INHA UNIVERSITY TASHKENT
DEPARTMENT OF CSE & ICE

SPRING SEMESTER 2019

SOC 2040 - SYSTEMS PROGRAMMING

LAB ASSIGNMENT 1

INSTRUCTIONS :

- **All LAB assignments are to be completed in groups of 4 to 6 students.**
- **Screen shots are to be provided wherever necessary.**
- **LAB Assignment Report should be prepared using the Template provided.**
- **One Hard Copy of the LAB Assignment of each group should be handed in at the office by the Group Leader.**
- **Every member of the team must upload the softcopy of the report at the E-Class portal.**
- **LAST DATE FOR SUBMISSION OF THE LAB ASSIGNMENT IS 12th FEB. 2018**
- **Late submissions are not entertained, Adhere to the deadline strictly**
- **READ THE QUESTIONS CORRECTLY & CAREFULLY**

QUESTIONS :

PART I : VIRTUAL BOX AND LINUX INSTALLATION

1. What is Virtualization? Discuss the Virtual Box features in detail. **(Provide proper References)**

2. Install the latest Virtual Box on your Computer and provide the complete step-by-step installation steps. **Include all the installation screenshots and also indicate the settings used.**

3. Explain what is meant by an Operating System. Discuss the functions performed by an operating System. **(Provide proper References)**

4. Write short notes on Unix & Linux Operating Systems describing their history and features. **(Provide proper References)**

5. Install Ubuntu 16.04 LTS/ 18.04 LTS on the Virtual Box in your computer and provide the complete step-by-step installation steps. **Include all the installation screenshots and also indicate the settings used.**

6. What is Unix/Linux Shell? Indicate which shell (default shell) who will be normally in when you login to the Linux Machine.

7. List out the contents of your home directory, root directory and subdirectories in root directory.

PART II : USE OF EDITORS, C PROGRAMMING, COMPILATION & EXECUTION ON LINUX

8. Describe the commands of Vi/Vim Editor.

9. a) Using **Vi/Vim** Editor, create the following file in your user directory by entering the command at the linux shell prompt : **$vi sum_avgl.c**

b) Compile the program using gcc (**$gcc -o sum_avgl sum_avgl.c**) and execute the executable program(**$./sum_avgl**).

c) **Provide screenshots of the program entered, command executed and all the results (you need to try out atleast 12 lists containing more than 10 numbers).**

//C Progam file name : **sum_avgl.c**

// C Program to read a list of n integers from the keyboard and compute the sum and average

```
#include <stdio.h>
int main()
{
  int i, n, sum, list[100];
  float avg;

  // Read the value of n
  printf("\n Ener the size of the list : ");
  scanf("%d", &n);

  // Read the  list of 'n' numbers from the keyboard
  printf("\n Enter %d Elements of the list : \n", n);
  for(i=0; i<n; i++)
  {
    printf("\n list[%d]=",i);
    scanf("%d",&list[i]);
```

```
    }
    //Compute the sum of the list
    sum=0;
    for(i=0; i<n; i++)
       sum += list[i];

    //Compute the average of the list
     avg=(float)sum/n;
     printf("\nsum of %d elements in the list= %d and average = %f\n",n,sum,avg);
}
```

10. a) Using **Vi/Vim** Editor, create the following file in your user directory by entering the command at the linux shell prompt : **$vi key_search.c**

b) Compile the program using gcc **($gcc -o key_search key_search.c**) and execute the executable program(**$./key_search**).

c) **Provide screenshots of the program entered, command executed and all the results (you need to try out atleast 12 lists containing more than 10 numbers).**

//C Progam file name : **key_search.c**

```
// C Program to read a list of n integers and a key from the keyboard and search for the key in
//the given list – If the key is found in the list then print that key is found and its location in the
//list. Also print the number of occurrences of the key in the list. If the key is not found in the
//list then print the message- key not found

#include <stdio.h>
int main()
{
      int i, n, key, count=0, list[100];
      float avg;

     // Read the value of n
      printf("\n Ener the size of the list : ");
      scanf("%d", &n);

      // Read the list of 'n' numbers from the keyboard
      printf("\n Enter %d Elements of the list : \n", n);
      for(i=0; i<n; i++)
      {
        printf("\nlist[%d]=",i);
        scanf("%d",&list[i]);
      }

      // To search for a key in the above list
      printf("\nEnter the search key :");
      scanf("%d", &key);
      for(i=0; i<n; i++)
      {
       if(list[i]==key)
        {
```

```
            printf("\nkey %d is found at location %d in the list\n",key,i);
            count++;
        }
      }
      if(count==0)
            printf("\nKey %d not found in the list \n", key);
      else
          printf("\nKey %d occurs %d times in the list\n", key, count);
}
```

11. a) Using **gedit** Editor, create the following 'C' program file in your user directory by entering the command at the linux shell prompt : **$gedit palindrome.c**

b) Compile the program using gcc **- $gcc -o palindrome palindrome.c** and execute the executable program - **$./palindrome**

c) **Provide screenshots of the program entered, command executed and all the results (you need to try out atleast 15 names of different characters)**.

//Progam file name : **palindrome.c**

// C Program to read a name from the keyboard and perform the following operations on the //name : //(i) find the length of the name (ii) reverse the name and (iii) check whether the given //name is palindrome

```
#include <stdio.h>
#include <string.h>
int main()
{
    int i, j, k, l, len;
    char  ch, name[25], rname[25], uname[25], lname[25];

    printf("\n Ener your name : ");
    fgets(name, sizeof(name), stdin);
    printf("\n HELLO  %s\n", name);

    len = strlen(name);
    printf("\n Length of your name : %d\n", --len);
            name[len]='\0';

    // Reversing the name
    strcpy(rname, name);
    j=len-1;
    k=len/2;
    for(i=0; i<k; i++)
    {
            ch=rname[j];
            rname[j]=rname[i];
            rname[i]=ch;
            j--;
    }
      rname[len]='\0';
     printf("\n REVERSED NAME : %s\n",rname);
```

```
    // Checking for Palindrome
    if  (!strncmp(rname, name,len))
            printf("\nThe given name is a PALINDROME \n");
    else
            printf("\nThe given name is not a PALINDROME \n");
}
```

12. a) Using **gedit** Editor, create the following 'C' program file in your user directory by entering the command at the linux shell prompt : **$gedit lower2upper.c**

b) Compile the program using gcc **- $gcc -o lower2upper lower2upper.c** and execute the executable program - **$./lower2upper**

c) **Provide screenshots of the program entered, command executed and all the results** (**you need to try out atleast 15 names of different characters**).

```
//Progam file name : lower2upper.c
// C Program to read a name from the keyboard and convert all the lowercase characters
//to uppercase and display the results


#include <stdio.h>
#include <string.h>
int main()
{
        int i, j, k, l, len;
        char  ch, name[25], rname[25], uname[25], lname[25];

        printf("\n Ener your name : ");
        fgets(name, sizeof(name), stdin);

        len = strlen(name);
        printf("\n Length of your name : %d\n", --len);
                name[len]='\0';

        // Converting Lowercase characters to Uppercase
        strcpy(uname, name);
        for(i=0; i<len; i++)
        {
          if((uname[i] >= 'a')  && (uname[i] <= 'z'))
                uname[i]=uname[i]-32;
        }
        printf("\nThe given name in UPPERCASE : %s \n", uname);
  }
```

13. a) Using **gedit** Editor, create the following 'C' program file in your user directory by entering the command at the linux shell prompt : **$gedit upper2lower.c**

b) Compile the program using gcc **- $gcc -o upper2lower upper2lower.c** and execute the executable program - **$./upper2lower**

c) **Provide screenshots of the program entered, command executed and all the results** (**you need to try out atleast 15 names of different characters**).

//Progam file name : **upper2lower.c**
// C Program to read a name from the keyboard and convert all the uppercase characters
//to lowercase and display the results

```
#include <stdio.h>
#include <string.h>
int main()
{
        int i, j, k, l, len;
        char  ch, name[25], rname[25], uname[25], lname[25];

        printf("\n Ener your name : ");
        fgets(name, sizeof(name), stdin);

        len = strlen(name);
        printf("\n Length of your name : %d\n", --len);
        name[len]='\0';

        //Converting Uppercase characters to Lowercase
        strcpy(lname, name);
        for(i=0; i<len; i++)
        {
           if((lname[i] >= 'A')  && (lname[i] <= 'Z'))
                lname[i]=lname[i]+32;
        }
        printf("\nThe given name in LOWERCASE : %s \n", lname);
}
```

14. Write a 'C' program to read a list of 'n' integers from the keyboard and rearrange the elements in the list in ascending order and display the results. Use **gedit** to enter the program, **gcc** to compile the program and execute. **Provide screenshots of the program entered, command executed and all the results** (**you need to try out atleast 15 lists containing more than 10 positive and negative integers**).

15. Write a 'C' program to read a list of 'n' integers from the keyboard and count the number of positive numbers, negative numbers, odd numbers, even numbers and prime numbers in the given list and display the results. Use **gedit** to enter the program, **gcc** to compile the program and execute. **Provide screenshots of the program entered, command executed and all the results** (**you need to try out atleast 15 lists containing more than 10 positive, negative, odd, even and prime integers**).

16. Write a 'C' program to read 'n' real numbers (floating point numbers) from the keyboard and rearrange the elements in descending order and display the results. Use **gedit** to enter the program, **gcc** to compile the program and execute. **Provide screenshots of the program entered, command executed and all the results** (**you need to try out atleast 15 lists containing more than 10 positive and negative real numbers**).

17. Write a **'C' program** to read 'n' names (strings) from the keyboard and rearrange the names in alphabetical order and display the results. Use **gedit** to enter the program, **gcc** to compile the program and execute. **Provide screenshots of the program entered, command executed and all the results** (**you need to try out atleast 15 lists containing more than 10 names of different characters**).

18. Write a **'C' program** to read a sentence containing uppercase characters, lowercase characters, numerals and other special characters from the keyboard and convert all the uppercase letters to lowercase letters and display the results. Use **gedit** to enter the program, **gcc** to compile the program and execute. **Provide screenshots of the program entered, command executed and all the results** (**you need to try out atleast 15 sentences of different characters**).

19. Write a **'C' program** to read a sentence containing uppercase characters, lowercase characters, numerals and other special characters from the keyboard and convert all the lowercase letters in the sentence to uppercase letters and display the results. Use **gedit** to enter the program, **gcc** to compile the program and execute. **Provide screenshots of the program entered, command executed and all the results** (**you need to try out atleast 15 sentences of different characters**).

20. Write a **'C' program** to read a sentence containing uppercase characters, lowercase characters, numerals and other special characters from the keyboard and count the number of characters, alphabets, numerals and words in the given sentence and display the results. Use **gedit** to enter the program, **gcc** to compile the program and execute. **Provide screenshots of the program entered, command executed and all the results** (**you need to try out atleast 15 sentences of different characters**).

PART III : UNIX/LINUX COMMANDS

21. Describe the purpose of the following Unix/Linux commands providing correct syntax(usage format of the command). Use **man** command at the shell to understand the use and purpose of these commands (For example: enter the command **$man pwd** to get help to understand **pwd** command).
Provide all the man commands output screen shots.

i) ls, ls -la	ii) pwd	iii) who	iv) date	v) cat	vi) cp
vii) mv	viii) cd	ix) mkdir	x) more	xi) less	xii) head
xiii) tail	xiv) ps, ps -la	xv) sudo	xvi) find	xvii) grep	
xviii) echox	ix) chmod	xx) free	xxi) df	xxii) du	
xxiii) passwd	xxiv) wc	xxv) rm			

22. This question must be attempted only after completing PART II – Questions 9 to 13.

Execute the following commands one after another in the same given order 1) to 85), examine the output correctly to understand the commands and indicate the outputs generated. Clearly explain the format of the output with all the fields and their meaning.

Provide all the screen shots of commands executed and the obtained output results.

1) **pwd**
2) **ls**
3) **ls -la**
4) **uname –r**
5) **uname –a**
6) **cat palindrome.c**
7) **cat search.c**
8) **cp sum_avgl.c sa_list.c**
9) **cat sa_list.c**
10) **cp lower2upper.c l2u.c**
11) **cat l2u.c**
12) **mkdir spha1**
13) **cp l2u.c spha1/f1.c**
14) **cp sa_list.c spha1/f2.c**
15) **cp search.c spha1/f3.c**
16) **cd spha1**
17) **ls**
18) **cat f1.c**
19) **mv f2.c f4.c**
20) **ls**
21) **mv f3.c f5.c**
22) **ls**
23) **cp f5.c f6.c**
24) **cat f6.c**
25) **ls**
26) **rm *.***
27) **cat f6.c**
28) **ls**
29) **pwd**
30) **cd ..**
31) **pwd**
32) **history**
33) **chmod 777 palindrome.c**
34) **ls –l palindrome.c**
35) **chmod 751 palindrome.c**
36) **ls –l palindrome.c**
37) **cat sum_avgl.c search.c > progall.c**
38) **cat progall.c**
39) **cat palindrome.c lower2upper.c upper2lower.c >> progall.c**
40) **more progall.c**
41) **less progall.c**
42) **head -15 progall.c**

43) tail -10 progall.c
44) head -1 progall.c
45) tail -1 progall.c
46) cat progall.c | more
47) ls
48) rm l2u.c
49) ls
50) cp search.c find.c && ls && cat find.c
51) echo "HELLO GOOD MORNING! HOW ARE YOU"
52) echo "HELLO GOOD MORNING! HOW ARE YOU" | rev
53) echo $PWD
54) echo $PATH | more
55) wc progall.c
56) wc -c progall.c
57) wc -w progall.c
58) wc -l progall.c
59) find / -name passwd
60) find / -name search.c
61) grep "stdio" progall.c
62) grep "scanf" progall.c
63) grep "printf" progall.c
64) grep "for" progall.c
65) grep "uname" progall.c
66) cd /
67) pwd
68) ls && ls -la
69) cd
70) pwd
71) cd ..
72) pwd
73) cd
74) pwd
75) df -h
76) du -h
77) ps
78) ps –la
79) date
80) cal
81) passwd
82) who
83) who -a
84) **cat > myteamlist**
[After you press Enter key, cursor will move to the next line and wait for input data. Now **enter your team member complete details – Name, Student ID, Section Number & Email ID one after another on separate lines** and at the end **press Control D** (**Ctrl & D keys** simultaneously) to save the entered text in file **myteamlist**]
85) ls && cat myteamlist

****