Vivian Do
Oct 29 2021
CRN 88089
Lab 9 Commands Report

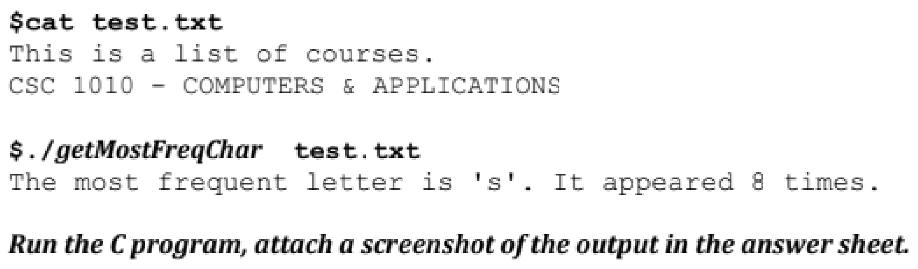
CSC3320 System Level Programming
Lab Assignment 9 - Post-Lab

Due at 11:59 pm on Sunday, March 21, 2021

Purpose: Learn how to use array in C. Understand the basic memory address in C.

Part 1:

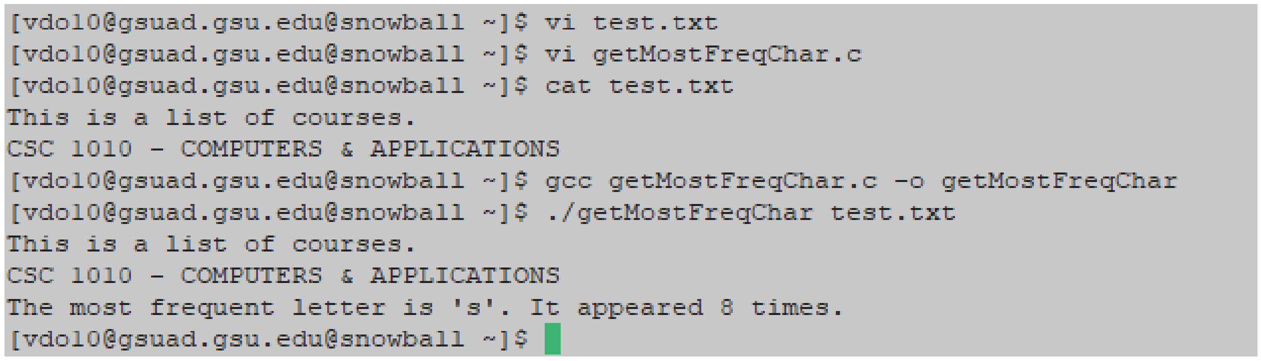
Write a C program named as getMostFreqChar.c that finds the most frequent letter from the input via ignoring the case sensitive and prints out its frequency. For example, sample outputs could be like below



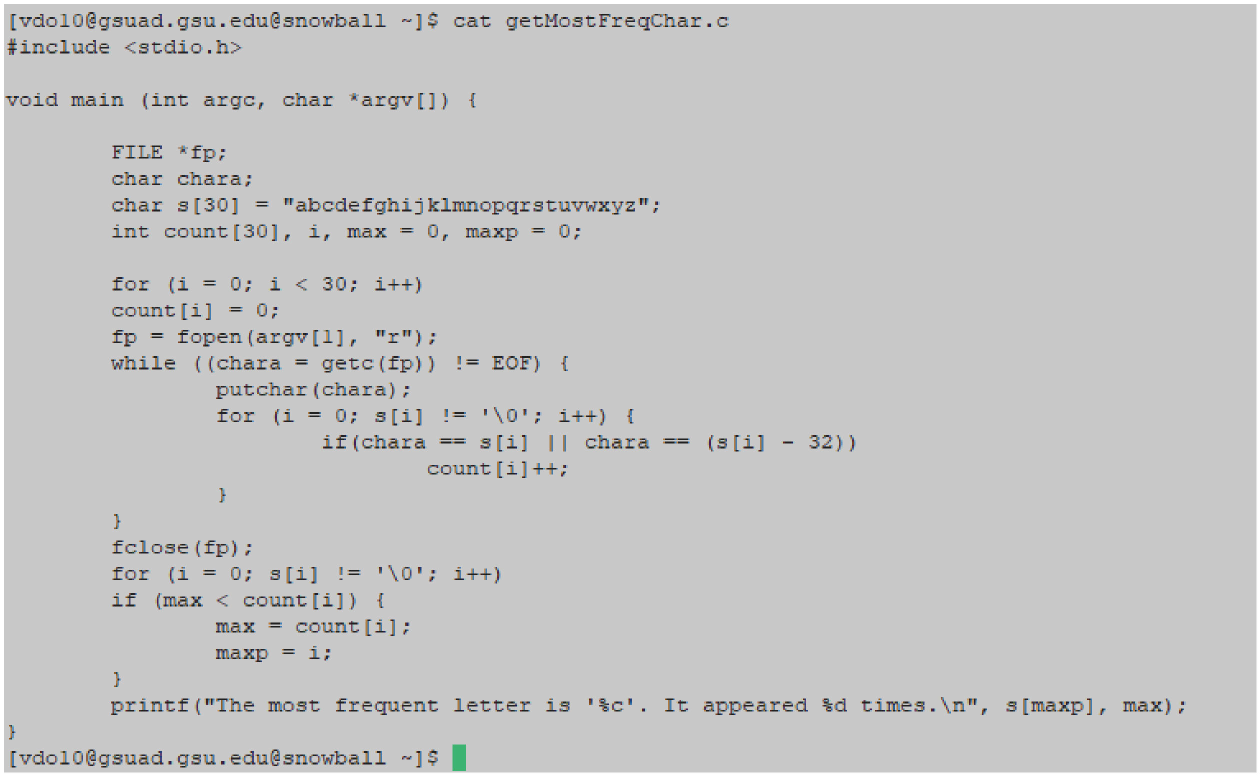
```
$cat test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS

$./getMostFreqChar  test.txt
The most frequent letter is 's'. It appeared 8 times.
```

Run the C program, attach a screenshot of the output in the answer sheet.

My Output/Screenshots:



```
[vdo10@gsuad.gsu.edu@snowball ~]$ vi test.txt
[vdo10@gsuad.gsu.edu@snowball ~]$ vi getMostFreqChar.c
[vdo10@gsuad.gsu.edu@snowball ~]$ cat test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS
[vdo10@gsuad.gsu.edu@snowball ~]$ gcc getMostFreqChar.c -o getMostFreqChar
[vdo10@gsuad.gsu.edu@snowball ~]$ ./getMostFreqChar test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS
The most frequent letter is 's'. It appeared 8 times.
[vdo10@gsuad.gsu.edu@snowball ~]$ 
```



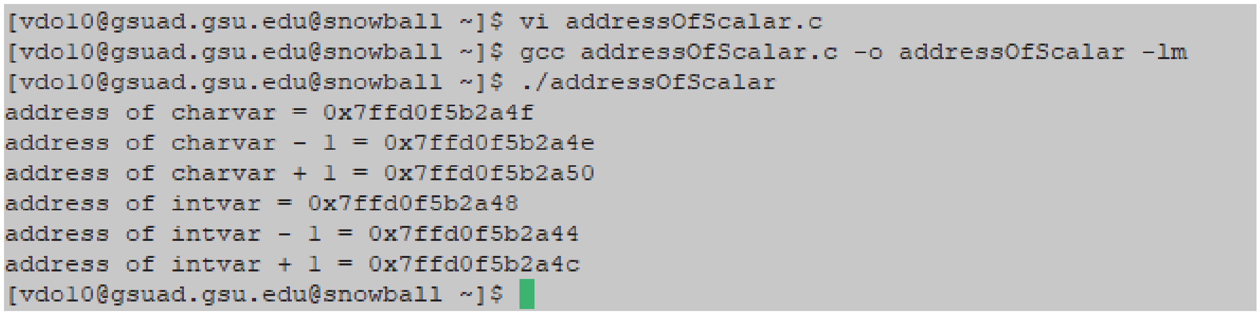
```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat getMostFreqChar.c
#include <stdio.h>

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

        FILE *fp;
        char chara;
        char s[30] = "abcdefghijklmnopqrstuvwxyz";
        int count[30], i, max = 0, maxp = 0;

        for (i = 0; i < 30; i++)
        count[i] = 0;
        fp = fopen(argv[1], "r");
        while ((chara = getc(fp)) != EOF) {
                putchar(chara);
                for (i = 0; s[i] != '\0'; i++) {
                        if(chara == s[i] || chara == (s[i] - 32))
                                count[i]++;
                }
        }
        fclose(fp);
        for (i = 0; s[i] != '\0'; i++)
        if (max < count[i]) {
                max = count[i];
                maxp = i;
        }
        printf("The most frequent letter is '%c'. It appeared %d times.\n", s[maxp], max);
}
[vdol0@gsuad.gsu.edu@snowball ~]$
```

Part 2:

When a variable is stored in memory, it is associated with an address. To obtain the address of a variable, the & operator can be used. For example, &a gets the memory address of variable a. Let's try some examples.
Write a C program addressOfScalar.c by inserting the code below in the main function.

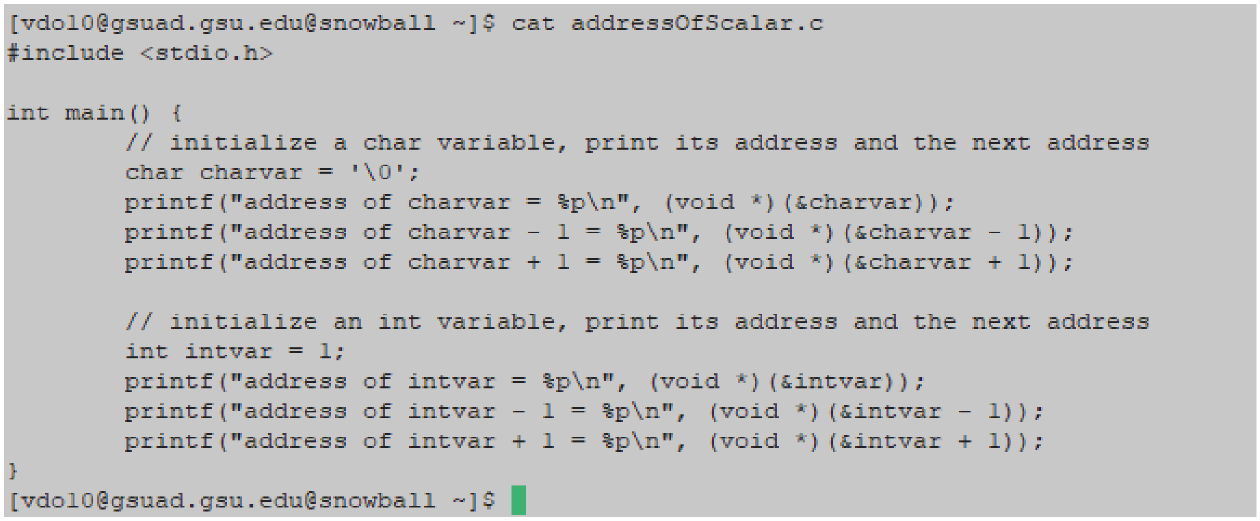
Questions:
1) Run the C program, attach a screenshot of the output in the answer sheet.

My Output/Screenshots:



```
[vdol0@gsuad.gsu.edu@snowball ~]$ vi addressOfScalar.c
[vdol0@gsuad.gsu.edu@snowball ~]$ gcc addressOfScalar.c -o addressOfScalar -lm
[vdol0@gsuad.gsu.edu@snowball ~]$ ./addressOfScalar
address of charvar = 0x7ffd0f5b2a4f
address of charvar - 1 = 0x7ffd0f5b2a4e
address of charvar + 1 = 0x7ffd0f5b2a50
address of intvar = 0x7ffd0f5b2a48
address of intvar - 1 = 0x7ffd0f5b2a44
address of intvar + 1 = 0x7ffd0f5b2a4c
[vdol0@gsuad.gsu.edu@snowball ~]$
```

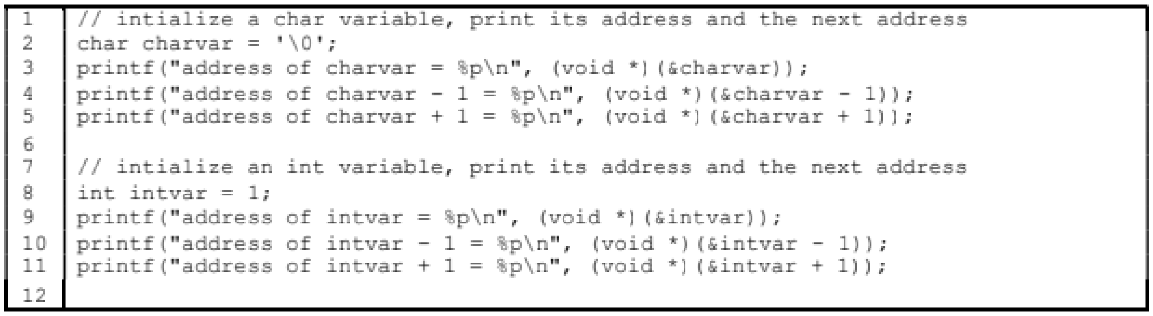
2) Attach the source code in the answer sheet



```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat addressOfScalar.c
#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", (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));
}
[vdol0@gsuad.gsu.edu@snowball ~]$ 
```

2) Then explain why the address after intvar is incremented by 4 bytes instead of 1 byte.

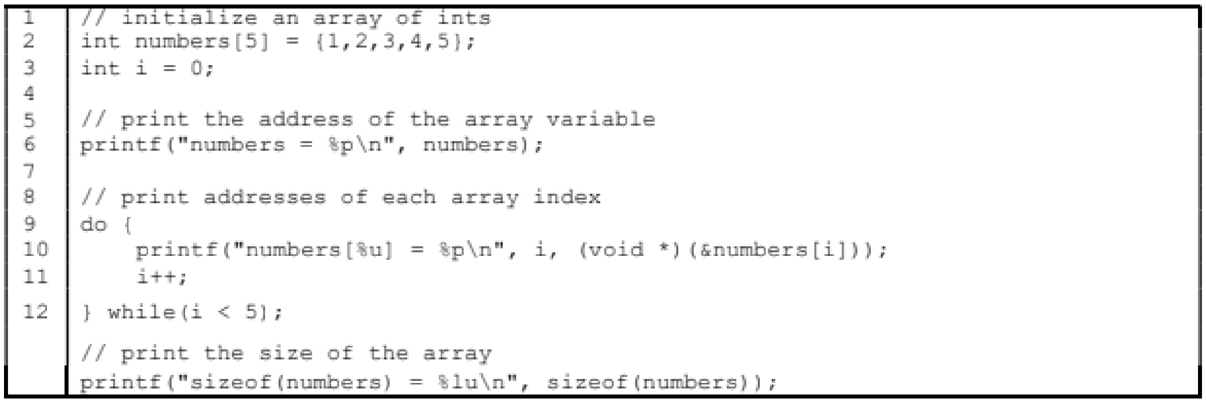
Each int variable takes up 4 bytes of memory. The next integer is incremented by 4 bytes of data. In this case, *intvar* - *1* is the element 4 bytes before *intvar*'s address, and *intvar* + *1* is the element 4 bytes after *intvar*'s address.



```
// intialize 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", (void *)(&charvar + 1));

// intialize 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));
```

Part 3:

Write a C program addressOfArray.c by inserting the code below in the main function.

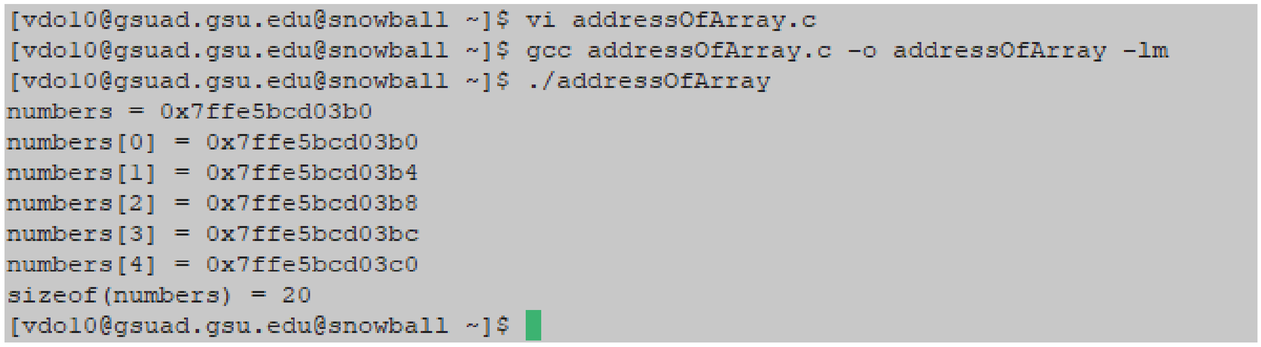


```
// initialize an array of ints
int numbers[5] = {1,2,3,4,5};
int i = 0;

// print the address of the array variable
printf("numbers = %p\n", numbers);

// print addresses of each array index
do {
    printf("numbers[%u] = %p\n", i, (void *)(&numbers[i]));
    i++;
} while(i < 5);

// print the size of the array
printf("sizeof(numbers) = %lu\n", sizeof(numbers));
```

Questions:

1) Run the C program, attach a screenshot of the output in the answer sheet.

My Output/Screenshots:



```
[vdo10@gsuad.gsu.edu@snowball ~]$ vi addressOfArray.c
[vdo10@gsuad.gsu.edu@snowball ~]$ gcc addressOfArray.c -o addressOfArray -lm
[vdo10@gsuad.gsu.edu@snowball ~]$ ./addressOfArray
numbers = 0x7ffe5bcd03b0
numbers[0] = 0x7ffe5bcd03b0
numbers[1] = 0x7ffe5bcd03b4
numbers[2] = 0x7ffe5bcd03b8
numbers[3] = 0x7ffe5bcd03bc
numbers[4] = 0x7ffe5bcd03c0
sizeof(numbers) = 20
[vdo10@gsuad.gsu.edu@snowball ~]$ 
```

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

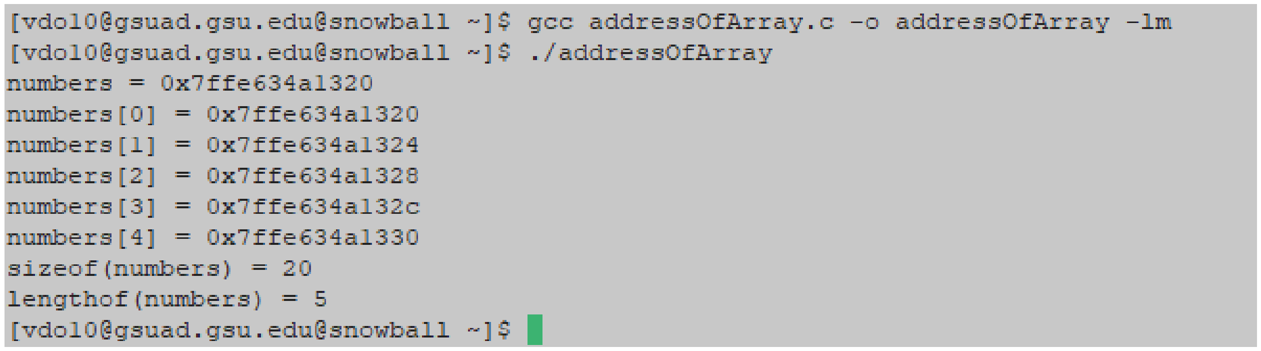
Yes, the address of the array and the address of the first element are the same!

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

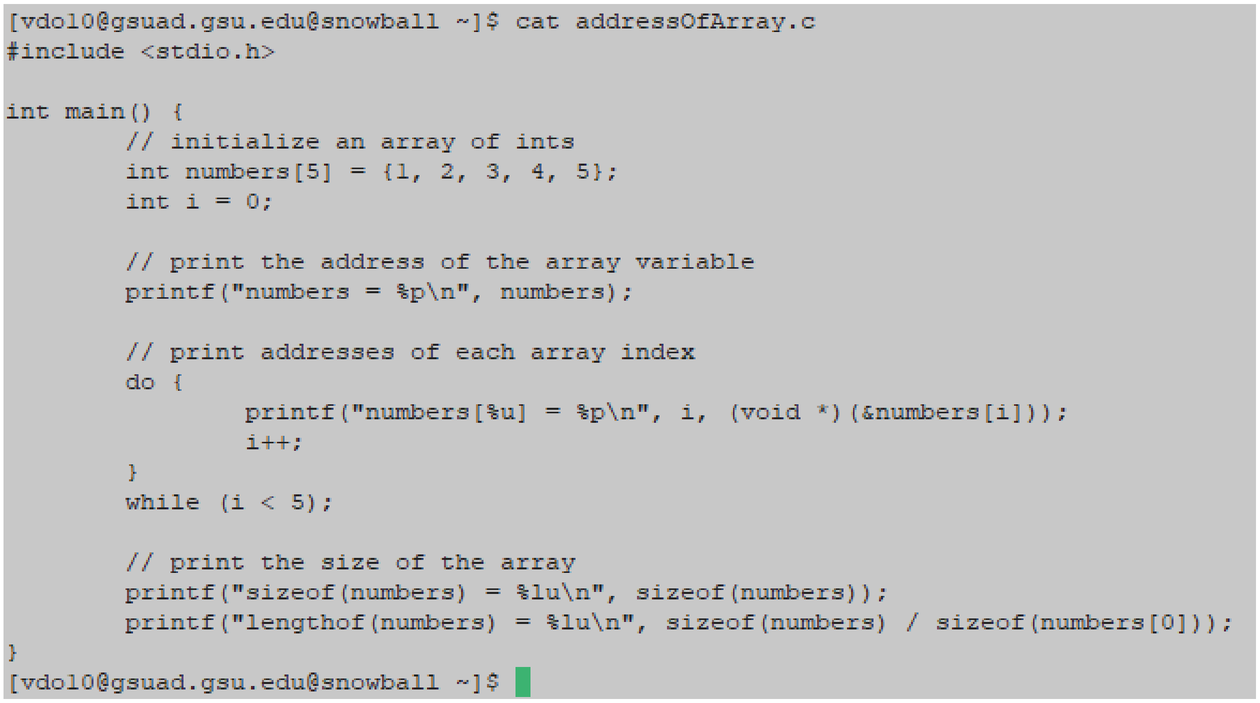
printf("lengthof(numbers) = %lu\n", sizeof(numbers) / sizeof(numbers[0]));

- the length of the array can be found by dividing *sizeof* to the size of any element in the array

Output/Screenshots:



```
[vdo10@gsuad.gsu.edu@snowball ~]$ gcc addressOfArray.c -o addressOfArray -lm
[vdo10@gsuad.gsu.edu@snowball ~]$ ./addressOfArray
numbers = 0x7ffe634a1320
numbers[0] = 0x7ffe634a1320
numbers[1] = 0x7ffe634a1324
numbers[2] = 0x7ffe634a1328
numbers[3] = 0x7ffe634a132c
numbers[4] = 0x7ffe634a1330
sizeof(numbers) = 20
lengthof(numbers) = 5
[vdo10@gsuad.gsu.edu@snowball ~]$ 
```



```
[vdo10@gsuad.gsu.edu@snowball ~]$ cat addressOfArray.c
#include <stdio.h>

int main() {
        // initialize an array of ints
        int numbers[5] = {1, 2, 3, 4, 5};
        int i = 0;

        // print the address of the array variable
        printf("numbers = %p\n", numbers);

        // print addresses of each array index
        do {
                printf("numbers[%u] = %p\n", i, (void *)(&numbers[i]));
                i++;
        }
        while (i < 5);

        // print the size of the array
        printf("sizeof(numbers) = %lu\n", sizeof(numbers));
        printf("lengthof(numbers) = %lu\n", sizeof(numbers) / sizeof(numbers[0]));
}
[vdo10@gsuad.gsu.edu@snowball ~]$
```

Submission:

- Upload an electronic copy (pdf) of your answer sheet to the folder named “Lab 9” in Google Classroom
- Please add the lab assignment number and your name at the top of your answer sheet.
- Upload the C files getMostFreqChar.c, addressOfArray.c and addressOfScalar.c to the folder named named “Lab 9” in Google Classroom
- Name your file in the format of Lab9_ FirstnameLastname (e.g Lab9_FilRondel.pdf)