CSC3320 System Level Programming
Lab Assignment 9 - Post-Lab

Due at 11:59 pm on Sunday, March 21, 2021

Purpose: Learn how to use array in C. Understand the basic memory address in C.

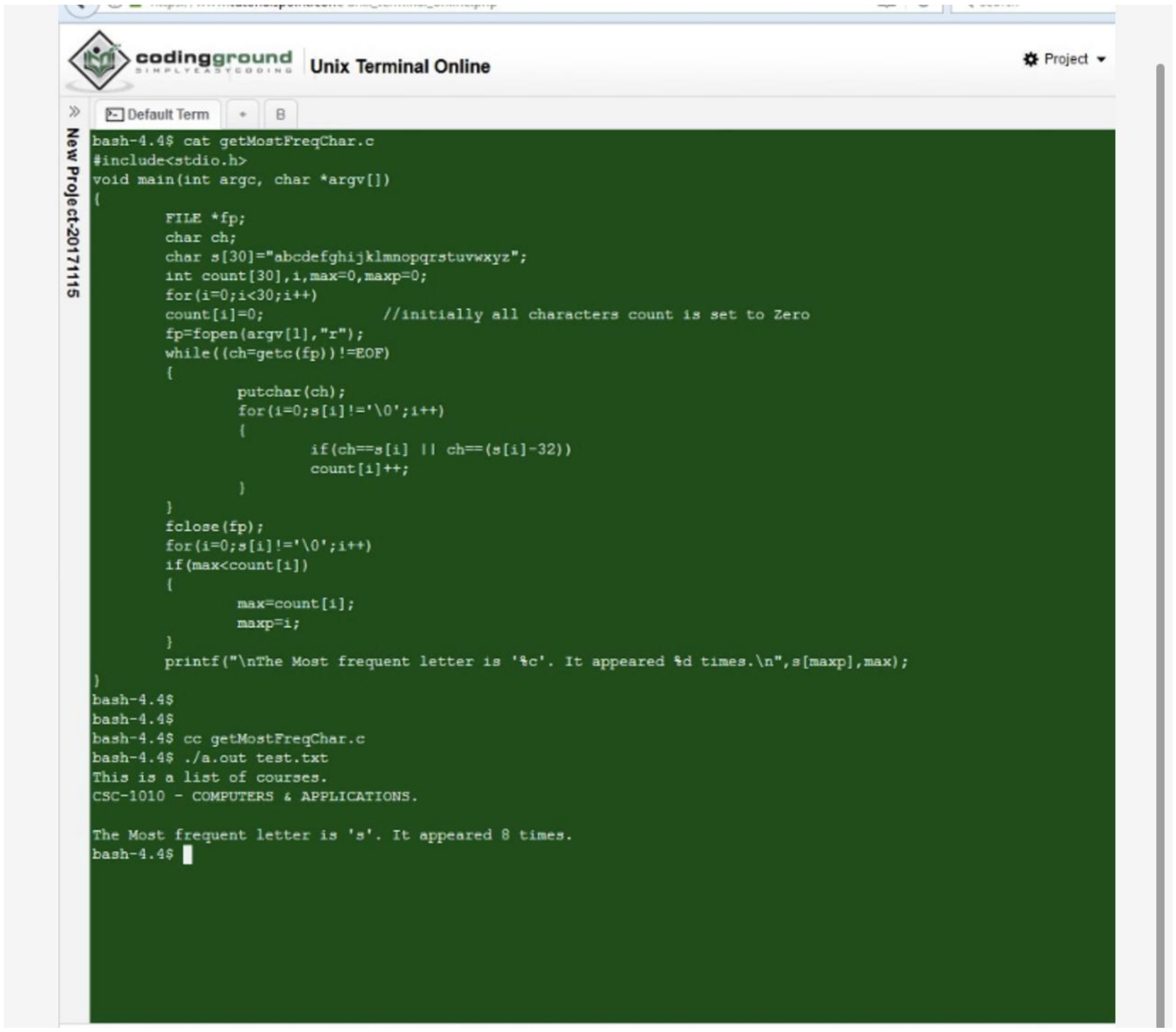
Part 1:

Write a C program named as getMostFreqChar.c that finds the most frequent letter from the input via ignoring the case sensitive and prints out its frequency. For example, sample outputs could be like below

```
$cat test.txt
This is a list of courses.
CSC 1010 - COMPUTERS & APPLICATIONS

$./getMostFreqChar test.txt
The most frequent letter is 's'. It appeared 8 times.
```

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



```
bash-4.4$ 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;             //initially all characters count is set to Zero
        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);
}
bash-4.4$
bash-4.4$
bash-4.4$ cc getMostFreqChar.c
bash-4.4$ ./a.out test.txt
This is a list of courses.
CSC-1010 - COMPUTERS & APPLICATIONS.

The Most frequent letter is 's'. It appeared 8 times.
bash-4.4$ 
```

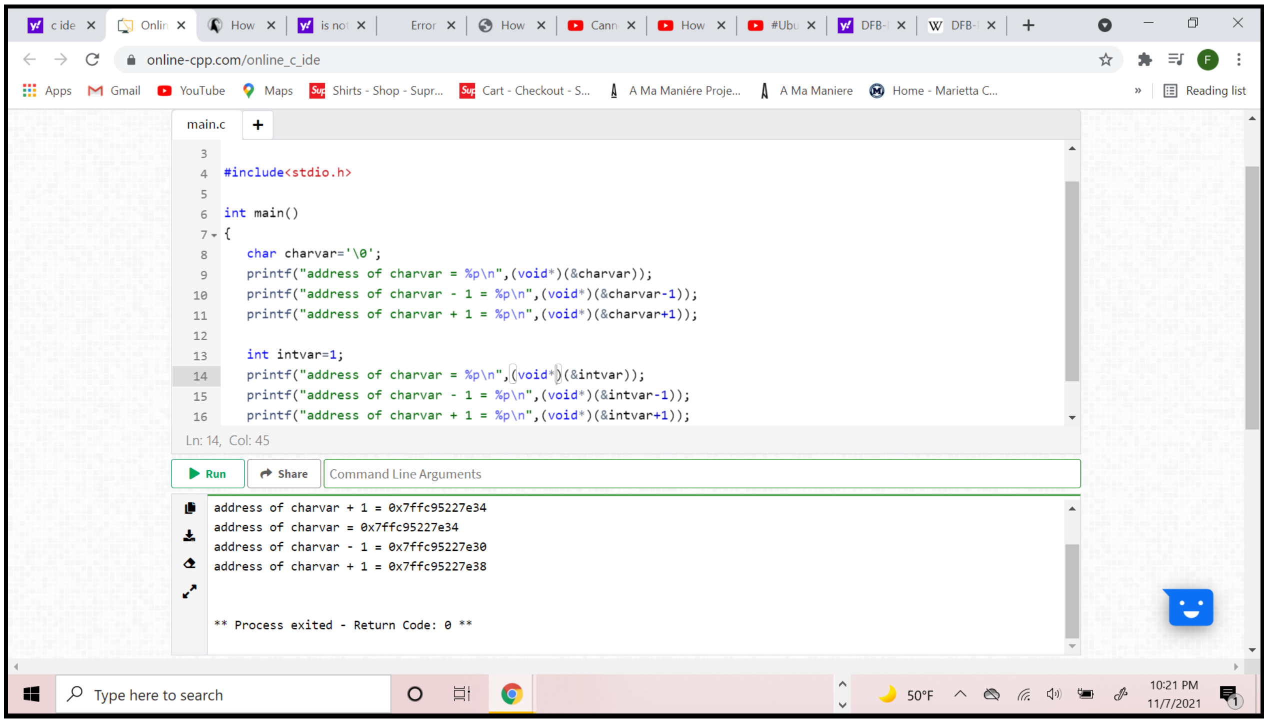
Part 2:

When a variable is stored in memory, it is associated with an address. To obtain the address of a variable, the & operator can be used. For example, &a gets the memory address of variable a. Let's try some examples.
Write a C program addressOfScalar.c by inserting the code below in the main function.

Questions:
1) Run the C program, attach a screenshot of the output in the answer

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

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



```c

#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 charvar = %p\n",(void*)(&intvar));
   printf("address of charvar - 1 = %p\n",(void*)(&intvar-1));
   printf("address of charvar + 1 = %p\n",(void*)(&intvar+1));
```

```
address of charvar + 1 = 0x7ffc95227e34
address of charvar = 0x7ffc95227e34
address of charvar - 1 = 0x7ffc95227e30
address of charvar + 1 = 0x7ffc95227e38


** Process exited - Return Code: 0 **
```

- (int intvar=1) Intvar is a variable declaration and is a data type of an integer and then the integer data type takes 4 bytes of memory and so the address of intvar increases by 4 bytes instead of 1 byte
- (char charvar='\0') charvar is a variable declaration and is a data type of a character and the character data takes 1-byte memory and so the address of charvar increases by 1 byte instead of 4 bytes

Part 3:

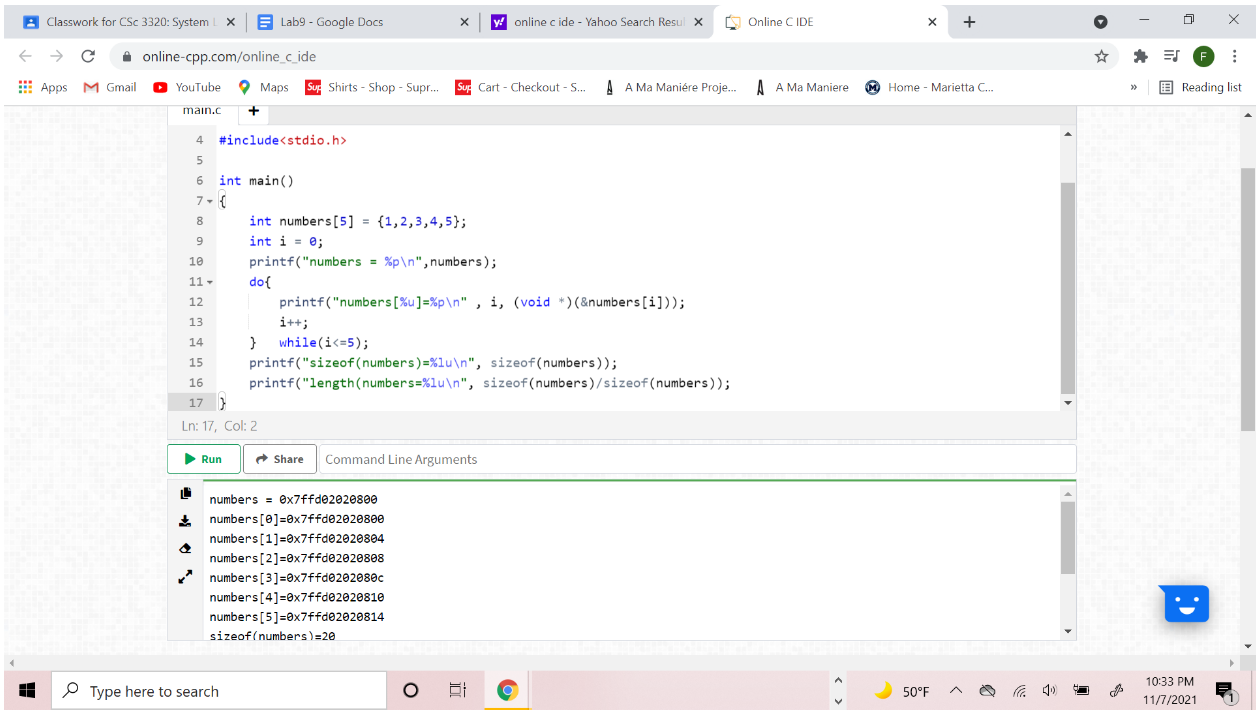
Write a C program addressOfArray.c by inserting the code below in the main function.

~~1 // initialize an array of ints~~
2 int numbers[5] = {1,2,3,4,5};
3 int i = 0;
4
5 // print the address of the array variable
6 printf("numbers = %p\n", numbers);
7
8 // print addresses of each array index
9 do {
10 printf("numbers[%u] = %p\n", i, (void *)(&numbers[i]));
11 i++;
12 } while(i < 5);

    // print the size of the array
    printf("sizeof(numbers) = %lu\n", sizeof(numbers));

Questions:

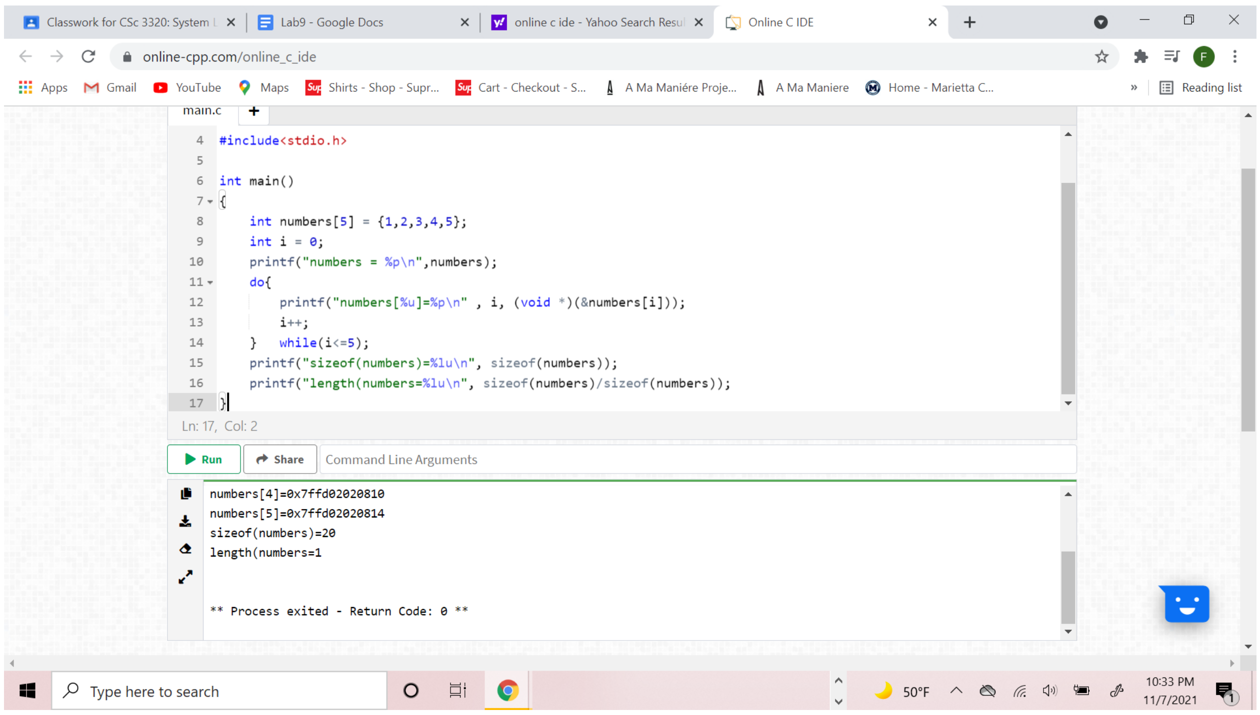
1) Run the C program, attach a screenshot of the output in the answer sheet.



```c
#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));
    printf("length(numbers=%lu\n", sizeof(numbers)/sizeof(numbers));
}
```

```
numbers = 0x7ffd02020800
numbers[0]=0x7ffd02020800
numbers[1]=0x7ffd02020804
numbers[2]=0x7ffd02020808
numbers[3]=0x7ffd0202080c
numbers[4]=0x7ffd02020810
numbers[5]=0x7ffd02020814
sizeof(numbers)=20
```



```c
#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));
    printf("length(numbers=%lu\n", sizeof(numbers)/sizeof(numbers));
}
```

```
numbers[4]=0x7ffd02020810
numbers[5]=0x7ffd02020814
sizeof(numbers)=20
length(numbers=1


** Process exited - Return Code: 0 **
```

2) Check the address of the array and the address of the first element in the array. Are they the same?

Both the element in 0 and the one before that have both the same addresses so we can say that both address of the array and 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]));

#The length of the array can be determined by the size of the divided by the size

of any element in the array

Submission:

- Upload an electronic copy (pdf) of your answer sheet to the folder named “Lab 9” in Google Classroom
- Please add the lab assignment number and your name at the top of your answer sheet.
- Upload the C files getMostFreqChar.c, addressOfArray.c and addressOfScalar.c to the

folder named named “Lab 9” in Google Classroom

- Name your file in the format of Lab9_ FirstnameLastname (e.g Lab9_FilRondel.pdf)