

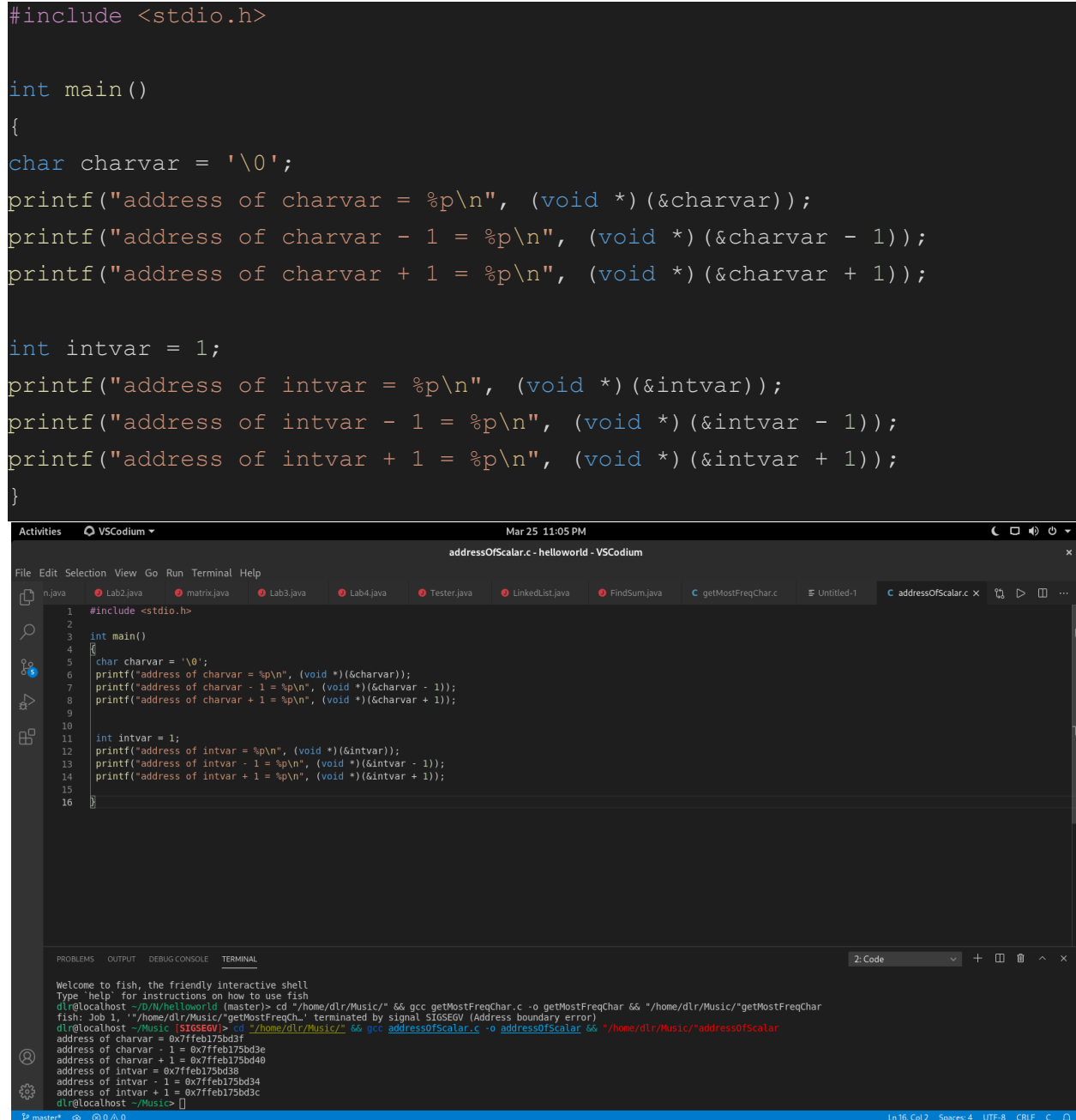
Name: David Louis Roy II  
Lab Number: 9

Part 2:

```
#include <stdio.h>

int main()
{
    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));

    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));
}
```



The screenshot shows a VS Code editor window with a C program named `addressOfScalar.c`. The code defines a `char` variable `charvar` and an `int` variable `intvar`, both with memory addresses printed before and after incrementing/decrementing by 1. The terminal output shows the program's execution, including a `SIGSEGV` error when attempting to access `&charvar - 1`, and the final memory addresses for both variables and their neighbors.

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Welcome to fish, the friendly interactive shell
Type 'help' for instructions on how to use fish
dlr@localhost ~/B/N/helloworld (master)> cd ~/home/dlr/Music/ && gcc getMostFreqChar.c -o getMostFreqChar && ~/home/dlr/Music/"getMostFreqChar
fish: Job 1. '~/home/dlr/Music/"getMostFreqChar.' terminated by signal SIGSEGV (Address boundary error)
dlr@localhost ~/Music (SIGSEGV)> cd ~/home/dlr/Music/ && gcc addressOfScalar.c -o addressOfScalar && ~/home/dlr/Music/"addressOfScalar
address of charvar = 0x7ffeb175bd3e
address of charvar - 1 = 0x7ffeb175bd3e
address of charvar + 1 = 0x7ffeb175bd40
address of intvar = 0x7ffeb175bd38
address of intvar - 1 = 0x7ffeb175bd34
address of intvar + 1 = 0x7ffeb175bd3c
dlr@localhost ~/Music>
```

Ln 16, Col 2 Spaces: 4 UTF-8 CRLF

The address after `intvar` is incremented by 4 bytes instead of 1 because an `int` contains 4 bytes of data.

### Part 3:

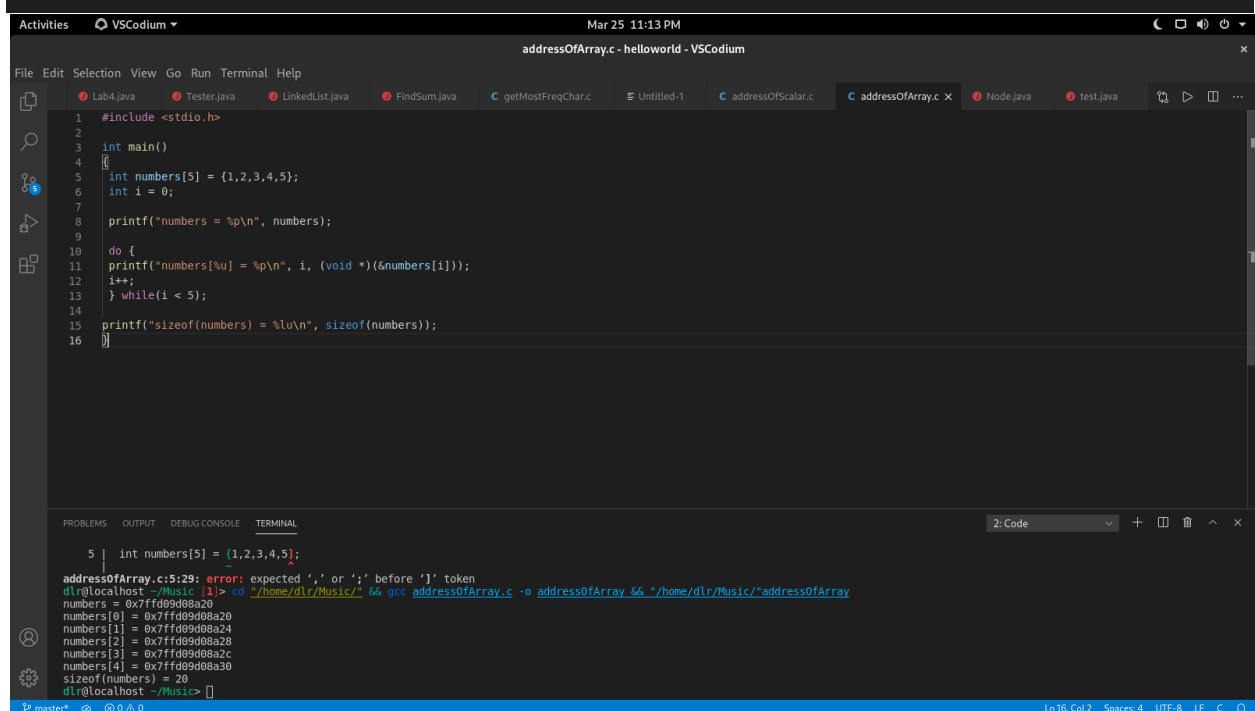
```
#include <stdio.h>

int main()
{
    int numbers[5] = {1,2,3,4,5};
    int i = 0;

    printf("numbers = %p\n", numbers);

    do {
        printf("numbers[%u] = %p\n", i, (void *)(&numbers[i]));
        i++;
    } while(i < 5);

    printf("sizeof(numbers) = %lu\n", sizeof(numbers));
}
```



The screenshot shows the VS Code interface with a C program in the editor and its output in the terminal. The program defines an array of 5 integers and prints its address, the address of each element, and its total size. The terminal output shows that the array address and the address of the first element are identical, and the size is 20 bytes.

```
5 | int numbers[5] = {1,2,3,4,5};
addressOfArray.c:5:29: error: expected ',' or ';' before ']' token
dhr@localhost ~/Music [1]> cd "/home/dhr/Music/" && gcc addressOfArray.c -o addressOfArray && "/home/dhr/Music/addressOfArray
numbers = 0x7ffdb09d08a20
numbers[0] = 0x7ffdb09d08a20
numbers[1] = 0x7ffdb09d08a24
numbers[2] = 0x7ffdb09d08a28
numbers[3] = 0x7ffdb09d08a2c
numbers[4] = 0x7ffdb09d08a30
sizeof(numbers) = 20
dhr@localhost ~/Music>
```

The address of the array and the address of the first element of the array are the same, because the base address is based on the address of the first element.

```
sizeof(numbers)
```

Is the statement that gives the size of an element of the array