CSC3320 System Level Programming
Lab Assignment 9 - Post-Lab
Rafid Shaon

Part 1:

```
[rshaon1@gsuad.gsu.edu@snowball ~]$ cat test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS
 [rshaon1@gsuad.gsu.edu@snowball ~]$ gcc getMostFreqChar.c -o getMostFreqChar
 [rshaon1@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.
[rshaon1@gsuad.gsu.edu@snowball ~]$
```

```
[rshaon1@gsuad.gsu.edu@snowball ~]$ cat getMostFreqChar.c
#include <stdio.h>
void main(int argc, char *argv[])
{
        FILE *fp;
        char ch;
        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((ch=getc(fp))!=EOF)
        {
                putchar(ch);
                for(i=0;s[i]!='\0';i++)
                {
                        if(ch==s[i] || ch==(s[i]-32))
                        count[i]++;
                }
        }
        fclose(fp);
        for(i=0;s[i]!='\0';i++)
        if(max<count[i])
        {
                max=count[i];
                maxp=i;
        }
        printf("\nThe most frequent letter is '%c'. It appeared %d times.\n",s[maxp],max);
}
[rshaon1@gsuad.gsu.edu@snowball ~]$
```

Part 2:

Questions:

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

```
[rshaon1@gsuad.gsu.edu@snowball ~]$ gcc addressOfScalar.c -o addressOfScalar -lm
[[rshaon1@gsuad.gsu.edu@snowball ~]$ ./addressOfScalar
 address of charvar = 0x7fff441284cf
[address of charvar - 1 = 0x7fff441284ce
 address of charvar + 1 = 0x7fff441284d0
[address of intvar = 0x7fff441284c8
 address of intvar - 1 = 0x7fff441284c4
 address of intvar + 1 = 0x7fff441284cc
 [rshaon1@gsuad.gsu.edu@snowball ~]$
```

2) Attach the source code in the answer sheet.

```
[[rshaon1@gsuad.gsu.edu@snowball ~]$ cat addressOfScalar.c
[#include<stdio.h>
[int main()
 {
 // 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));
 }
```

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

In C programming language, an int variable takes 4 bytes of memory. So any arithmetic on integer address, always considers it as 4 bytes of data. So intvar-1 refers to a location 4 bytes before intvar's address and intvar+1 refers to 4 bytes after intvar's address.

Part 3:

Questions:

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

```
[rshaon1@gsuad.gsu.edu@snowball ~]$ gcc addressOfArray.c -o addressOfArray -lm
[rshaon1@gsuad.gsu.edu@snowball ~]$ ./addressOfArray
numbers = 0x7ffed40a7990
numbers[0] = 0x7ffed40a7990
numbers[1] = 0x7ffed40a7994
numbers[2] = 0x7ffed40a7998
numbers[3] = 0x7ffed40a799c
numbers[4] = 0x7ffed40a79a0
sizeof(numbers) = 20
[rshaon1@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, we can say that both the address of the array and the address of the first element of the array are the same.

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

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

Array length can be determined by size of the divided by size of any element in the array.

```
[rshaon1@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("length(numbers)= %lu\n", sizeof(numbers)/sizeof(numbers[0]));
}
[rshaon1@gsuad.gsu.edu@snowball ~]$
```

```
[rshaon1@gsuad.gsu.edu@snowball ~]$ gcc addressOfArray.c -o addressOfArray -lm
[rshaon1@gsuad.gsu.edu@snowball ~]$ ./addressOfArray
numbers = 0x7ffd86d0e560
numbers[0] = 0x7ffd86d0e560
numbers[1] = 0x7ffd86d0e564
numbers[2] = 0x7ffd86d0e568
numbers[3] = 0x7ffd86d0e56c
numbers[4] = 0x7ffd86d0e570
sizeof(numbers) = 20
length(numbers)= 5
[rshaon1@gsuad.gsu.edu@snowball ~]$
```