

Lab 9

Hawa Sylla

Part 1

Run the C program, attach a screenshot of the output below:

```
[hsylla2@gsuad.gsu.edu@snowball ~]$ ./getMostFreqChar
Input the String :I am programming a C Program for my System-Level Course.
The most frequent letter is 'm' . It appeared 6 times. [hsylla2@gsuad.gsu.edu@snowball ~]$
```

Part 2

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

```
[hsylla2@gsuad.gsu.edu@snowball ~]$ vi addressOfScalar.c
[hsylla2@gsuad.gsu.edu@snowball ~]$ gcc addressOfScalar.c -o addressOfScalar.c
[hsylla2@gsuad.gsu.edu@snowball ~]$ ./addressOfScalar.c
address of charvar = 0x7ffdde99cc1f
address of charvar - 1 = 0x7ffdde99cc1e
address of charvar + 1 = 0x7ffdde99cc20
address of intvar = 0x7ffdde99cc18
address of intvar - 1 = 0x7ffdde99cc14
address of intvar + 1 = 0x7ffdde99cc1c
[hsylla2@gsuad.gsu.edu@snowball ~]$
```

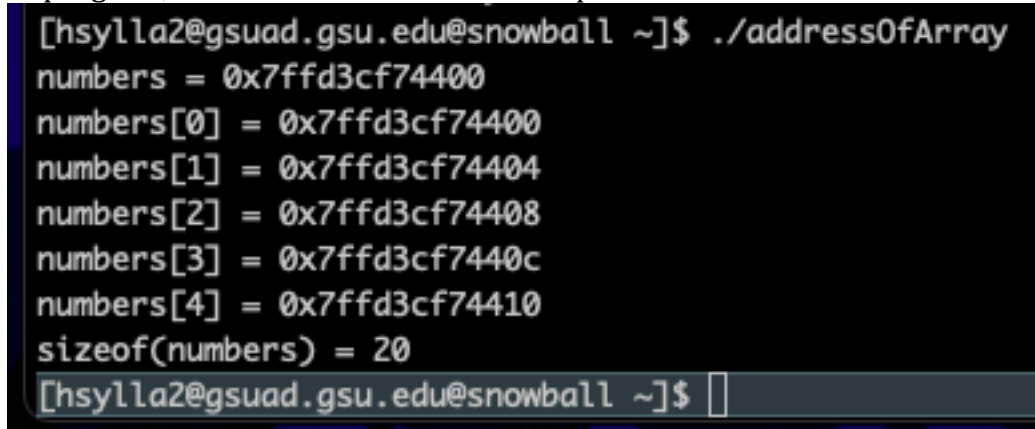
a.
2) Attach the source code in the answer sheet

1. #include <stdio.h>
- 2.
3. int main()
4. {
- 5.
6. char charvar = '\0';
7. printf("address of charvar = %p\n", (void *) (&charvar));
8. printf("address of charvar - 1 = %p\n", (void *) (&charvar - 1));
9. printf("address of charvar + 1 = %p\n", (void *) (&charvar + 1));
- 10.
11. int intvar = 1;
12. printf("address of intvar = %p\n", (void *) (&intvar));
13. printf("address of intvar - 1 = %p\n", (void *) (&intvar - 1));
14. printf("address of intvar + 1 = %p\n", (void *) (&intvar + 1));

- 15.
 16. }
- 3) Then explain why the address after **intvar** is incremented by 4 bytes instead of 1 byte.
- a. In C, int takes 4 bytes of memory. Any arithmetic on int address will be received as 4 bytes of data.
 - i. Therefore, **intvar-1** is equal to location 4 bytes before **intvar** address, and **intvar+1** is location 4 bytes after **intvar** address.

Part 3

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



```
[hsylla2@gsuad.gsu.edu@snowball ~]$ ./addressOfArray
numbers = 0x7ffd3cf74400
numbers[0] = 0x7ffd3cf74400
numbers[1] = 0x7ffd3cf74404
numbers[2] = 0x7ffd3cf74408
numbers[3] = 0x7ffd3cf7440c
numbers[4] = 0x7ffd3cf74410
sizeof(numbers) = 20
[hsylla2@gsuad.gsu.edu@snowball ~]$
```

- a.
- 2) Check the address of the array and the address of the first element in the array. Are they the same?
 - a. Yes, they are.
- 3) Write down the statement to print out the length of the array by using sizeof operator.
 1. `n=sizeof(numbers)/sizeof(int);`
 2. `printf("Number of elements are: %d\n",n);`