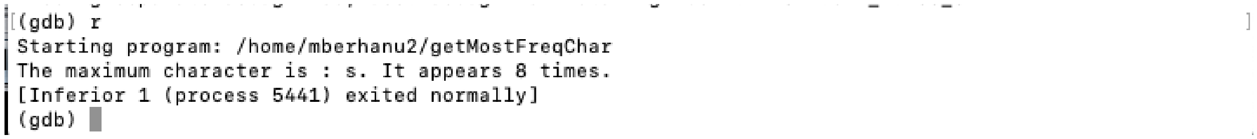
Mahi Berhanu
Lab 9 Post Lab

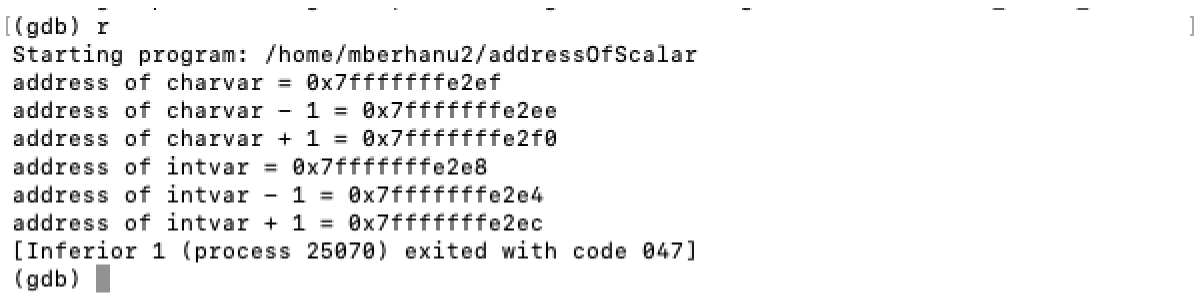
Part 1:

Output:



```
(gdb) r
Starting program: /home/mberhanu2/getMostFreqChar
The maximum character is : s. It appears 8 times.
[Inferior 1 (process 5441) exited normally]
(gdb)
```

Part 2:

1. 

```
(gdb) r
Starting program: /home/mberhanu2/addressOfScalar
address of charvar = 0x7fffffffe2ef
address of charvar - 1 = 0x7fffffffe2ee
address of charvar + 1 = 0x7fffffffe2f0
address of intvar = 0x7fffffffe2e8
address of intvar - 1 = 0x7fffffffe2e4
address of intvar + 1 = 0x7fffffffe2ec
[Inferior 1 (process 25070) exited with code 047]
(gdb)
```

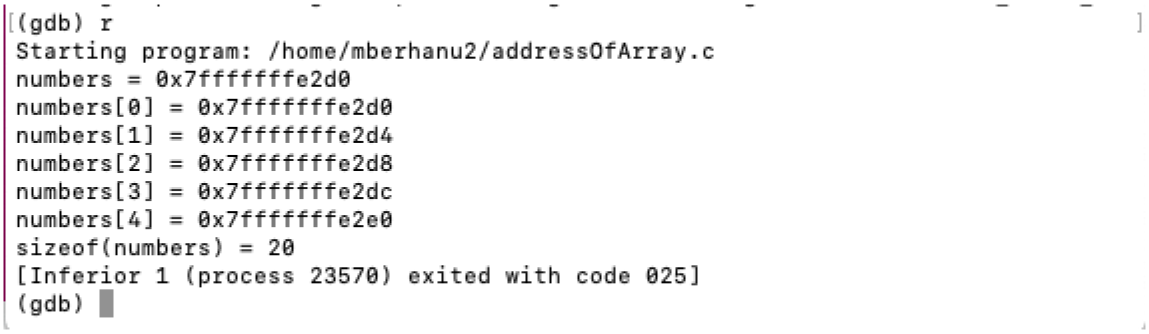
2.

```
#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));

//initalize 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));
}
```

3. This is because whenever we are incrementing a pointer it increments as the size of its own data type. Since the size of an int is 4 bytes, it increases by 4 bytes. If the type was a char, it would have increased by 8 bytes instead of 4 bytes.

Part 3:

1. 

```
(gdb) r
Starting program: /home/mberhanu2/addressOfArray.c
numbers = 0x7fffffffe2d0
numbers[0] = 0x7fffffffe2d0
numbers[1] = 0x7fffffffe2d4
numbers[2] = 0x7fffffffe2d8
numbers[3] = 0x7fffffffe2dc
numbers[4] = 0x7fffffffe2e0
sizeof(numbers) = 20
[Inferior 1 (process 23570) exited with code 025]
(gdb)
```

2. Yes, the address of the array and the address of the first element in the array is the same because the way array works is that it has elements that are of the same type and they are all stored at a contiguous memory location.

3.
   ```
   int n = sizeof(numbers) / sizeof(int);
   printf(“The length of the array is: %d”, n);
   ```