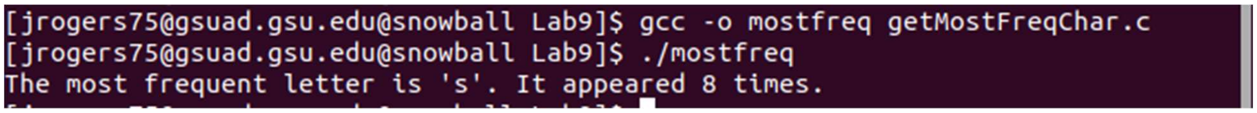
Jamie Lopez

CSC3320

Lab 9 - Due 3/21/21

Part 1

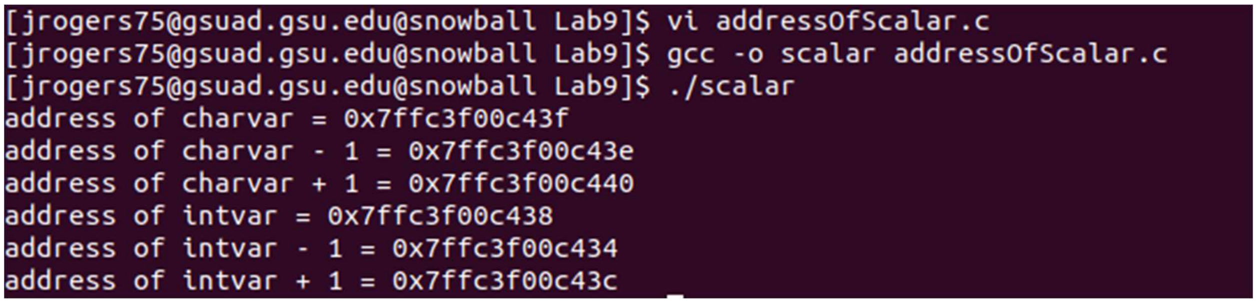
Attach a screenshot of the output from getMostFreqChar.c



```
[jrogers75@gsuad.gsu.edu@snowball Lab9]$ gcc -o mostfreq getMostFreqChar.c
[jrogers75@gsuad.gsu.edu@snowball Lab9]$ ./mostfreq
The most frequent letter is 's'. It appeared 8 times.
```

Part 2

Question 1: Run the C program and attach a screenshot of the output.



```
[jrogers75@gsuad.gsu.edu@snowball Lab9]$ vi addressOfScalar.c
[jrogers75@gsuad.gsu.edu@snowball Lab9]$ gcc -o scalar addressOfScalar.c
[jrogers75@gsuad.gsu.edu@snowball Lab9]$ ./scalar
address of charvar = 0x7ffc3f00c43f
address of charvar - 1 = 0x7ffc3f00c43e
address of charvar + 1 = 0x7ffc3f00c440
address of intvar = 0x7ffc3f00c438
address of intvar - 1 = 0x7ffc3f00c434
address of intvar + 1 = 0x7ffc3f00c43c
```

Question 2: Attach the source code to the answer sheet.

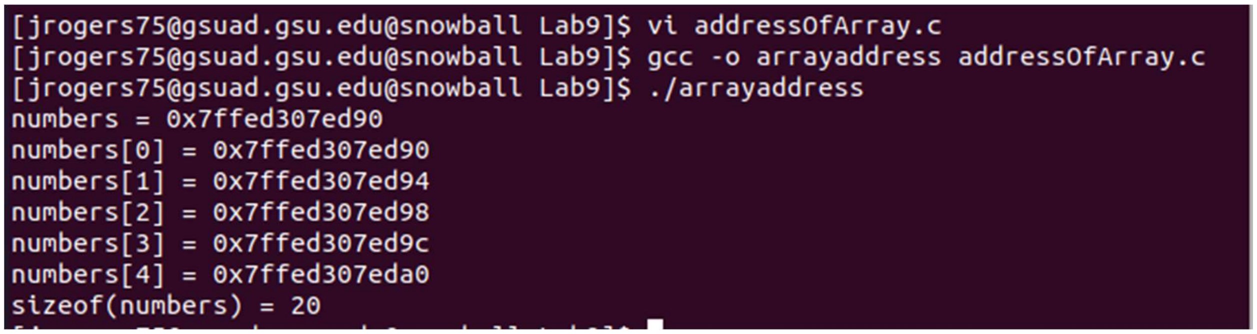
```c
#include <stdio.h>
/* Jamie Lopez
 * CSC3320
 * Lab 9  part 2 */

int main (void)
{
   //initialize a char variable, print its address and 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", (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;

}
```

Question 3: Explain why the address after **intvar** is incremented by 4 bytes instead of 1 byte.

Integers are allotted 4 bytes for storage whereas characters are 1 byte.

__Part 3__

Question 1: Run the C program and attach a screenshot of the output.



```
[jrogers75@gsuad.gsu.edu@snowball Lab9]$ vi addressOfArray.c
[jrogers75@gsuad.gsu.edu@snowball Lab9]$ gcc -o arrayaddress addressOfArray.c
[jrogers75@gsuad.gsu.edu@snowball Lab9]$ ./arrayaddress
numbers = 0x7ffed307ed90
numbers[0] = 0x7ffed307ed90
numbers[1] = 0x7ffed307ed94
numbers[2] = 0x7ffed307ed98
numbers[3] = 0x7ffed307ed9c
numbers[4] = 0x7ffed307eda0
sizeof(numbers) = 20
```

Question 2: Check the address of the array and the address of the first element in the array. Are they the same?

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

0x7ffed307ed90.

Question 3: Write down the statement to print out the length of the array by using **sizeof** operator.

printf(“sizeof(numbers) = %1u\n”, sizeof(numbers));