INTRODUCTION

- Programming cannot be learned by watching others do it. Students must spend numerous hours working on programs themselves.
- This laboratory manual is a tool that will allow students to experiment with computer science & this is the beginning. As students progress through each laboratory, they may wonder how or why something works. The best way to discover the answer is to try things out.
- The purpose of this lab. manual is to acquaint the students to know the programming language as well as developing programming skills using C language.

STRUCTURE OF THIS LAB. MANUAL

- This lab. manual provides study aids from programming assignments to scheduled exercises using prepared materials.
- This lab manual is divided into 10 laboratory classes. Each laboratory class consists of the following:
  a) **Sample Answers (SA):** These are the complete program samples that students will go through in detail before coming to the laboratory class, may refer during solving lab assignments.
  b) **Lab Assignments (LA):** These are the assignments that ask each student to independently create small programs during the lab time.
  c) **Home Assignments (HA):** These are the assignments to be done during lab time if lab. assignments are completed before lab. time or may be assigned as post-lab homework and submitted in the next lab class.

The approach of each Lab: **SA-LA-HA**

INSTRUCTIONS FOR STUDENTS

To make laboratory experiments effective, each student must obey the following rules:

1. **General instructions**
   - Once you create a directory named as your rollno_section under the home directory of UBUNTU OS system using command-line or by GUI.
   - In Each lab, store programs within appropriate folders named as LAB01, LAB02, LAB03...etc. which are the sub folders under your rollno_section folder.
   - Always save programs files with the meaningful name preceded by lab assignment no within specified folders. If you want solve a lab assignment no. HA3.5 (3.5 means 5th assignment of 3rd lab) which is to find roots of a quadratic equation, then name the program as HA35_quadratic.c or HA35_quadeq.c etc.

2. **Attendance:** Attendance is required at all labs without exception. There are no make-up labs in this course. Performance will be judged based on the experiments conducted, quality and punctual submission of the labs reports for each experiment. Faculty/Instructor will take attendance. Failure to be present for an experiment will result in loosing entire marks for the corresponding lab. However, genuine cases may be considered for repeat lab. If a student misses a lab session due to unavoidable circumstances can provide a legitimate proof as soon as possible, he/she may be then be allowed by the lab instructor, to make-it-up.
3. **Laboratory Report:** At the end of every lab student will be assigned to write-up one of the experiment's problem. Your report must present a clear and accurate account, results you obtained. Student should develop habit to submit the laboratory report/assignments continuously and progressively on the scheduled dates and should get the assessment done.
4. Read the write up of each experiment to be performed, a day in advance. Understand the purpose of experiment and its practical implications.
5. Student should not hesitate to ask any difficulty faced during conduct of practical / exercise.
6. The student shall study all the questions given in the laboratory manual and practice to write the answers to these questions.
7. Student shall develop the habit of evolving more ideas, innovations, skills etc. those included in the scope of the manual.
8. Student should develop the habit of not to depend totally on teachers but to develop self learning techniques.
9. While entering into the LAB students should wear their ID cards.
10. Shut down your system after you have finished with your experiment.

PROCEDURE FOR EVALUATION

The entire lab course consists of 100 marks. The marking scheme is as follows

Continuous Evaluation marks	60
End Sem. Lab Examination	40
Total	**100**

Scheme for continuous evaluation

Students will be evaluated bi-weekly. Minimum 6 evaluations should be conducted for each student. Each evaluation carries 10 marks. The scheme is as follows:

Program & Execution	5
Observation	3
Viva-Voce	2
Total	**10**

Scheme for end sem lab examination

End sem. lab exam will be conducted after the completion of all the weekly exercises. The student will not be allowed for exam if he/she is found short of attendance and has not completed all the experiments. The marking scheme for end sem lab exam is as follows:

Write-up of program	15
Program execution & Checking Results for all inputs	15
Final Viva-Voce	10
Total	**40**

CONTENTS

LAB - 1

Linux/Unix Commands

Compilation, Execution of a program in GCC Compiler

CONTENTS

Experiment No-1

Sample Answers

SA1.1 To get familiar with LINUX/UNIX (UBUNTU) Operating System and practice some frequently used commands on terminal (Command Prompt).

SA1.2 To get familiar with **gedit** editor to create a new file, read the contents of a file, write into a file or modify the contents of a file.

SA1.3 To learn how to compile and execute a C file that displays "Welcome to C Programming Laboratory" in **gcc** compiler on terminal (Command Prompt).

SA1.4 WAP to display "IIT" using the character '*'.

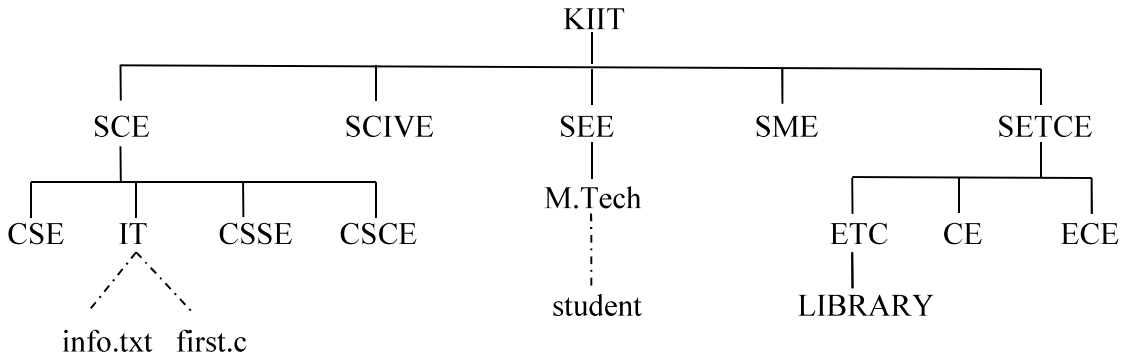
SA1.5 WAP to display the following message by using multiple printf statement.

A Good End
Can Only Be Achieved
Only By Good Means.

SA1.6 WAP to display the above message by using a single printf statement.

Lab. Assignments

LA1.1 First create a sub-directory named as your roll number under your home directory. Then create the following directory structure under your rollno directory.



N.B. The names under solid lines are assumed as directories and dotted lines as file names.

Do the following operations

a) Create the file names under the directories as mentioned in the figure and write some relevant data into the files.
b) Rename the file info.txt as itstudentsdata.txt.
c) Copy the file first.c into the directory CE with the same name.
d) Copy the file first.c into the directory SME with a new name as hello.c.
e) Transfer the file student into the directory SCIVE and check whether transferred or not.

LA1.2 WAP to display "KIMS" using the character '#'.

LA1.3 WAP to display the following message by using multiple printf statement.

If The End Is Good,
Then It Is Good,
Whatever Be The Means.

LA1.4 WAP to display the message of LA 1.3 by using single printf statement.

Home Assignments

HA1.1 WAP to print your BIO-DATA (Name, Regd.no", Branch, JEE Rank, Gender, Phone no., Address etc.) using printf statement.

PROGRAM NO. SA1.1

To get familiar with LINUX/UNIX (UBUNTU) Operating System and practice some frequently used commands on terminal (command prompt).

LINUX/UNIX COMMANDS

$ _ Command Prompt
Anything written within [ ] is optional.

Sl. No.	Command	Description	Example
1.	man	**Manual** **Syntax** **man commandName** It displays an on-line manual page for a command that it gives detailed information of a command how to use it.	**$ man ls** It gives the manual page of ls command **$ man pwd** You will see the manual for the pwd command.
2.	ls	**List** **Syntax** **ls [option(s)] [file(s)]** It lists the contents of a directory, and can be used to obtain information on the files and directories within it.	**$ ls** It lists the files & subdirectories available in the current directory. **$ ls -l** Same as above except it lists the files 'long format', which contains lots of useful information, e.g. the exact size of the file, who owns the file and who has the right to look at it, and when it was last modified. **$ ls dir1** It lists the files & subdirectories available in dir1.
3.	pwd	**Print Working Directory** **Syntax** **pwd**	**$ pwd** It tells you where you currently are, in which directory.

		It Shows the current location in the directory tree. In other words, the command gives the full pathname of your current directory.	
4.	**cd**	**Change Directory** **Syntax** **cd [options(s)] [directory]** It changes the current directory to other directory depending on the options and/or name of the directory.	**$ cd** It changes to the user's home directory. **$ cd ~** Same as above. **$ cd dir1** It changes to the directory dir1 if dir1 is a sub directory of your current working directory. **$ cd /home/user1/kiit/csit** It changes to csit directory as mentioned in the full path from your current working directory. **$ cd ..** It simply move up one directory. For example, if you are in /home/user1/kiit/csit and you type "cd ..", you will end up in /home/user1/kiit After applying cd .. yoi can verifywith pwd command.
5.	**mkdir**	**Make Directory** **Syntax** **mkdir [option(s)] directoryName** It creates a new directory.	**$ mkdir sce** It creates a new directory named as sce under your current directory. After executing this command, check through ls whether sce directory is created or not. If you want to create a new directory under other than your current directory, then mention the full path name before the new directory name. As for example, if your current working directory is /home/user1/kiit, but you want to create a new directory named as ece under /home/user1/kiit/setce, then ececute the following command: **$ mkdir** /home/user1/kiit/setce/ece
6.	**cp**	**Copy** **Syntax**	**$ cp file1 file2** It copies the contents of the file file1 into

		cp [option(s)] sourcefile targetfile Copies sourcefile to targetfile. Both file will be present.	a new file called file2. If you apply ls command, it will show you both the files. **$ cp ak.txt bk.txt dir1** It creates copies of files ak.txt and bk.txt (with the same names), within the directory dir1. dir1 must already exist for the copying to succeed. **$ cp file1 /home/user1/kiit/scive** It copies the contents of the file file1 into the directory scive with the same name. **$ cp -r dir1 dir2** It recursively copies the directory dir1, together with its contents and subdirectories, to the directory dir2. **$ cp -i quard.c quradeq.c** It waits for confirmation, if necessary, before an existing targetfile quardeq.c is overwritten. quard.c is copied as quradeq.c. Now If you apply ls command Then it will show you both the files quard.c and quradeq.c.
7.	**mv**	**Move** **Syntax** **mv [option(s)] sourcefile targetfile** It moves a file to a new location, or renames it. Source file name will be deleted.	**$ mv info.txt itstudentsdata.txt** It simply renames the file **info.txt as itstudentsdata.txt** . info.txt is deleted and only file available in current directory is itstudentsdta.txt. **$ mv -b info.txt itstudentsdata.txt** It ceates a backup copy of the sourcefile info.txt before moving it to itstudentsdata.txt. It is similar to cp command. **$ mv /home/user1/kiit/sce/it/first.c home/user1/kiit/sme** It simply move or transfer the file first.c into sme directory, no matter where is your current directory as both source and destination paths are mentioned. **$ mv /home/user1/kiit/sce/it/first.c home/user1/kiit/sme/firstcprog.c**

			It simply move or transfer the file first.c into sme directory with a new name firstcprog.c **$ mv -i quard.c quradeq.c** It waits for confirmation, if necessary, before an existing targetfile quardeq.c is overwritten. quard.c is renamed as quradeq.c. Now If you apply ls command Then it will show you only quradeq.c.
8.	**rm**	**Remove** **Syntax** **rm [option(s)] file(s)** It removes the specified files from the file system. Directories are not removed by rm unless the option -r is used.	**$ rm quard.c** It deleted the file quard.c available in the current directory. **$ rm –i quard.c** It waits for confirmation before deleting quard.c
9.	**rmdir**	**Remove Directory** **Syntax** **rmdir [option(s)] directoryName** It deletes the specified directory, provided it is already empty.	**$ rmdir dir1** If dir1 is empty, then it deletes the directory dir1 present under current directory.
10.	**whereis**	**Syntax** **whereis file** `It shows possible locations of file.`	**$ whereis quard.c** It shows you the location of the quard.c file.

Other LINUX/UNIX Commands (to know the detail about the following command use man)
date, cat, tail, which, locate, find, ps, id, du, clear, echo, grep, sot, su, ln, kill, chmod, ssh, tar, gzip, ping etc.

PROGRAM NO. SA1.2

To get familiar with **gedit** editor to create a new file, read the contents of a file, write into a file or modify the contents of a file.

gedit:

Text Editor (gedit) is the default GUI text editor in the Ubuntu operating system

1. To create a new file in c (first.c), run the following in command prompt.
   $ gedit first.c
   It will open the gedit editor window with the name first.c where you can write anything (program code for first.c).

Then save the contents of this file by choosing the appropriate options from gedit menu as follows:
File → Save

Now quit from gedit window and return to command prompt, do the following:
File → Exit

2. To open an existing file (say first.c) for editing do the following:
$ gedit first.c
After editing will be over, save the file and quit from gedit window.

PROGRAM NO. SA1.3

To learn how to compile and execute a C file that displays "Welcome to C Programming Laboratory" in **gcc** compiler on terminal (Command Prompt).

PROCEDURE

Step-1: Create a file named as **sa13_first.c** in gedit editor and write the following program code in it, then save the file and quit from gedit window.

Step-2: Compile the C Program file named as sa13_first.c
$ gcc sa13_first.c
It compiles the file sa13_first.c, if it is error free, then go for execution to get output. Else open the file again in gedit to correct the errors, again compile it till it does not show any errors.

Step-3: To get the output do the following
$ ./a.out

PROGRAM CODE

```
#include <stdio.h>
int main()
{
   printf("\n Welcome to C Programming Laboratory \n");
   return 0;
}
```

INPUT/OUTPUT

RUN-1
Welcome to C Programming Laboratory

PROGRAM NO. SA1.4

WAP to display "IIT" using the character '*'.

PROGRAM CODE

```
#include <stdio.h>
int main()
{
    printf("\n\n");
    printf("*****   *****   *******  \n");
    printf("    *       *         *      \n");
    printf("    *       *         *      \n");
    printf("    *       *         *      \n");
    printf("    *       *         *      \n");
    printf("    *       *         *      \n");
    printf("*****   *****       *      \n");
    return 0;
}
```

INPUT/OUTPUT

RUN-1

```
*****   *****   *******
   *       *       *
   *       *       *
   *       *       *
   *       *       *
   *       *       *
*****   *****      *
```

PROGRAM NO. SA 1.5

WAP to display the following message by using multiple printf statement.

A Good End
Can Only Be Achieved
Only By Good Means.

PROGRAM CODE

```
#include <stdio.h>
int main()
{
    printf("\n A Good End ");
    printf("\n Can Only Be Achieved ");
    printf("\n Only By Good Means.");
    return 0;
}
```

INPUT/OUTPUT

RUN-1
A Good End
Can Only Be Achieved
Only By Good Means.

PROGRAM NO. SA 1.6

WAP to display the following message by using multiple printf statement.

A Good End
Can Only Be Achieved
Only By Good Means.

PROGRAM CODE

```
#include <stdio.h>
int main()
{
    printf("\n A Good End\nCan Only Be Achieved\nOnly By Good Means.");
    return 0;
}
```

INPUT/OUTPUT

RUN-1
A Good End
Can Only Be Achieved
Only By Good Means.