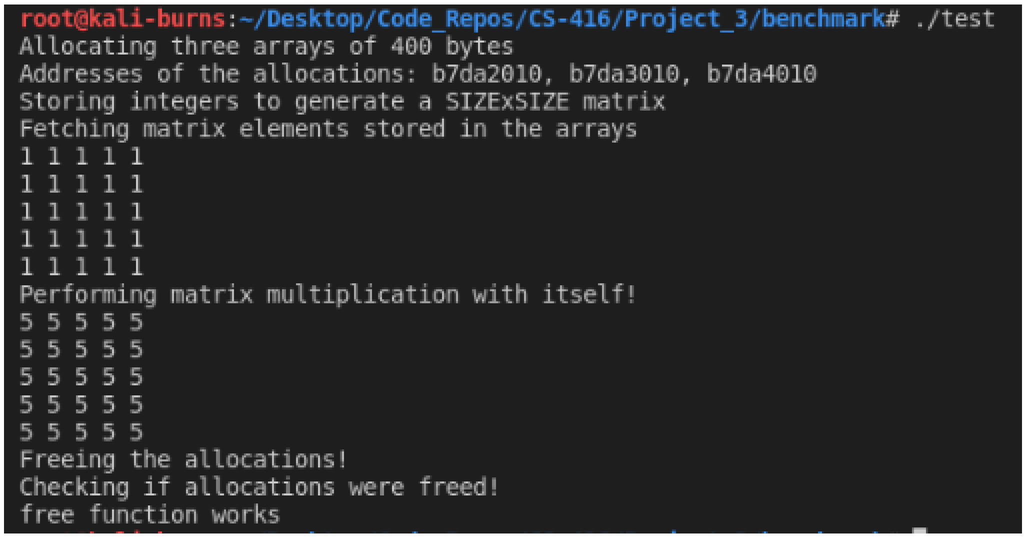
Thomas Burns

tjb234

CS416

Project 3 writeup

I worked solo on this project and was able to successfully implement all required methods and functionalities for this project. I obtained the required output from the test program as shown below. I did not attempt the extra credit for this project.



```
root@kali-burns:~/Desktop/Code_Repos/CS-416/Project_3/benchmark# ./test
Allocating three arrays of 400 bytes
Addresses of the allocations: b7da2010, b7da3010, b7da4010
Storing integers to generate a SIZExSIZE matrix
Fetching matrix elements stored in the arrays
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
Performing matrix multiplication with itself!
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
5 5 5 5 5
Freeing the allocations!
Checking if allocations were freed!
free function works
```

For most of the early stage of this project I was very confused on how to implement the structures required for the mallocs in memory. Talking it out with some friends who are not in the class helped as well as drawing some of my ideas out on paper. I posted on Piazza about my understanding of the implementation structure and then started working with that interpretation. (https://piazza.com/class/k2l7b6odccqk2?cid=15)

I struggled first with trying to have a 2D array declared as a pde_t ** and then even thinking of doing a pde_t* malloced by 1024 which each individual one would each refer to 1024 pte_t*. I finally settled on a very simple and easy to understand approach where I used a 2D array of pde_t pointers (pde_t * array[1024][1024]) which allowed for me to store the physical address of the memory that was allocated but also refer to the page directory and table correctly to index into this array and allocate it in the correct space. Once I understood the 10 bits for both the directory and table indexing (obtained by the address's most significant bit, 10+10+12(for the offset)), I was able to essentially hash into the array I had created and store the address I wanted into the correct space. I made sure that the memory that I was allocating in the physical memory array I maintained was done properly in conjunction with this. I maintained the virtual page states appropriately as well for the proper simulation of memory.

While I was able to get all of the functionalities of this project in accordance with the project guidelines, I was not able to implement any edge testing. My program is not equipped to handle almost any edge case if it were to be presented with one. I did not deal with edge cases in order to ensure that the primary functionalities of the program work as intended by the deadline, but this would be the first thing that I fixed if given additional time. The most glaring issues are most likely in the malloc and free function where I don’t do enough adequate checks or might not even properly handle m_allocs over a page size. I was not able to test for this appropriately as it would require writing an entirely new test file or massively overhauling the given test.c. The final issue I see with my code is I don’t do any actual frees of memory that I allocate at the beginning. Some of these frees are just negligent like in the matrix multiplication method, while others I am not sure of how to handle such as the memory structures I maintain for the fundamental use of the my_vm library. I am not sure when to free those or if I should not at all until the user quits their program entirely. The issues of edge cases and many of the frees, however, could easily be fixed since the core implementation seems to be correct based on the results of the test file given.

The most challenging part of this project besides the theoretical understanding of the project and its design and implementation was writing the matrix multiplication method. The method required multiple dereferences and for my implementation to work I used temporary arrays that I had to fill using the addresses in memory of the matrices meant to be multiplied. After writing my own version of this method and having it absolutely fail due to improper references and pointer casts and types I had to follow the typing that was used in the test file to ensure that I was using the different addresses and the values at each address properly. Once I did this, with some minor edits I was able to get it to work.