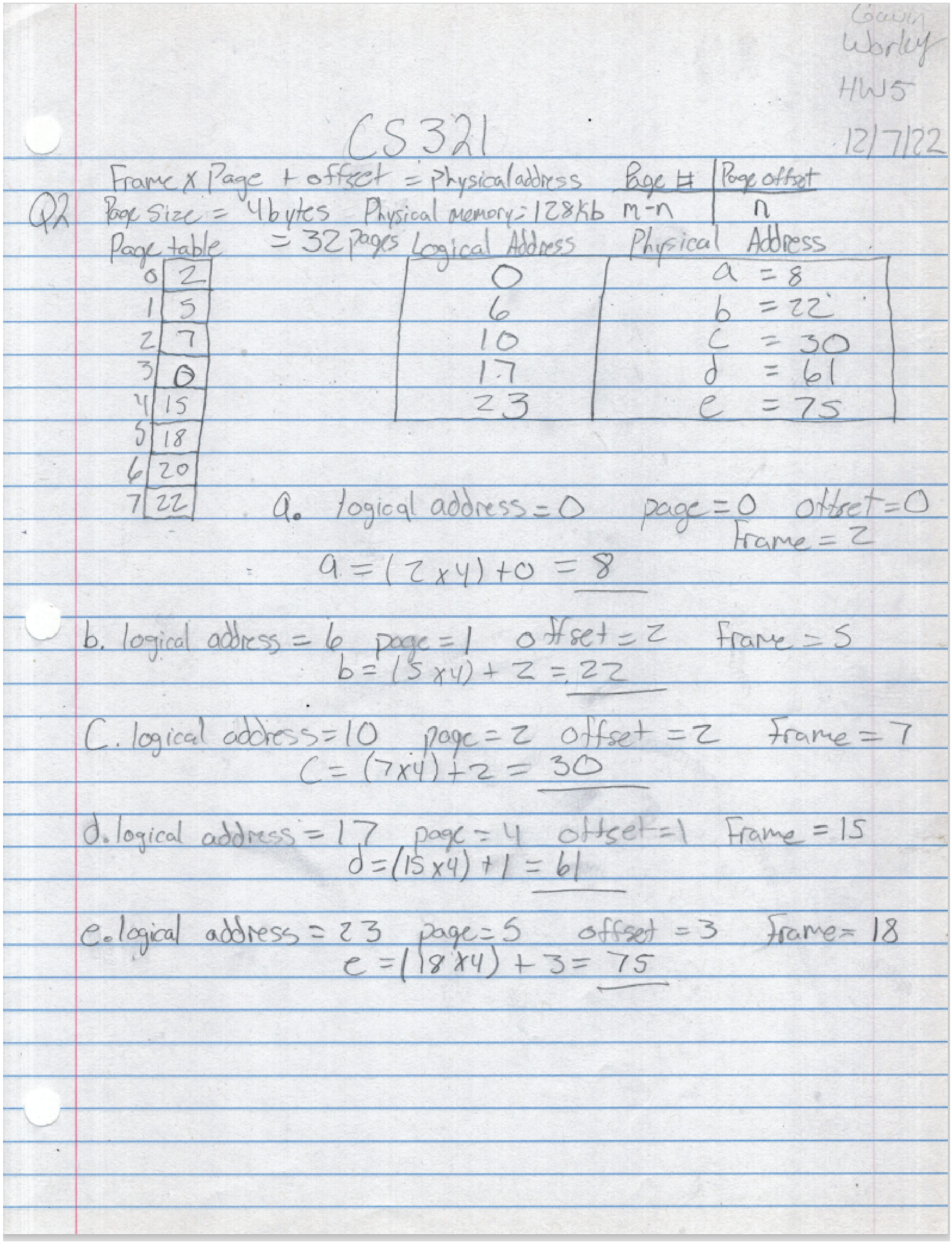
Q1: Some pros and cons of Dynamic Linking vs Static linking, starting with the pros of dynamic linking. The first pro of Dynamic linking is memory usage, while dynamic linking the program only compiles the libraries that are necessary to run the program. Whereas, a con of static linking is the fact that it uses more data by compiling whole libraries when the code may only need one function. A second advantage of dynamic linking is that it can be shared among multiple processors. Lastly, a con of dynamic linking is that it is more susceptible to cyber attacks.

Q2:



Gavin Worley
HW5
12/7/22

CS 321

Frame x Page + offset = Physical address

Page #	Page offset
m-n	n

Q2 Page Size = 4bytes Physical memory = 128kb

Page table = 32 pages

0	2
1	5
2	7
3	0
4	15
5	18
6	20
7	22

Logical Address	Physical Address
0	a = 8
6	b = 22
10	c = 30
17	d = 61
23	e = 75

a. logical address = 0 page = 0 offset = 0 Frame = 2

$a = (2 \times 4) + 0 = \underline{8}$

b. logical address = 6 page = 1 offset = 2 Frame = 5

$b = (5 \times 4) + 2 = \underline{22}$

c. logical address = 10 page = 2 offset = 2 Frame = 7

$c = (7 \times 4) + 2 = \underline{30}$

d. logical address = 17 page = 4 offset = 1 Frame = 15

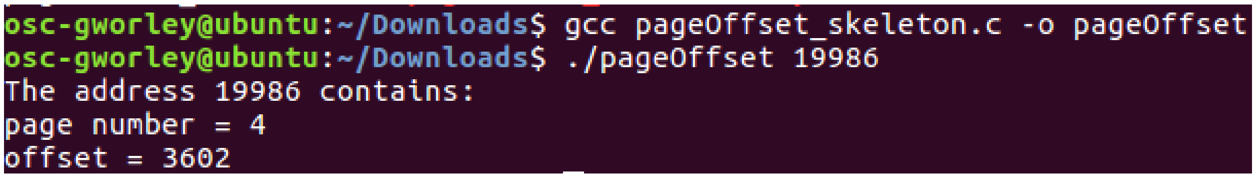
$d = (15 \times 4) + 1 = \underline{61}$

e. logical address = 23 page = 5 offset = 3 Frame = 18

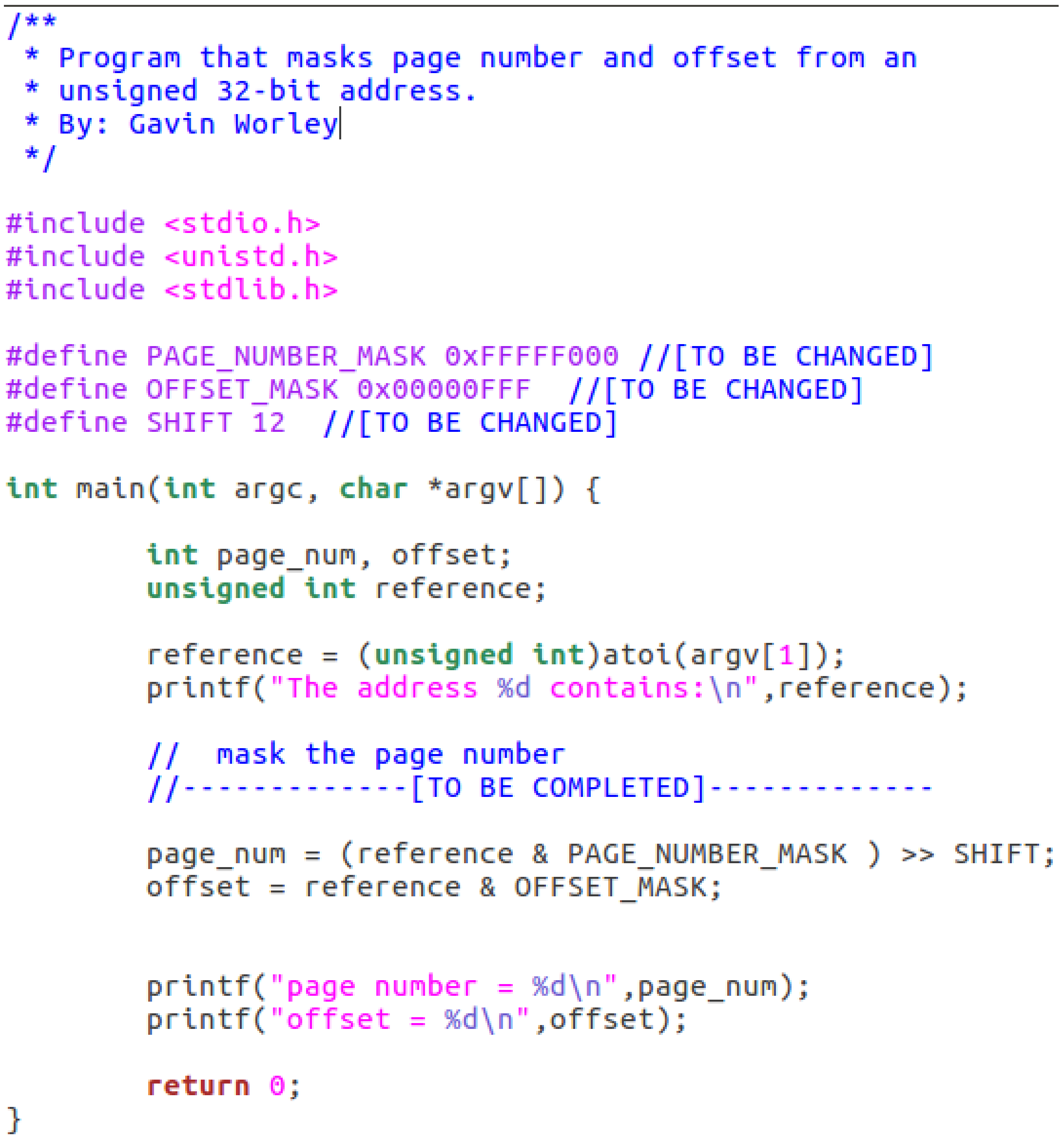
$e = (18 \times 4) + 3 = \underline{75}$

Q3: Paging eliminates external fragmentation by permitting the logical address space of processes to be noncontiguous thus allowing the process to be allocated physical memory wherever it is available. However, paging is still prone to internal fragmentation when a page has not used all of its memory. This leftover memory is the internal fragmentation.

Q4: Github link: https://github.com/gworley-Bradley/CS321/tree/main/Homework5



```
osc-gworley@ubuntu:~/Downloads$ gcc pageOffset_skeleton.c -o pageOffset
osc-gworley@ubuntu:~/Downloads$ ./pageOffset 19986
The address 19986 contains:
page number = 4
offset = 3602
```



```
/**
 * Program that masks page number and offset from an
 * unsigned 32-bit address.
 * By: Gavin Worley
 */

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define PAGE_NUMBER_MASK 0xFFFFF000 //[TO BE CHANGED]
#define OFFSET_MASK 0x00000FFF  //[TO BE CHANGED]
#define SHIFT 12  //[TO BE CHANGED]

int main(int argc, char *argv[]) {

        int page_num, offset;
        unsigned int reference;

        reference = (unsigned int)atoi(argv[1]);
        printf("The address %d contains:\n",reference);

        //  mask the page number
        //-------------[TO BE COMPLETED]-------------

        page_num = (reference & PAGE_NUMBER_MASK ) >> SHIFT;
        offset = reference & OFFSET_MASK;


        printf("page number = %d\n",page_num);
        printf("offset = %d\n",offset);

        return 0;
}
```