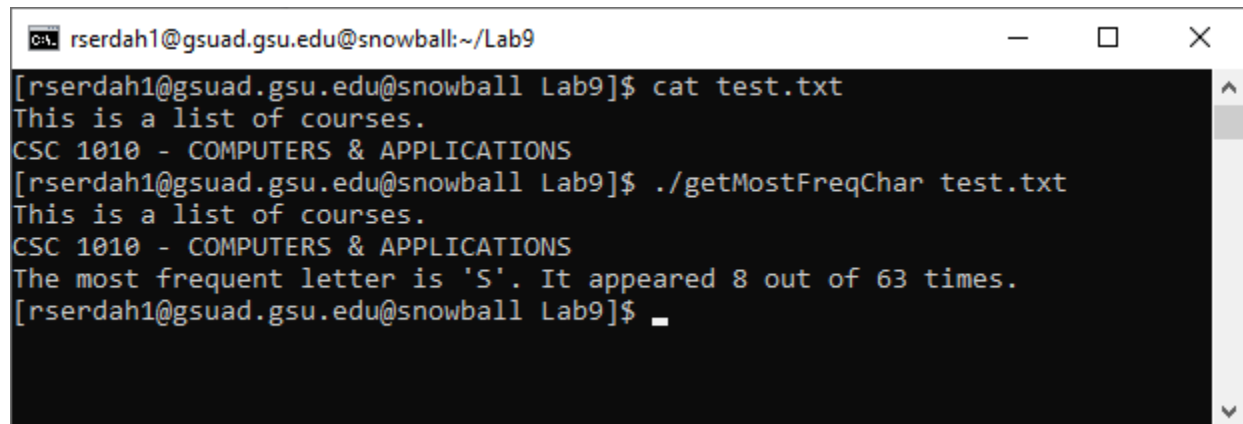


## Lab 9

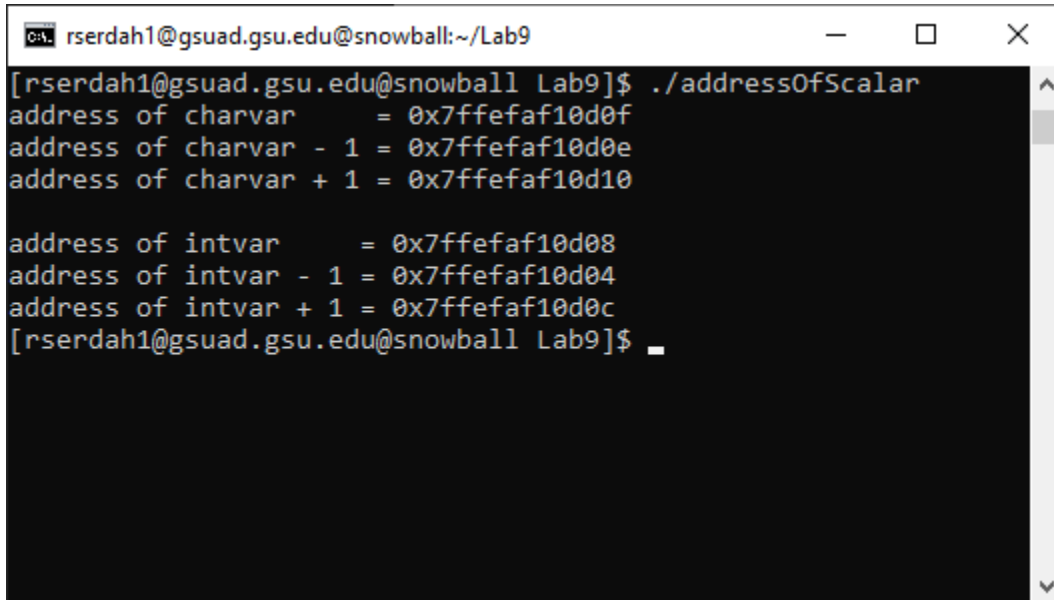
Ramey Serdah

getMostFreqChar.c

A terminal window titled "rserdah1@gsuad.gsu.edu@snowball:~/Lab9" with standard window controls. The terminal shows the following commands and output:

```
[rserdah1@gsuad.gsu.edu@snowball Lab9]$ cat test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS
[rserdah1@gsuad.gsu.edu@snowball Lab9]$ ./getMostFreqChar test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS
The most frequent letter is 'S'. It appeared 8 out of 63 times.
[rserdah1@gsuad.gsu.edu@snowball Lab9]$
```

addressOfScalar.c



```
rserdah1@gsuad.gsu.edu@snowball:~/Lab9
[rserdah1@gsuad.gsu.edu@snowball Lab9]$ ./addressOfScalar
address of charvar      = 0x7ffefaf10d0f
address of charvar - 1 = 0x7ffefaf10d0e
address of charvar + 1 = 0x7ffefaf10d10

address of intvar       = 0x7ffefaf10d08
address of intvar - 1 = 0x7ffefaf10d04
address of intvar + 1 = 0x7ffefaf10d0c
[rserdah1@gsuad.gsu.edu@snowball Lab9]$
```

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    // initialize a char variable, print its address and the next address
    char charvar = '\0';
    printf("address of charvar      = %p\n", (void *)&charvar);
    printf("address of charvar - 1 = %p\n", (void *)&charvar - 1);
    printf("address of charvar + 1 = %p\n\n", (void *)&charvar + 1);
```

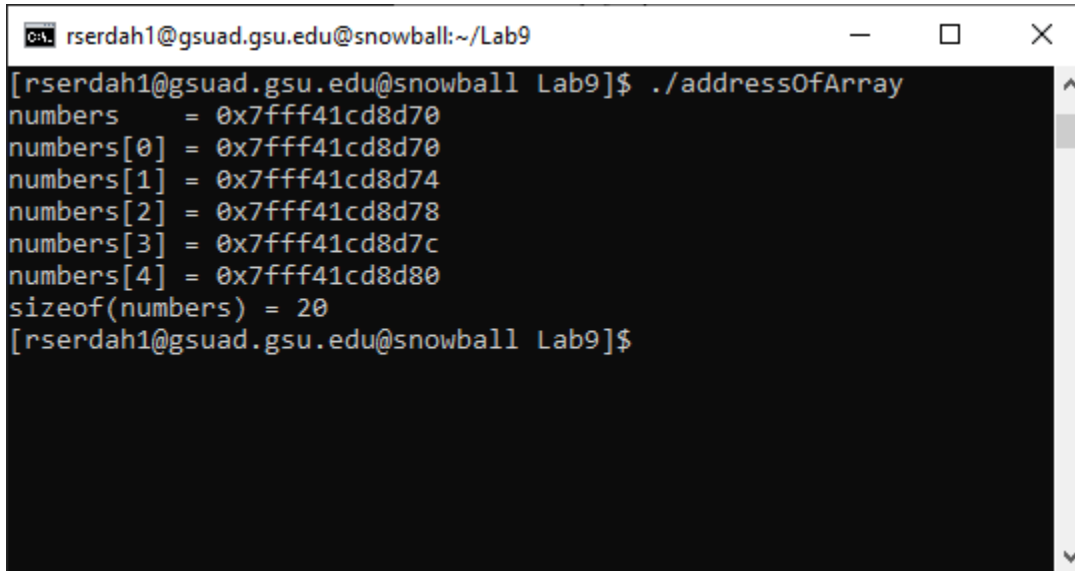
```
    // initialize an int variable, print its address and the next address
    int intvar = 1;
    printf("address of intvar       = %p\n", (void *)&intvar);
    printf("address of intvar - 1 = %p\n", (void *)&intvar - 1);
    printf("address of intvar + 1 = %p\n", (void *)&intvar + 1);
```

```
    return 0;
```

```
}
```

The address of `intvar` is incremented by 4 bytes because integers are 4 bytes each, so the next int after `intvar` would be located 4 bytes after the address of `intvar`. Char is the type that contains only one byte.

addressOfArray.c

A terminal window with a black background and white text. The window title is "rserdah1@gsuad.gsu.edu@snowball:~/Lab9". The prompt is "[rserdah1@gsuad.gsu.edu@snowball Lab9]\$". The command executed is "./addressOfArray". The output shows the memory addresses of an array and its elements. The array itself is at 0x7fff41cd8d70. The first element, numbers[0], is also at 0x7fff41cd8d70. Subsequent elements are at increasing addresses: numbers[1] at 0x7fff41cd8d74, numbers[2] at 0x7fff41cd8d78, numbers[3] at 0x7fff41cd8d7c, and numbers[4] at 0x7fff41cd8d80. Finally, the sizeof(numbers) is printed as 20. The prompt returns to "[rserdah1@gsuad.gsu.edu@snowball Lab9]\$".

```
[rserdah1@gsuad.gsu.edu@snowball Lab9]$ ./addressOfArray
numbers      = 0x7fff41cd8d70
numbers[0]   = 0x7fff41cd8d70
numbers[1]   = 0x7fff41cd8d74
numbers[2]   = 0x7fff41cd8d78
numbers[3]   = 0x7fff41cd8d7c
numbers[4]   = 0x7fff41cd8d80
sizeof(numbers) = 20
[rserdah1@gsuad.gsu.edu@snowball Lab9]$
```

Yes, the address of the array and the address of the first element are the same.

```
printf("length of numbers  = %lu\n", sizeof(numbers) / 4);
```