Part 1

```
The Most frequent letter is 's'. It appeared 8 times.
[[ttahir1@gsuad.gsu.edu@snowball Lab9]$ cat Test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS
[[ttahir1@gsuad.gsu.edu@snowball Lab9]$ ./getMostFreqChar Test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS

The Most frequent letter is 's'. It appeared 8 times.
```

Part 2

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

```
[ttahir1@gsuad.gsu.edu@snowball Lab9]$ vi addressOfScalar.c
[ttahir1@gsuad.gsu.edu@snowball Lab9]$ gcc -o addressOfScalar addressOfScalar.c
[ttahir1@gsuad.gsu.edu@snowball Lab9]$ ls
addressOfScalar  addressOfScalar.c  getMostFreqChar.c  test.txt
[ttahir1@gsuad.gsu.edu@snowball Lab9]$ ./addressOfScalar
address of charvar = 0x7ffd06ff1d6f
address of charvar - 1 = 0x7ffd06ff1d6e
address of charvar + 1 = 0x7ffd06ff1d70
address of intvar = 0x7ffd06ff1d68
address of intvar - 1 = 0x7ffd06ff1d64
address of intvar + 1 = 0x7ffd06ff1d6c
```

2. Attach the source code in the answer sheet

```c
#include <stdio.h>
// intialize a char variable, print its address and the next address
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));

// 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));
return 0;
}
```

3. Then explain why the address after intvar is incremented by 4 bytes instead of 1 byte.
    a. The address of intvar is incremented by 4 bytes because the variable intvar is declared as an integer and in C int data types take 4 bytes of memory. For intvar to be incremented by 1 byte it would have to be declared as a char data type.

Part 3

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

```
[[ttahir1@gsuad.gsu.edu@snowball Lab9]$ vi addressOfArray.c
[[ttahir1@gsuad.gsu.edu@snowball Lab9]$ gcc -o addressOfArray addressOfArray.c
[[ttahir1@gsuad.gsu.edu@snowball Lab9]$ ./addressOfArray
numbers = 0x7ffd1def5190
numbers[0] = 0x7ffd1def5190
numbers[1] = 0x7ffd1def5194
numbers[2] = 0x7ffd1def5198
numbers[3] = 0x7ffd1def519c
numbers[4] = 0x7ffd1def51a0
sizeof(numbers) = 20
```

```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("length = %lu\n", sizeof(numbers)/sizeof(numbers[0]));
return 0;
```

2. Check the address of the array and the address of the first element in the array. Are they the same?
    a. Yes the address of the array and the first element in the array are the same.
3. Write down the statement to print out the length of the array by using sizeof operator.
    a. printf("length = %lu\n", sizeof(numbers)/sizeof(numbers[0]));