RESEARCH METHODS IN COMPUTER SCIENCE

Laboratory Manual

August – November 2008

Table of Contents

This laboratory manual lists the procedures to be followed by the students during the RESEARCH METNODS IN COMPUTER SCIENCE laboratory sessions.

The manual consists of the experiments to be performed each week and also the format for reporting the results.

SESSION 1

DATA PREPROCESSING

1.1 Objectives

The main aim of the experiments on data preprocessing is to introduce tools and utilities that enable raw data coming as output from various programs to be converted into formats more suitable for analysis. For example, numerical data coming from programs may be converted into multi-column format for analysis in a spreadsheet or a plotting package. In particular, students are expected to become familiar with Linux/Unix utilities such as GREP, SORT, TR and CUT for data analysis.

1.2 Experiments

The following **3 experiments must be completed before the end of the lab session** and the results be submitted to the TAs on duty before leaving the lab.

Experiment 1: Extracting word frequency data

Students are required to extract the following information about the content of the file `rmcs2008-expt1.1` using any of the Linux/Unix utilities discussed thus far.

1. divide the file into multiple files containing no more than 50 lines each and find the numbers of words of length < 5 characters, of length between 5 and 7 characters and words longer than 7 characters from the file. Output the data as *file name*<TAB>*short word count*<TAB>*medium word count*<TAB>*long word count*. Store the output in a file named `wordcount_sml.dat`

2. divide the file into multiple files containing no more than 50 lines each. For each such file, find out the number of words beginning with vowels and the number of words beginning with consonants and output them as: *file name*<TAB>*Vowel word count*<TAB>*Consonant word count*. Store the output in the file `wordcount_vc.dat`

Experiment 2: Finding number of files in directories

Students are required to generate data on the number of files in different directories in the following formats.

1. Find the number of files (excluding directories) in `/, /bin, /usr/bin` and `/var` directories. Output them as *dir name*<TAB>*no. of files*
2. Find the number of directories in the above directories and output them in the same two column format as above
3. Find the number of files with sizes $> 100KB$ in `/, /bin, /usr, /usr/bin` and `/usr/sbin` directories and output them in a two column format with the name of the directory and the number of files.

Experiment 3: Finding the number of unique words in a file

Students are required to find the number of unique words in a file using `sort` command.

Divide the file `rmcs2008-expt1.2` into a number of files containing no more than 50 lines each. For each file, count and output the number of unique words it contains in a two column format with the file name and the number of words. Save the above information into the file `wordcountun.dat`

RMCS LAB REPORT Date:

Roll No. **Experiment No.**

AIM OF THE EXPERIMENT:

INPUT SPECIFICATIONS:

EXPECTED OUTPUT:

DESIGN AND APPROACH:

RESULTS:

REMARKS AND ANALYSIS:

SESSION 2

USING GNUPLOT

2.1 Objectives

The broad aim of this experiment is to introduce the extremely powerful plotting package GNUPLOT that is freely available on Linux systems. A more specific aim is to make the user aware of the plotting capabilities of GNUPLOT, how data needs to be prepared for input and the different forms in which the output may be rendered within and exported from GNUPLOT.

> HELP FOR PLOTTING DATA MAY BE OBTAINED WITHIN GNUPLOT ITSELF BY TYPING 'HELP PLOT' AT GNUPLOT PROMPT.

2.2 Experiments

The following **5 experiments must be completed before the end of the laboratory session** and the results submitted to the TAs on duty before leaving the lab.

Experiment 1: Simple x-y plots

Problem Definition

Numerical data is given in the file `rmcs2008-expt2.1` in three columns. The first column is the *x-coordinate or independent variable* and the second column is the *y-coordinate or the dependent variable*. The data should be plotted such that

1. data points should be displayed as points and the points should be connected by lines
2. data points should be represented by vertical lines (as in histograms)
3. data points should be displayed as points connected by lines and a coordinate grid should be displayed in the background

Repeat the above for the data in the *third* column.

Experiment 2: Multiple x-y plots

Problem Definition

Data is given in the file `rmcs2008-expt2.2` containing information about certain files and percentages.

1. Plot the *S. No. vs. Accuracy* data for documents labelled "D" under "Class" column using *impulses*
2. Plot the *S. No. vs. Accuracy* for the first nine input images as separate curves using *lines-points* style in the same plot for "Class B, Class C" and "Class D" documents. The y-axis should range from $70 - 100$. Show the key as "Class A", "Class B", etc. in the top-right corner.

Experiment 3: Plotting functions and data together

Problem Definition

In this experiment, we learn how to define functions and plot them along with data obtained from files.

Let us suppose that you wish to plot the parabola $y = 0.6x^2 - 3$. You can do so by typing

```
plot y = 0.6*x**2 - 3
```

The default is to plot using line style. You can also plot a function by *defining* it. First, type

```
myfunc(x) = 0.6*x**2 - 3
```

to define your own function `myfunc(x)`. Now plot it as `plot myfunc(x)`.

Finally, plot the data in the file `rmcs2008-expt2.3` using *points* style. Define a function $y = 1.95x - 4.21$ and plot it along with the data from the file. Do you think the function is a good approximation to the data?

RMCS LAB REPORT Date:

Roll No. **Experiment No.**

AIM OF THE EXPERIMENT:

INPUT SPECIFICATIONS:

EXPECTED OUTPUT:

DESIGN AND APPROACH:

RESULTS:

REMARKS AND ANALYSIS:

SESSION

3 DRAWING AND STATISTICAL ANALYSIS

One of the most common tasks in Computer Science research is data analysis and visualization of concepts. Most computer science papers contain figures, tables and statistical analysis to project various salient aspects of the paper. The aim of this session is to familiarize students with some drawing tools and spreadsheet packages to enable visualization and analysis of data.

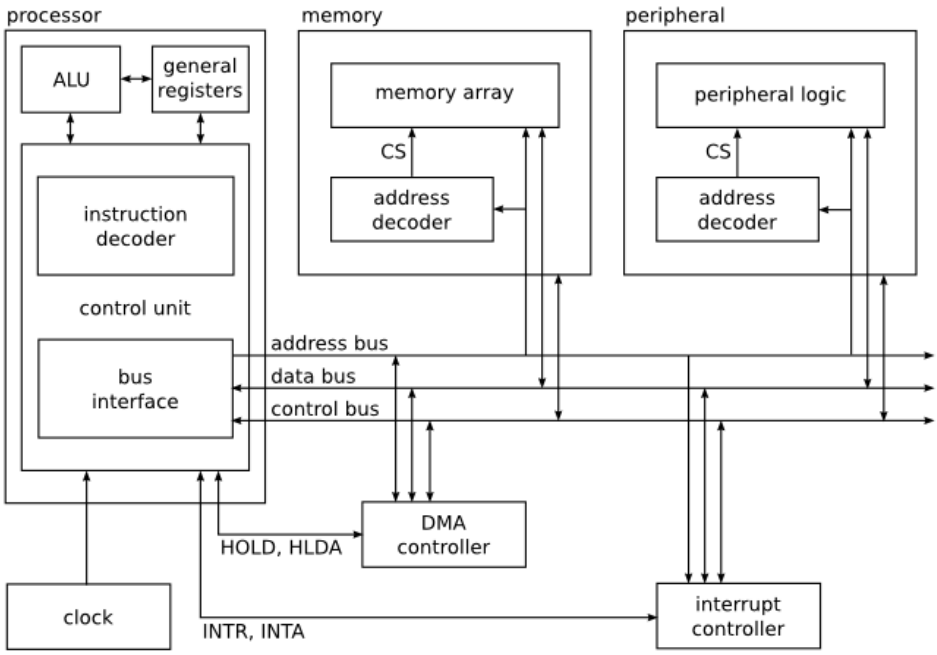
> MOST DRAWING PACKAGES COME WITH THE SAME SET OF TOOLS AND FUNCTIONS ALTHOUGH THE ICONS AND THEIR LOCATIONS MAY BE DIFFERENT. BUT FAMILIARITY WITH ONE TOOL SHOULD BE SUFFICIENT TO LEARN OTHER TOOLS WITHOUT MUCH ADO. THE SAME APPLIES FOR SPREADSHEETS TOO!

3.1 Experiments

This session contains three experiments — two on drawing and one on spreadsheets. Please complete them before you finish the session.

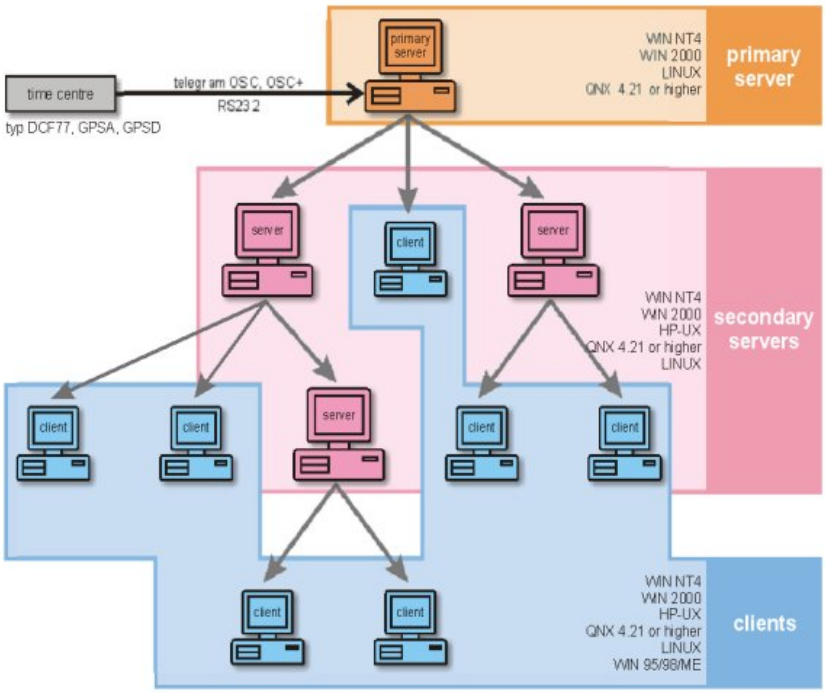
Experiment 1: Block Diagrams

Use the line, polyline, rectangle and other basic shapes available in the drawing tool `xfig` or `skencil` to draw the following figure. After saving the figure, export it to *encapsulated postscript (.eps)* format.



Experiment 2: Advanced Drawing

Use the colors and gradients in addition to copying and pasting tools for the drawing given below. Once again, export the result into *encapsulated postscript (.eps)* format after saving.



Experiment 3: Using Spreadsheets for Data Analysis

Refer to the file `rmcs2008-expt2.2` given in the previous session and read it into `oocalc` spreadsheet. Answer the following questions about the data:

1. There are two sets of 20 data files each for *Class C* documents. Is there any difference in the quality of the two sets based on the distribution of *accuracy* values?
2. Is there any difference between the data sets for *Class B* and *Class D*?
3. Open the file `rmcs2008-expt3.3.ods`. There are three columns of data. Is the data in Column 2 similar in distribution to the data in Column 3?

Note that you need to use the different statistical functions, AVERAGE, VARIANCE, STDEV and statistical tests FTEST, TTEST for answering the above questions. Also, remember that if the values returned by the tests are close to 1, the two sets are similar and if the values are close to 0, they are dissimilar.

RMCS LAB REPORT Date:

Roll No. **Experiment No.**

AIM OF THE EXPERIMENT:

INPUT SPECIFICATIONS:

EXPECTED OUTPUT:

DESIGN AND APPROACH:

RESULTS:

REMARKS AND ANALYSIS:

SESSION

4 TYPESETTING WITH LaTeX 2ε

This lab assignment is to familiarize you with the fundamentals of the LaTeX 2ε package for creating technical documents of the highest professional quality. By the way, LaTeX is pronounced "Lay-tech" or "Lah-tech." If you refer to LaTeX in an ASCII environment, you type `LaTeX`. LaTeX 2ε is pronounced "Lay-tech two e" and typed `LaTeXe`.

4.1 History

The TeX[1] project was started in 1978 by D. Knuth, while revising the second volume of his *Art of Computer Programming*. When he got the galleys back, he saw that the publisher had switched to a new digital typesetting system and was shocked at the poor quality.

He reasoned that because digital typesetting meant arranging 1's and 0's (ink and no ink) in the proper pattern, as a computer scientist he should be able to do the job better. He originally estimated that this would take six months but ultimately it took nearly ten years. He had to handle not only the challenges of routine typesetting such as right-justification and page formatting flexible enough to allow for different output styles, but also the additional demands of academic publishing – footnotes, floating figures and tables, etc. And, beyond that, he had to tell the computer how to typset formulas and other technical materials.

A year after he began, Knuth was invited to present one of the principal lectures at the AMS's annual meeting. He spoke on his TeX work, and also on Metafont (his system for developing fonts). He presented not only the roots of the typographical concepts, but also the mathematical notions on which these two programs are based. TeX's popularity took off from there.

An important boost to that popularity came in 1985 with the introduction by L. Lamport of LaTeX, a set of commands that allows interaction with the system at a higher level than Knuth's original set (which is called Plain TeX).

TeX systems remain popular today. Every day, research preprints, drafts of textbooks, and conference proceedings are produced with TeX. And, active development of

[1] Information about TeX, LaTeX and other derivatives is available at the CTAN (Comprehensive Tex Archive Network) website, "http://www.ctan.org". The URL for this material is "http://www.ctan.org/what_is_tex.html"

the system continues. Members of the community contribute a steady stream of new and updated enhancement packages, there have been great improvements in LaTeX's font-handling, and also improvements in TeX's ability with multilingual texts, there is now a version of TeX that outputs directly to the web-friendly PDF format, and much more.

4.2 Why LaTeX 2ε?

Word processors place text while you type it. The marketing term for this is *WYSIWYG*, "what you see is what you get". In contrast, LaTeX 2ε is a formatter: it separates the steps of entering the material and placing it on the page.

To see the difference, consider how a typical user of each system might start a new section. In a word processor a typical user might start that section by hitting `<Enter>` twice to get two lines of vertical space, typing in `"Section 1.2: New results"`, clicking to highlight that text, then clicking to select a larger type size and a new type style for that highlighted area, and entering two more lines of vertical space. A typical user LaTeX 2ε user will type into a file the line `"\section{New results}"`. The word processing user is formatting as they enter the text, while the LaTeX 2ε user describes the structural meaning of the text and will later run the file through LaTeX 2ε to have the program format it.

To a beginner, the word processing approach seems appealing. But as that user starts to try bigger and tougher jobs, laying it out by hand becomes hard. In a twenty page article, keeping the vertical space between sections uniform is error-prone work, and so is making sure that all of the bibliographic entries follow the required format. In addition, very few authors have the knowledge and aesthetic eye to correctly lay out and size the symbols in an equation. So, as a user becomes more experienced and knowledgable, the (La)TeX philosophy – to have the typesetting done by the program, as far as possible – becomes the better choice. (Some word processors offer as advanced features LaTeX 2ε-like facilities for organizing input text, although few users take advantage of them.)

Another problem with the approach of having users click in their material appears when the text is automatically generated, say as a report drawn from a database. Getting a word processor into that work flow is a challenge. But LaTeX 2εexpects input from a plain text file, so the database output is easy to format.

4.3 Using LaTeX 2ε

If LaTeX 2ε is correctly installed on the system, the following steps show its use in producing fully-formatted documents.

1. Type the material as a regular text file (in any editor such as `vi` or `emacs`). Save the text in a file with a `.tex` extension such as `foo.tex`.
2. Type `latex foo.tex` at command prompt of your terminal. This will generate normally three files: `foo.dvi`, `foo.aux` and `foo.log`. The `.dvi` file contains the output in a *device-independent format*, i.e., it needs to be converted for printing or display on almost any device.
3. To view it in `.dvi` format, use a *DVI previewer* such as `xdvi`. Type `xdvi foo.dvi` at the command prompt to view the output.
4. The dvi output can also be converted into POSTSCRIPT by typing `dvips foo -o`. It will produce an output file `foo.ps` which can be viewed in a standard document viewer such as `evince`. Type `evince foo.ps` at a command prompt to view the ps file.

4.4 Structure of a LaTeX 2ε document

LaTeX 2ε documents are in plain text, i.e., you simply type in the text in your favourite editor such as `vi, emacs` or even `gedit`. You may want to look for a LaTeX aware editor such as `kile` if you desire but such editors are really not necessary.

LaTeX 2ε document comprises two parts: PREAMBLE and TEXT. The preamble specifies the defaults and sets up various parameters that are global to the document. For example, the margins are specified in the preamble. The text is formatted and printed as *the* document.

All formatting instructions in LaTeX 2ε fall into three categories: *declarations, commands* and *environments*. If they take any arguments, the arguments are enclosed within a `{}` pair. We will see examples of each and use them in today's exercises. All commands and declarations begin with the '\' character. All environments are enclosed by a `\begin{`<environment name>`}` and `\end{`<environment name>`}` pair.

The preamble

Every LaTeX 2ε document starts with a preamble. A preamble starts with a `\documentclass` declaration. It is used to specify *the type of the document, paper size, font size for regular text* and a few other such document-related properties. The declarations and commands commonly used in the preamble are listed below:

documentclass Syntax: `\documentclass[`<options>`]{`<document type>`}`

Options are font size, paper size and others. Document type can be *report, article, book, letter* and others. We will look only at *article* type today. A typical document-class declaration is:

```
\documentclass[12pt,a4paper]{article}
```

which declares the default font size as 12 points (a point is approximately 1/72 inches), the paper as A4 (default is the US Letter size which is 8.5×11 inches) and the document type as article.

usepackage Syntax: `\usepackage[`<options>`]{`<package name>`}`

This is used to request for linking other packages that provide additional functionality (analogous to `#include <math.h>` in a C program to link mathematical functions). Packages commonly used are *geometry* and *graphicx*. The former allows simpler and greater control over margins while the latter enables inclusion of pictures, drawings and other graphical objects. Typical declarations are:

```
\usepackage[top=1in,left=1in,right=1in,bottom=1in]{geometry}
\usepackage{graphicx}
```

These specify text with 1in margins on all four sides and that graphics may be used later in the document.

pagestyle Syntax:`\pagestyle{`<options>`}`

Allows specification of headers and footers. If no pagestyle declaration is specified in the preamble, the default is no header and a footer displaying page number. Options are *empty, headings* and *myheadings*.

title Syntax: `\title{`<Title of the Document>`}`

Self-explanatory!

author Syntax: `\author{`<author(s)>`}`

Self-explanatory except that multiple authors may be separated with `\and`. Example:

```
\author{Chakravarthy Bhagvati \and Narayana Murthy Kavi}
```

date Syntax: `\date{`<date>`}`

Specifies the date on the document. It can be given explicitly as *23 August 2007* or as `\today` in which case, `\today` is substituted for the date on which the `.dvi` file is last created.

There are many other declarations and commands which can be found in the *cheat sheet* given or in any of the reference materials on LaTeX 2_ε.

Text

Text section contains the stuff that is displayed or printed. The text section begins with `\begin{document}` and ends with `\end{document}`. If you read the text at the beginning of Section 5, you realize that the text is inside a *document environment*. The

basic unit of text is a paragraph which is generated as you type in the text. Whitespaces such as *space, tab* and *newline* are treated in identical fashion: each is treated as a word terminator. A paragraph ends when you type a blank line in the text. For example, if you type:

```
Today is my first day with
\LaTeXe.

This is my second sentence in \LaTeXe.
```

It will be printed as

Today is my first day with LaTeX 2$_\varepsilon$.

This is my second sentence in LaTeX 2$_\varepsilon$.

Useful commands

`\maketitle` command is used to create the title once the necessary information is properly declared in the preamble. It can be placed anywhere after the `\begin{document}` declaration.

The following commands change text style.

boldface `\textbf{`<text>`}`

emphasis `\emph{`<text>`}`

typewriter style `\texttt{`<text>`}`

small capitals `\textsc{`<text>`}`

These commands can be nested. For example, typing `\textbf{\emph{`<text>`}}` causes the text to become boldface with emphasis.

Useful environments

These generate various kinds of lists.

enumerate is used to generate numbered lists.

```
\begin{enumerate}
\item   This is my first list item.
\item   This is my second list item.
\item   This is my third list item.
\end{enumerate}
```

produces

1. This is my first list item.
2. This is my second list item.
3. This is my third list item.

itemize is used to generate 'bulleted' lists. The syntax is the same as that for the `enumerate` environment.

center is used to produce centered lines of text. All the text between a `\begin{center}` and `\end{center}` pair is centered.

There are many other environments such as *quote*, *equation*, *description* and others.

SESSION

5 ADVANCED LaTeX 2ε

The primary motivation for developing the TeX and the LaTeX 2ε systems is formatting mathematics such that the output is of the highest quality possible. In this set of experiments, the aim is to learn how to write maths, insert tables and graphical objects into a LaTeX 2ε document.

5.1 Typesetting Equations

The real power and beauty of LaTeX 2ε becomes evident when we write equations. There are three major types of equations: *in-line, simple equation* and *system of equations*.

In-line equations are produced by enclosing the equation between a pair of $ symbols.

Typing

```
A second-degree polynomial is of the form:
$a_0 + a_1x + a_2x^2$
```

results in

A second-degree polynomial is of the form: $a_0 + a_1x + a_2x^2$

Simple equation is produced by the *equation* environment.

Typing

```
Consider a second-degree polynomial of the form:
\begin{equation}
        a_0 + a_1x + a_2x^2
\end{equation}
```

results in

Consider a second-degree polynomial of the form:

$$a_0 + a_1x + a_2x^2 \tag{5.1}$$

Note that the equation is now centered on a separate line and also numbered. If you want to suppress the numbering, use

`\begin{equation*}` and `\end{equation*}`. You can also use `\[` and `\]` instead of `\begin{equation}` and `\end{equation}` respectively.

System of equations are produced by the *eqnarray* environment. For example, typing

```
\begin{eqnarray}
      a(t_k) & = & x(t_k) + 3y(t_k) - 7x(t_k) + 2y(t_k) \\
             & = & 5y(t_k) - 6x(t_k)
\end{eqnarray}
```

produces

$$\begin{aligned} a(t_k) &= x(t_k) + 3y(t_k) - 7x(t_k) + 2y(t_k) && (5.2)\\ &= 5y(t_k) - 6x(t_k) && (5.3) \end{aligned}$$

Note the use of '\\' to separate the different equations, and & to tell where the equations are to be aligned. Also, note that the equations are automatically numbered in a sequence. There are two ways to suppress numbering: either use a *eqnarray** environment or use a `\nonumber` declaration within the *eqnarray* environment.

Typing

```
\begin{eqnarray}
  a(t_k) & = & x(t_k)+3y(t_k)-7x(t_k)+2y(t_k) \nonumber \\
         & = & 5y(t_k) - 6x(t_k)
\end{eqnarray}
```

produces

$$\begin{aligned} a(t_k) &= x(t_k) + 3y(t_k) - 7x(t_k) + 2y(t_k)\\ &= 5y(t_k) - 6x(t_k) && (5.4) \end{aligned}$$

Notice now that the first equation is not numbered.

Complex arrangements Now suppose we want to format an enumerated function as shown below.

$$f(x) = \begin{cases} \dfrac{x}{3} & \text{if } x \text{ is divisible by 3} \\ 3x+1 & \text{if } x \bmod 3 \equiv 2 \\ 3x+2 & \text{otherwise} \end{cases} \qquad (5.5)$$

Then, no doubt about it, we are in trouble! But, rest assured, far less trouble than if you did not have LaTeX 2_ε! This is what produced the enumerated function of Equation 5.5.

```
f(x) = \left\{
   \begin{array}{ll}
      \displaystyle{\frac{x}{3}} & \mbox{if~} x \mbox{~mod~} 3
                                      \equiv 0 \\
      3x + 1                     & \mbox{if~} x \mbox{~mod~} 3
                                      \equiv 2 \\
      3x + 2                     & \mbox{otherwise}
\end{array}\right.
```

These are only a few of the things you can do with LaTeX 2_ε. The full extent of the maths formatting abilities may be seen either by purchasing Leslie Lamport's book or by browsing the LaTeX 2_ε website `www.ctan.org`

5.2 Inserting Figures and Graphics Objects

It is very easy to insert a figure into a LaTeX 2_ε document. The only restriction is that the figure **must be in Encapsulated Postscript or eps format**. If one is interested only in PDF output, then it is possible to insert JPEG figures too.

Figures are inserted using the `graphicx` package. Type

`\usepackage{graphicx}`

in the *preamble* of the document. A figure named `uohlogo.eps`, for example, is inserted into the document with `\includegraphics{uohlogo.eps}` command. This command is normally placed inside a `figure` environment to attach a caption and be able to refer to it elsewhere in the text.

5.3 Experiments

The following experiments must be done before leaving the lab.

Expt - 1: Mathematics

Typeset the following matter and show the formatted output.

1. Although this equation looks very complicated, it should not present many difficulties in typesetting it.

$$\int_{-1}^{8} \frac{1}{\sqrt[3]{x}} dx = \frac{3}{2}(8^{2/3} + 1^{2/3}) = \frac{15}{2}$$

2. The function

$$\begin{aligned}\Gamma(x) &= \lim_{n\to\infty}\prod_{\nu=0}^{n-1}\frac{n!n^{x-1}}{x+\nu} \\ &= \lim_{n\to\infty}\frac{n!n^{x-1}}{x(x+1)(x+2)\cdots(x+n-1)} \\ &= \int_0^\infty e^{-t}t^{x-1}dt\end{aligned}$$

3. Define a function

$$y(x) = \begin{cases} -1 & : \quad x < 0 \\ 0 & : \quad x = 0 \\ +1 & : \quad x > 0 \end{cases}$$

Expt - 2: Tables

Create the table shown below.

1st Regional Hockey League – Final Results							
	Club	*W*	*D*	*L*	*Goals*	*Points*	*Remarks*
1	Hyderabad Sultans	19	13	1	66:31	51	Champions
2	Bangalore Bombers	18	10	5	65:37	45	Automatic
3	Chennai Challengers	15	10	8	63:53	41	Selection
4	Patiala Pathans	14	10	9	64:47	39	for
5	Delhi Dangers	10	15	8	62:58	35	next year
6	Mumbai Monsters	10	13	10	58:56	31	
7	Silchar Strikers	10	10	13	42:54	28	Demoted

RMCS LAB REPORT Date:

Roll No. **Experiment No.**

AIM OF THE EXPERIMENT:

INPUT SPECIFICATIONS:

EXPECTED OUTPUT:

DESIGN AND APPROACH:

RESULTS:

REMARKS AND ANALYSIS: