values, from 1-30, of number of frames available for each simulation that is run. The number of page faults is the metric used in order to compare and analyze the impact of the chosen algorithm utilized on the number of page faults incurred across a varying number of physical-memory page frames available for the page reference string. In each of the algorithms that are run there is a relationship between the number of page faults and the number of frames that are available. This relationship has the number of page faults as a dependent variable to the independent variable number of frames available.

A page replacement algorithm is a method of selecting which page element should be replaced in memory. It is useful when attempting to manage virtual memory in an operating system because it helps to decide which page should be replaced in order to allocate another page that is needed. The main task of page replacement is to attempt to keep the number of page faults at a minimum and optimize physical memory utilization.

Screen Results Of Different Runs Of The Simulation :



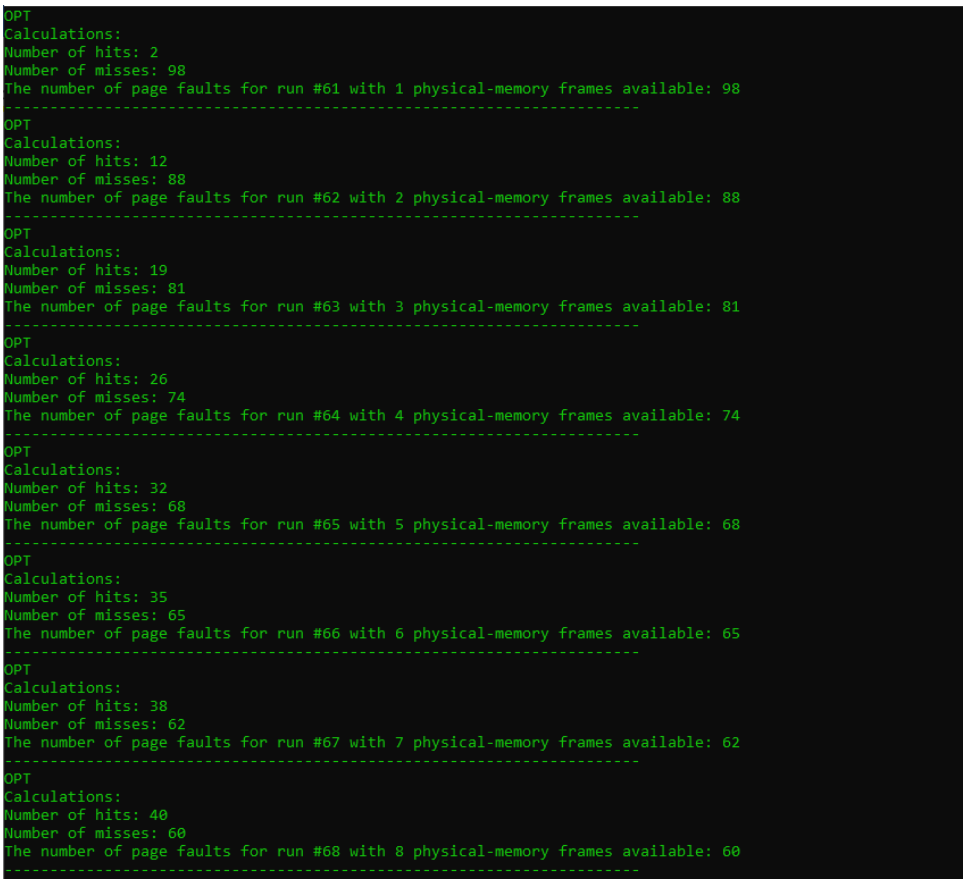


Number Of Frames Avail	FIFO	LRU	OPT
1	98	97	98
2	98	89	88
3	96	86	81
4	95	83	74
5	94	81	68
6	91	76	65
7	91	74	62
8	86	73	60
9	82	72	58
10	79	66	56
11	76	66	55
12	75	65	54
13	74	62	53
14	72	62	52
15	71	62	51
16	70	62	50
17	67	63	49
18	65	64	48
19	65	64	47
20	64	65	46
21	60	65	45
22	57	66	44
23	56	66	43
24	55	67	42
25	50	68	41
26	50	68	41
27	49	68	41
28	49	69	41
29	49	68	41
30	49	68	41

```
Page Reference String: 7 2 16 19 37 32 8 27 5 44 48 10 8 45 5 6 9 28 11 45 2 48 49 1 34 29 28 48 12 25 11 48 16 6 20 41
4 4 37 35 4 12 23 49 17 14 32 31 21 6 23 42 36 30 24 17 49 2 28 14 23 40 9 21 30 38 2 29 17 23 4 36 10 10 8 12 11 18 44
21 3 10 45 2 41 20 37 6 35 44 12 7 38 12 35 33 37 8 24 3
FIFO
Calculations:
The number of page faults for run #1 with 1 physical-memory frames available: 98
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #2 with 2 physical-memory frames available: 98
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #3 with 3 physical-memory frames available: 96
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #4 with 4 physical-memory frames available: 95
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #5 with 5 physical-memory frames available: 94
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #6 with 6 physical-memory frames available: 91
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #7 with 7 physical-memory frames available: 91
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #8 with 8 physical-memory frames available: 86
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #9 with 9 physical-memory frames available: 82
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #10 with 10 physical-memory frames available: 79
----------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #11 with 11 physical-memory frames available: 76
----------------------------------------------------------------------
FIFO
```

```
Page Reference String: 7 2 16 19 37 32 8 27 5 44 48 10 8 45 5 6 9 28 11 45 2 48 49 1 34 29 28 48 12 25 11 48 16 6 20 41
4 4 37 35 4 12 23 49 17 14 32 31 21 6 23 42 36 30 24 17 49 2 28 14 23 40 9 21 30 38 2 29 17 23 4 36 10 10 8 12 11 18 44
21 3 10 45 2 41 20 37 6 35 44 12 7 38 12 35 33 37 8 24 3
FIFO
Calculations:
The number of page faults for run #1 with 1 physical-memory frames available: 98
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #2 with 2 physical-memory frames available: 98
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #3 with 3 physical-memory frames available: 96
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #4 with 4 physical-memory frames available: 95
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #5 with 5 physical-memory frames available: 94
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #6 with 6 physical-memory frames available: 91
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #7 with 7 physical-memory frames available: 91
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #8 with 8 physical-memory frames available: 86
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #9 with 9 physical-memory frames available: 82
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #10 with 10 physical-memory frames available: 79
------------------------------------------------------------------------
FIFO
Calculations:
The number of page faults for run #11 with 11 physical-memory frames available: 76
------------------------------------------------------------------------
FIFO
```

```
LRU
Calculations:
The number of page faults for run #31 with 1 physical-memory frames available: 97
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #32 with 2 physical-memory frames available: 89
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #33 with 3 physical-memory frames available: 86
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #34 with 4 physical-memory frames available: 83
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #35 with 5 physical-memory frames available: 81
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #36 with 6 physical-memory frames available: 76
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #37 with 7 physical-memory frames available: 74
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #38 with 8 physical-memory frames available: 73
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #39 with 9 physical-memory frames available: 72
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #40 with 10 physical-memory frames available: 66
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #41 with 11 physical-memory frames available: 66
----------------------------------------------------------------------
LRU
Calculations:
The number of page faults for run #42 with 12 physical-memory frames available: 65
----------------------------------------------------------------------
```



Design and Implementation:

This program is implemented using queue data structures and methods in order to provide page replacement algorithm functionality to the program. The program uses a queue/array data structure to create a queue of page elements that can be accessed and allocated during the simulation. The program will remove pages from the front of its queue and will remove them from the system once the page element is not currently needed and there are other page elements that need to be added according to the algorithm being utilized. While running the program the simulator generates a list of page elements randomly and places them into a page reference string array. Then for each page the program generates a randomized number and assigns a number of physical memory frames available. After the program generates a random page reference string of 100 page elements with numbers ranging from 0 to 49, it then uses the same page-reference string created initially in the program for each algorithm and records the number of page faults that occur in each run of each algorithm. The program is built to run so that for each run 1 to 30 for an algorithm the number of physical-memory frames available will also be varied from 1 to 30. Each of the 3 different algorithms execute a total of 30 runs for a total of 90 executed runs in the program. The number of page faults is calculated for each run. The data from the program is

displayed on the console and also is sent to a .csv file in order to be loaded to an excel file. The excel output file is then used to create a visual graph of the data. The graph has the number of frames available on the x-axis and the number of page faults on the y-axis. The graph uses a different-color line to represent each of the three algorithms. The optimal page replacement algorithm seems to result in the fewest page faults incurred.

The simulator stops after it has successfully handled a total of 90 runs. After it is done, the program will then output the metrics that we want from the experiment. All throughout the experiment different aspects of the memory management state are updated. The numbers that are used to represent the page element numbers were generated using the srand(time(NULL)) method which calls the generate a random number that can be used to create randomized values. This helps to yield different results that are based on different page reference strings each time that you rerun the program. Once the program has gone through all runs of the experiment the program will terminate. The log output is stored in a file called “myexcel.csv” which I then used to create a plot of the data that was collected from the experiment runs. And the screen output of the code and the metrics should also show in the terminal console.

Some of the challenges of this project were aligning the output to the excel file properly and also in getting the data from multiple runs to be collected and output into a single file without overwriting the previous data. Another issue was finding ways to get rid of errors that showed up when attempting to run the program in Linux servers versus the IDE or in my command line terminal. I am using -std=gnu++0x in order to compile because it allows the program to be able to compile directly from a terminal by including C++11 flags in the command for the compiler.

Instructions On How To Compile And Run The Program On The CS Linux Servers:

If you would like to run the program on the CS Linux Servers and if you are using a software like Putty you can connect to either the eros.cs.txstate.edu or zeus.cs.txstate.edu as the host by typing in the host url name of the server that you would like to connect to. After typing in the host name fill in the Port entry field as “22” and making sure that the “SSH” radio button is selected. Then press open and the terminal will open. In the terminal that pops up, write your username and press enter then write your password and press enter.

I have the program file stored in the students directory folder called “s_l427”. If you do not have access to this folder, you can move the program file from my submission to your personal account and run them from there by unzipping the file and selecting the proj.cpp file to transfer. The log output is stored in a file called “myexcel.csv”. And the screen output should show in the terminal.

To compile the program in Linux type in:

“g++ -std=c++11 -o program3 program3.cpp” and press enter (note: the letter after W is a lowercase L)

Then type in “ ./ program3”and press enter.

Then it will run the program.

Example of what it should look like:

[s_l427@eros ~]$ g++ -std=c++11 -o program3 program3.cpp

[s_l427@eros ~]$ ./ program3