Laboratory Manual

MACHINE DESIGN-II LAB

for

B. Tech.
Mechanical Engineering

Department of Mechanical Engineering

Rukmani & Sons

Brij Bhooshan Education Web Portal
Web: www.brijrbedu.org

LABORATORY MANUAL

MACHINE DESIGN-II LAB

for

B. Tech.
Mechanical Engineering

Prepared by

A team of
Rukmani & Sons

E-mail: enquiry@brijrbedu.org
Website: http://www.brijrbedu.org

TABLE OF CONTENTS

LABORATORY OBJECTIVE

The primary objective of the programme is to train the students in Machine design field by imparting broad based knowledge on the state of the art engineering design methods analysis tools.

Machine design-II lab deals with the techniques to provide software based result of any type of design problem and in future for other data same program can be easily modified, but this is not possible without software thus provides comforts to perform better and have longer lives.

By conducting the Experiments in this laboratory as per this manual, following objectives will be fulfilled:

- To innovate and to create.
- A design which is best and economical for the given objective functions under the specified constraints.
- Carry out fault finding program.
- Undergo the conclusion and result by varying data.
- Conduct the trials on the different data.
- Get acquainted with the latest know-how of the field

ABOUT THE LABORATORY

This laboratory consists of computer with installed C language software for Study the C language programme for design mechanical component.

This laboratory contains the following setups and equipments:

1. Computer System
2. Installed C language Software.
3. C language setup

GUIDELINES FOR TEACHERS/TECHNICAL ASSISTANTS

1. Know the laboratory: The teacher is expected to understand the layout of laboratory, specifications of equipments/instruments/materials, procedure of experiments, method of working in groups, planning time etc.
2. Ensure that required equipments are in working condition before start of experiment and also keep the operating or instruction/user manuals of equipments/instruments and this laboratory manual available.
3. On the first day of the lab, inform the students about the importance of subject/laboratory, various equipments/instruments that will be used in the lab etc. Also instruct them how to make the practical record file for this lab.
4. Explain the theoretical concepts, relevant to the experiment, to the students before start of each practical.
5. Demonstrate the experiment(s) clearly to the students group-wise.
6. Instruct the students to perform the practical. While taking reading/observation, each student must get a chance to perform or observe the experiment.
7. If the experimental setup has variations in the specifications of the equipment, the teachers are advised to make the necessary changes.
8. Teacher shall assess the performance of students by observation or by asking viva related questions to the students to tap their achievements regarding related knowledge/skills so that students can prepare accordingly.
9. The teacher must check carefully and sign the practical record file of the students periodically.
10. Teacher shall ensure that the industrial/site/plant visits recommended as per the syllabus of laboratory are covered.
11. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
12. Teacher may provide additional knowledge and skills to the students albeit not covered in the manual but are expected from students by the industries.
13. Teacher may suggest the students to refer additional related literature of the technical papers, reference books, seminar proceedings etc.
14. Teacher can organize group discussions/brain storming sessions/seminars to facilitate the exchange of practical knowledge amongst the students.

GENERAL PRECAUTIONS AND SAFETY PROCEDURES

1. Teacher/technical assistant must ensure that all the electrical equipments/ instruments are used and periodically performance tested as per manufacturer's recommendations (permissible electrical and ambient temperature ratings).
2. Before use, the electrical equipment, extension cords, power tools etc. must be inspected for any damage (worn insulation, bent/missing pins, etc.). Any equipment found to be damaged or otherwise unsafe must be removed from service.
3. The mains plug of equipments must only be inserted in a socket outlet provided with a protective earth contact.
4. **WARNING**: The protective earth connection inside or outside the equipments/instruments must NEVER be interrupted or tampered. **IT CAN MAKE THE EQUIPMENT DANGEROUS**.
5. If an instrument shows visible damage or fails to perform the intended measurements, it is likely that the protection has been impaired. In such case the instrument must be made inoperative and the necessary repairs should be carried out.
6. Extension cords or power strips must not be plugged into one another so as to increase the overall reach.
7. Report all problems with building electrical systems to the teacher/technical assistant/maintenance for corrective action.
8. In case of any electrical hazard/fire reach out for the nearest fire-extinguisher or sand and use it for putting out the fire. Report immediately to the teacher/ technical assistant nearby.
9. For reasons of safety, every student must come to the laboratory in shoes (covering the whole feet).
10. Avoid wearing garments with loose hanging parts. The students should also ensure that floor around the equipment/machine is clear and dry (not oily) to avoid slipping. Please report immediately to the lab staff on seeing any coolant/oil spillage.
11. The student should take the permission and guidance of the lab staff/teacher before operating any equipment/machine. Unauthorized usage of any machine without prior guidance may lead to fatal accidents and injury.
12. The student will not lean on the equipment/machine or take any kind of support of the machine at any point of time.

INSTRUCTIONS FOR STUDENTS

1. Listen carefully to the lecture and instructions given by the teacher about importance of subject/laboratory, curriculum structure, skills to be developed, information about equipment and instruments, procedure, method of continuous assessment, tentative plan of work in laboratory and total amount of work to be done in the semester/session.
2. Read and understand the theory of each experiment to be performed, before coming to the laboratory.
3. Understand the purpose of experiment and its practical implications. Observe carefully the demonstration of the experiment. When you perform it, organize the work in your group and make a record of all observations.
4. In case of absence, the student must perform the experiment(s) on the next turn or in his/her spare time with permission from the teacher/lab assistant.
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 or asked by the teacher and know the answers to these questions properly.
7. The required instruments/tools will be issued from the laboratory store. They must be returned to the store on the same day at the end of lab hours.
8. Laboratory reports (practical file) should be submitted in a bound file or on A4 size sheets, properly filed, on the next turn completed in all respects i.e. with experiment(s) written, graphs attached (if applicable) and entries made in the list of contents of the file and get them checked from your laboratory teacher. Laboratory reports have associated grades/marks.
9. Student should not bring any food or drink item to the laboratory.
10. Student should develop habit of group discussion related to the experiments/exercises enabling exchange of knowledge/skills.
11. Student shall gain knowledge and develop required practical skills and competencies as expected by the industries.
12. Student shall develop the habit of evolving more ideas, innovations, skills etc. than included in the scope of the manual.
13. Student shall refer technical magazines, proceedings of the seminars; refer websites related to the scope of the subjects and update their knowledge and practical skills.

EXPERIMENT – 1

OBJECTIVE:

Introduction of C language.

INTRODUCTION:

C is a general-purpose programming language, and is used for writing programs in many different domains, such as operating systems, numerical computing, graphical applications, etc. It is a small language, with just 32 keywords. It provides "high-level" structured programming constructs such as statement grouping, decision making, and looping, as well as "low level" capabilities such as the ability to manipulate bytes and addresses. C achieves its compact size by providing Spartan services within the language proper, foregoing many of the higher-level features commonly built-in to other languages.

A First Program

A C program, whatever its size, consists of functions and variables. A function contains
Statements that specify the computing operations to be done, and variables store values
used during the computation.

```
/* First C program: Hello World */
        #include <stdio.h>
        #include <conio.h>

        main(void)
         {
         printf("Hello World!\n");
         }
```

- start with /* and are terminated with */. They can span multiple lines and are notnestable. For example,
  /* this attempt to nest two comments /* results in just one comment, ending here: */ and the remaining text is a syntax error. */
- Inclusion of a standard library header-file. Most of C's functionality comes from libraries. Headerfiles contain the information necessary to use these libraries, such as function declarations and macros.

- All C programs have main() as the entry-point function. This function comes in two forms:
  int main(void)
  int main(int argc, char *argv[])

The first takes no arguments, and the second receives command-line arguments from the environment in which the program was executed—typically a command-shell. The function returns a value of type int (i.e., an *integer)*

- The braces {and} delineate the extent of the function block. When a function completes, the program returns to the calling function.
- In the case of main(), the program terminates and control returns to the environment in which the program was executed. The integer return value of main() indicates the program's exit status to the environment, with 0 meaning normal termination.
- This program contains just one statement: a function call to the standard library function printf(), which prints a character string to standard output (usually the screen). Note, printf() is not a part of the C language, but a function provided by the standard library (declared in header stdio.h). The standard library is a set of functions mandated to exist on all systems conforming to the ISO Standard. In this case, the printf() function takes one argument.

Variants of Hello World

```
#include <stdio.h>
#include <conio.h>
main(void)
{
 printf("Hello World \n ");
 }
```

The next program also prints "Hello World!" but, rather than printing the whole string in one go, it prints it one character at a time. This serves to demonstrate several new concepts, namely

EXPERIMENT – 2

OBJECTIVE:

Types, Operators, and Expressions of C language.

INTRODUCTION:

Variables and constants are the basic data objects manipulated in a program. Declarations list the variables to be used, and state what type they have and perhaps what their initial values are. Operators specify what is to be done to them.

Identifiers

Identifiers (i.e., variable names, function names, etc) are made up of letters and digits, and are Case-sensitive. The first character of an identifier must be a letter, which includes underscore (_).1 The C language has 32 keywords which are reserved and may not be used as identifiers (eg, int, while, etc).

Types

C is a *typed* language. Each variable is given a specific type which defines what values it can
Represent, how its data is stored in memory, and what operations can be performed on it. By forcing the programmer to explicitly define a type for all variables and interfaces, the type system enables the compiler to catch type-mismatch errors, thereby preventing a significant source of bugs.

C Data Types	
Char	usually 8-bits (1 byte)
Int	usually the natural word size for a machine or OS (e.g., 16, 32, 64 bits)
short int	at least 16-bits
long int	at least 32-bits
float	usually 32-bits
float double	usually 64-bits
long double	usually at least 64-bits

Constants

Constants can have different types and representations. This section presents various constant types by example. First, an integer constant 1234 is of type int.

An constant of type long int is suffixed by an L, 1234L; (integer constants too big for int are implicitly taken as long).

```
int x = 1234, y = 02322, z = 0x4D2;
printf("%d\t%o\t%x\n", x, x, x);
printf("%d\t%d\t%d\n", x, y, z);
```

Conversion Specifies

The standard function printf() facilitates formatted text output. It merges numerical values of any type into a character string using various formatting operators and conversion specifiers.

```
printf("Character values %c %c %c\n", 'a', 'b', 'c');
printf("Some floating-point values %f %f %f\n", 3.556, 2e3, 40.1f);
printf("Scientific notation %e %e %e\n", 3.556, 2e3, 40.1f);
printf("%15.10s\n", "Hello World\n");
```

Declarations

All variables must be *declared* before they are used. They must be declared at the top of a block (a section of code enclosed in brackets *{* and *}*) before any statements. They may be *initialized* by a constant or an expression when declared. The following are a set of example declarations.

```
{ /* bracket signifies top of a block */
int lower, upper, step; /* 3 uninitialised ints */
char tab = '\t'; /* a char initialised with '\t' */
char buf[10]; /* an uninitialised array of chars */
int m = 2+3+4; /* constant expression: 9 */
int n = m + 5; /* initialised with 9+5 = 14 */
float limit = 9.34f;
const double PI = 3.1416;
```

Arithmetic Operations

The arithmetic (or numerical) operators come in two varieties: unary and binary. The binary operators are plus +, minus −, multiply *, divide /, and the modulus operator %. The first four operators can be used on integer or floating-point types, although it is important to notice that integer division truncates any fractional part.

```
int a=2, b=7, c=5, d=9;
printf("a*b + c*d = %d\n", a*b + c*d);
```

Relational and Logical Operations

There are six relational operators: greater-than >, less-than <, greater-than-or-equal-to >=, less than-or-equal-to <=, equal-to == and not-equal-to !=. Relational expressions evaluate to 1 if they are TRUE and 0 if they are FALSE. For example, 2.1 < 7 evaluates to one, and x != x evaluates to zero.

```
while (x = 3) {
/* various statements here */
}
```

RESULT:

Experiment has been performed successfully.

EXPERIMENT – 3

OBJECTIVE:

Branching and Iteration in C language.

INTRODUCTION:

The C language provides three types of decision-making constructs: if-else, the conditional expression?:, and the switch statement. It also provides three looping constructs: while, do-while, and for. And it has the infamous goto, which is capable of both non-conditional branching and looping.

If-Else

The basic if statement tests a conditional expression and, if it is non-zero (i.e., TRUE), executes the subsequent statement. For example, in this code segment

```
if (a < b)
b = a;
```

the assignment = a will only occur if a is less-than b. The else statement deals with the alternative case where the conditional expression is 0 (i.e., FALSE).

```
if (a < b)
b = a;
else
b += 7;
```

The if-else statement can also command multiple statements by wrapping them in braces. Statements so grouped are called a *compound statement*, or *block*, and they are syntactically equivalent to a single statement.

```
if (a < b) {
b = a;
a *= 2;
}
else {
b += 7;
--a;
}
```

Conditional Expression

The conditional expression is a ternary operator; that is, it takes three operands. It has the following form

(expression 1) ? (expression 2) : (expression 3)

If the first expression is TRUE (i.e., non-zero), the second expression is evaluated, otherwise the third is evaluated. Thus, the result of the ternary expression will be the result of either the second or third expressions, respectively. For example, to calculate the maximum of two values,

```
c = (a > b) ? a : b; /* c = max(a,b) */
```

As a branching construct, the ?: operator appears far less frequently than the if-else and switch constructs.

Switch

The switch statement is a multi-way decision that tests whether an expression matches one of a number of constant integer values, and branches accordingly.
The general form of the switch statement is as follows.

```
switch (expression) {
case const-int-expr: statements
case const-int-expr: statements
default: statements
}
```

While Loops

The while loop has the general form while (expression) statement;
If the conditional expression is TRUE, then the while will execute the following statement, after which it will reevaluate the conditional. It continues this iteration while ever the conditional remains TRUE. As with the if-else statement, the while loop can execute multiple statements as a block by enclosing them in braces. For example, the following code segment computes the *greatest common divisor* (GCD) of two positive integers m and n (i.e., the maximum value that will divide both m and n). The loop iterates until the value of n becomes 0, at which point the GCD is the value of m.

```
while (n) {
int tmp = n;
n = m%n;
m = tmp;
```

```
}
```

Do-While Loops

The do-while loop has the general form

```
do
statement;
while (expression);
```

Its behavior is virtually the same as the while loop except that it always executes the statement at least once. The statement is executed first and then the conditional expression is evaluated to decide upon further iteration. Thus, the body of a while loop is executed zero or more times, and the body of a do-while loop is executed one or more times.

For Loops

The for loop has the general form

```
for (expr1; expr2; expr3)
statement;
```

Its behavior is equivalent to a while loop with the following arrangement of expressions.

```
expr1;
while (expr2) {
statement;
expr3;
}
```

In the for loop, expressions 1, 2, and 3 are optional, although the semicolons must remain. If expressions 1 or 3 are not there, then the loop simple behaves like the while loop above without expressions 1 or 3. If expression 2 is omitted, then the conditional is always TRUE, and an infinite loop results.

Break and Continue

As seen previously, break can be used to branch out of a switch-statement. It may also be used to branch out of any of the iterative constructs. Thus, a break may be used to terminate the execution of a switch, while, do-while, or for. It is important to realise that a break will only branch out of an inner-most enclosing

block, and transfers program-flow to the first statement following the block. For example, consider a nested while-loop,

```
while (expression) {
while (expression) {
if (expression)
break;
statements
}
statements
}
```

Goto

The goto statement has a well-deserved reputation for being able to produce unreadable "spaghetti"code. It is almost never necessary to use one, and they should be avoided in general. However, on rare occasions they are convenient and safe.

```
#include <stdio.h> /* for printf() */
2 #include <stdlib.h> /* for rand() */
3 #include <string.h> /* for memset() */
4
5 #define SIZE 1000
6 enum { VAL1='a', VAL2='b', VAL3='Z' };
7
23
8 int main(void)
9 /* Demonstrate a legitimate use of goto (adapted from K&R page 66). This example is contrived, but the idea is to
10 * find a common element in two arrays. In the process, we demonstrate a couple of useful standard library functions. */
 {
  char a[SIZE], b[SIZE];
  int i, j;
  /* Initialise arrays so they are different from each other */
  memset(a, VAL1, SIZE);
  memset(b, VAL2, SIZE);
  /* Set a random element in each array to VALUE */
  a[rand()%SIZE] = VAL3;
  b[rand()%SIZE] = VAL3;
  /* Search for location of common elements */
```

```
for (i=0; i<SIZE; ++i)
for (j=0; j<SIZE; ++j)
if (a[i] == b[j])
goto found;
/* Error: match not found */
printf("Did not find any common elements!!\n");
return 0;
found: /* Results on success */
printf("a[%d] = %c and b[%d] = %c\n", i, a[i], j, b[j]);
```

RESULT:

Experiment has been performed successfully.

EXPERIMENT – 4

OBJECTIVE:

Write a programme in C language for Sliding Bearing.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int w, n, d;
float l, c, z, p, k, bm, u, q, kj;
printf ("enter the values");
printf ("diameter, load, speed, viscocity, bearing clearence");
scanf ("%d%d%d%f%f ",&d,&w,&n,&z,&c);
k=0.002;
l=1.5*d;
p=w/l*d;
bm=z*n/p;
u=(33*bm*c*10*10*10*10*10*10*10*10)+kj;
q=(u*w*3.14*d)/60;
printf ("length of bearing=%f ",l);
printf ("heat generated=%f ",q);
getch();
}
```

EXPERIMENT – 5

OBJECTIVE:

Write a programme in C language for design of Spur Gear.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
float cv, m, dg, dp, vr, sp, sg, yp, yg,y,w,wt,v,n,np,ng,s,d,b,tg,tp,t;
printf ("enter the values");
printf ("module, pinion speed, gear diameter, velocity ratio, pinion stress,
gear stress, face with, pinion teeth, tangential load, gear speed");
scanf ("%f%f%f%f%f%f%f%f%f%f ",
&m,&np,&dg,&vr,&sp,&sg,&tp,&wt,&ng);
ng=np*vr;
dp=dg/vr;
tg=tp*vr;
yp=0.175-0.951/tp;
yg=0.175-0.951/tg;
if (yp*sp>yg*sg);
{
s=sp;
n=np;
t=tp;
d=dg;
y=yp;
s=sp;
}
if (yg*sg>yp*sp);
{
```

```
s=sg;
n=ng;
t=tg;
y=yg;
s=sg;
}
v=(3.14*d*n)/60;
cv=6.1/(6.1+v);
if (wt<w);
printf ("design is not safe");
printf ("calculate tangential load=%f ",wt);
getch();
}
```

EXPERIMENT – 6

OBJECTIVE:

Write a programme in C language for design of cylinder head.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int bp, n, em, s;
float p, d, l, t,ip,k,ns,q,gf,dc,du,dp,pm,b,D,dcl,r;
printf ("enter the values");
printf ("Brake power, speed, mean effective pressure, stress, diameter,
mechanical efficieny");
scanf ("%d%d%f%d%f%d",&bp,&n,&pm,&s,&d,&em);
ip=(bp*100)/em;
l=1.5*d;
p=q*pm;
r=p/s;
ns=(0.015*p)+4;
gf=(3.14*d*d*pm)/4;
dcl=(gf*4)/(ns*3.14*s);
dc=sqrt(dcl);
dp=(3.14*d)+1.5;
p=(3.14*d)+1.5;
t=(0.045*d)+1.5;
printf ("the calculated values are");
printf ("thickness of cylinder head=%f ",t);
printf ("number of stud=%f ",ns);
printf ("length=%f ",l);
printf ("pitch=%f ",p);
```

```
getch();
}
```

EXPERIMENT – 7

OBJECTIVE:

Write a programme in C language for check the cylinder is leak proof or not.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int bp, em, s;
float pm,p, d, l, t,ip,n,r,q,rs,gt,dc,dp,gl,b,v,qt,qg,qd,m;
printf ("enter the values");
printf ("Brake power, mean effective pressure, stress, diameter,
mechanical efficieny");
scanf ("%d%f%d%f%d",&bp,&pm,&s,&d,&em);
ip=(bp*100)/em;
l=1.5*d;
b=l*pm;
v=p/8;
r=d/2;
q=sqrt(r);
t=0.31*d*q;
dc=(gl*4)/(rs*3.14*s);
qt=qg-qd;
m=(qt/s)*t;
getch();
}
```

EXPERIMENT – 8

OBJECTIVE:

Write a programme in C language for design of pistion.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
float pm,p,hpr,d,r,rt,ll,ip,th,tr,b,pw,st,dc,tb,s,rp;
printf ("enter the values ");
printf ("Mean effective pressure, stress, diameter, ring stress, radial
pressure on cylinder wall");
scanf ("%f%f%f%f%f",&pm,&s,&d,&st,&pw);
ll=0.45*d;
p=pm*10;
r=p/s;
rp=sqrt(r);
rt=(3*pw)/st;
dc=sqrt(rt);
th=(0.43*d*rp);
tr=th/3;
b=d*dc;
hpr=(0.7*b);
tb=(0.03*d)+b+4.9;
printf ("The calculate values are");
printf ("thickness of piston head=%f ",th);
printf ("thickness of rib=%f ",tr);
printf ("radial width of ring=%f ",b);
printf ("axial thickness of piston ring=%f ",hpr);
```

```
printf ("thickness of piston barrel at the top=%f ",tb);
printf ("length of pin=%f ",ll);
getch();
}
```