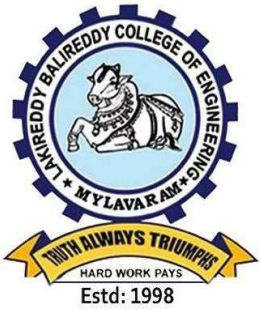
LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)



23CS51 – Computer Programming Lab Manual

Department of Computer Science and Engineering

Vision of the Department

The Computer Science & Engineering aims at providing continuously stimulating educational environment to its students for attaining their professional goals and meet the global challenges.

Mission of the Department

- **DM1:** To develop a strong theoretical and practical background across the computer science discipline with an emphasis on problem solving.
- **DM2:** To inculcate professional behaviour with strong ethical values, leadership qualities, innovative thinking and analytical abilities into the student.
- **DM3:** Expose the students to cutting edge technologies which enhance their employability and knowledge.
- **DM4:** Facilitate the faculty to keep track of latest developments in their research areas and encourage the faculty to foster the healthy interaction with industry.

Program Educational Objectives (PEOs)

- **PEO1:** Pursue higher education, entrepreneurship and research to compete at global level.
- **PEO2:** Design and develop products innovatively in computer science and engineering and in other allied fields.
- **PEO3:** Function effectively as individuals and as members of a team in the conduct of interdisciplinary projects; and even at all levels with ethics and necessary attitude.
- **PEO4:** Serve ever-changing needs of society with a pragmatic perception.

PROGRAMME OUTCOMES (POs):

PO 1 **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2 **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3 **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4 **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5 **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6 **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7 **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8 **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9 **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10 **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11 **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12 **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

PROGRAMME SPECIFIC OUTCOMES (PSOs):

PSO 1	The ability to apply Software Engineering practices and strategies in software project development using open-source programming environment for the success of organization.
PSO 2	The ability to design and develop computer programs in networking, web applications and IoT as per the society needs.
PSO3	To inculcate an ability to analyze, design and implement database applications.

Pre-requisites: Mathematics, Basic Computer Terminology

Course Educational Objectives (CEOs): The course aims to give students hands – on experience and train them on the concepts of the C- programming language.

Course Outcomes (COs): At the end of the course, the student will be able to :

CO-1: Read, understand, and trace the execution of programs written in C language. (Understand)

CO-2: Apply the right control structure for solving the problem. (Apply)

CO-3: Develop, Debug and Execute programs to demonstrate the applications of arrays, functions, basic concepts of pointers in C. (Apply).

CO-4:Improve individual / teamwork skills, communication and report writing skills with ethical values.

Course Code	COs	Programme Outcomes												PSOs		
		1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
23CS51	CO1	3	2	-	-	3	-	-	-	-	-	-	-	3	3	3
	CO2	3	2	2	-	3	-	-	-	-	-	-	-	3	3	3
	CO3	3	2	2	-	3	-	-	-	-	-	-	-	3	3	3
	CO4	-	-	-	-	-	-	-	2	2	2			-	-	-

1 = Slight (Low) 2 = Moderate (Medium) 3-Substantial(High)

LIST OF EXPERIMENTS:

Week 1: Familiarization with programming environment

a) Basic Linux environment and its editors like Vi, Vim & Emacs etc.
b) Writing simple programs using printf(), scanf()

Week 2: Converting algorithms/flow charts into C Source code.
Developing the algorithms/flowcharts for the following sample programs

a) Sum and average of 3 numbers
b) Conversion of Fahrenheit to Celsius and vice versa
c) Simple interest calculation

Week 3: Simple computational problems using arithmetic expressions.

a) Finding the square root of a given number
b) Finding compound interest
c) Area of a triangle using heron's formulae
d) Distance travelled by an object

Week 4: Simple computational problems using the operator precedence and associativity.

a) Evaluate the following expressions.
   I. A+B*C+(D*E) + F*G
   II. A/B*C-B+A*D/3
   III. A+++B---A
   IV. J= (i++) + (++i)
b) Find the maximum of three numbers using conditional operator
c) Take marks of 5 subjects in integers, and find the total, average in float

Week 5: Problems involving if-then-else structures.

a) Write a C program to find the max and min of four numbers using if-else.
b) Write a C program to generate electricity bill.
c) Find the roots of the quadratic equation.
d) Write a C program to simulate a calculator using switch case.
e) Write a C program to find the given year is a leap year or not.

Week 6: Iterative problems e.g., the sum of series

a) Find the factorial of given number using any loop.
b) Find the given number is a prime or not.
c) Checking a number palindrome
d) Construct a pyramid of numbers.

Week 7:1D Array manipulation, linear search

a) Find the min and max of a 1-D integer array.
b) Perform linear search on1D array.
c) The reverse of a 1D integer array
d) Eliminate duplicate elements in an array.

Week 8: Matrix problems, String operations, Bubble sort

a) Addition of two matrices
b) Multiplication two matrices

c) Sort array elements using bubble sort
d) Concatenate two strings without built-in functions
e) Reverse a string using built-in and without built-in string functions

Week 9: Pointers and structures, memory dereference.

a) Write a C program to find the sum of a 1D array using malloc()
b) Write a C program to find the total, average of n students using structures
c) Read student name and marks from the command line and display the student details along with the total.
d) Write a C program to implement realloc()

Week 10 :

a) Create and display a singly linked list using self-referential structure.
b) Demonstrate the differences between structures and unions using a C program.
c) Write a C program to copy one structure variable to another structure of the same type.

Week 11: Simple functions using call by value, solving differential equations using Eulers theorem.

a) Write a C function to calculate NCR value.
b) Write a C function to find the length of a string.
c) Write a C function to transpose of a matrix.

Week 12: Recursive functions

a) Write a recursive function to generate Fibonacci series.
b) Write a recursive function to find the LCM of two numbers.
c) Write a recursive function to find the factorial of a number.
d) Write a recursive function to find the sum of series.

Week 13: Simple functions using Call by reference,

a) Write a C program to swap two numbers using call by reference.
b) Write a C program to copy one string into another using pointer.
c) Write a C program to find no of lowercase, uppercase, digits and other characters using pointers.

Week 14: File operations

a) Write a C program to write and read text into a file.
b) Write a C program to write and read text into a binary file using fread() and fwrite()
c) Copy the contents of one file to another file.
d) Write a C program to merge two files into the third file using command-line arguments.
e) Find no. of lines, words and characters in a file
f) Write a C program to print last n characters of a given file.

WEEK 1: Familiarization with programming environment

i) Basic Linux environment and its editors like Vi, Vim & Emacs etc.

We use an editor in the UNIX system to create a C source program.

1. The following command is used to compile C program "helloworld.c"
2. Compilation Command: gcc helloworld.c -o helloworld
3. This command compiles the c program and generates an executable file "helloworld" for running the program.
4. In the command if we don't specify "-o helloworld" it creates an executable file "a.out"
5. The following command is used to execute C program "helloworld.c"
6. ./a.out - to execute the program if executable file name is not specified.
7. ./helloworld - to execute the program if executable file name is specified

ii) Writing simple programs using printf(), scanf()

```
#include <stdio.h>
void main()
{
        int a,b;
        printf("Enter a,b values\n");
        scanf("%d%d",&a,&b);
        printf("a=%d\nb=%d",a,b);
}
```

Week-2: Converting algorithms/flow charts into C Source code.

i. Write a C program on Sum and average of 3 numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int
  a,b,c,sum;
  float avg;
  clrscr();
  printf("enter the value of a,b,c:");
  scanf("%d %d %d",&a,&b,&c);
  sum=a+b+c;
  avg=(a+b+c)/3.0;
  printf("\nthe sum=%d",
  sum);
  printf("\nthe average=%f", avg);
  getch();
}
```

ii. Write a C program on Conversion of Fahrenheit to Celsius and vice versa

```
//CONVERSION OF FAHRENHEIT TO CELSIUS
#include<stdio.h>
#include<conio.h>
void main()
{
  float c,f;
  clrscr();
  printf("enter the value of
  fahrenheit:"); scanf("%f",&f);
  c=(f-32)*5/9;
  printf("fahrenheit of %f in celsius is %f",f,c);
  getch();
}
```

```
//CONVERSION OF FAHRENHEIT TO CELSIUS
#include<stdio.h>
#include<conio.h>
void main()
{
  float c,f;
  clrscr();
  printf("enter the value of celsius:");
  scanf("%f",&c);
  f=(c*9/5)+32;
  printf("celsius of %f in fahrenheit is %f",c,f);
  getch();
}
```

iii. Write a C program on Simple interest calculation.

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int p,t;
  float
  SI,r;
  clrscr();
  printf("enter the value of p,t,r:");
  scanf("%d%d%f",&p,&t,&r);
  SI=(p*t*r)/100;
  printf("SI is %f",SI);
  getch();
}
```

WEEK-3: Simple computational problems using arithmetic expressions.

i. Write a C program on Finding the square root of a given number?

```
#include<stdio.h>
#include<math.h>
void main()
{
int n;
float res;
read:
printf("Enter n value");
scanf("%d",&n);
if(n<0)
goto read;
res=sqrt(n);
printf("Square root of a given number=%.2f",res);
}
```

ii. Write a C program on finding compound interest?

```
// The formula for compound interest is given by: A=P(1+r/n)^nt
#include<stdio.h>
#include<math.h>
void main()
{
int P,r,n,t;
float A;
printf("enter the principal amount : ");
scanf("%d",&P);
printf("enter the rate of interest : ");
scanf("%d",&r);
printf("enter the compounding frequency per annum : ");
scanf("%d",&n);
printf("enter the time in years : ");
```

```
scanf("%d",&t);
A=P*pow((1+r/n),n)*t;
printf("compound intrest is %f",A);
}
```

iii. Write a C program on Area of a triangle using heron's formulae.

```
//area = sqrt( s*(s-a)*(s-b)*(s-c) )
#include<stdio.h>
#include<math.h>
void main()
{
int s,a,b,c,area,p;
printf("enter the value of a : "); //one side of triangle
scanf ("%d",&a);
printf("enter the value of b : "); //one side of triangle
scanf("%d",&b);
printf("enter the value of c : "); //one side of triangle
scanf("%d",&c);
s = (a+b+c)/2;
p = s*(s-a)*(s-b)*(s-c);
area = sqrt(p);
printf("%d",area);
}
```

iv. Write a C program on Distance travelled by an object?

```
//s = ut + (at^2)/2
#include<stdio.h>
void main()
{
int a,u,t;
float s;
printf("enter the value of a"); //a is acceleration
scanf("%d",&a);
printf("enter the value of u"); //u is initial velocity
```

```
scanf("%d",&u);
printf("enter the value of t"); //t is time
scanf("%d",&t);
s = u*t + (a*t*t)/2;
printf("distance travelled by an object is %f",s);
}
```

WEEK-4: Operators and the precedence and as associativity

i. Evaluate the following expressions.

a. A+B*C+(D*E) + F*G

b. A/B*C-B+A*D/3

c. A+++B---A

d. J= (i++) + (++i)

A) #include<stdio.h>

#include<conio.h>

#include<math.h>

void main()

{

int a,b,c,d,e,f;

int g,h,i,j,k,l;

printf("enter values of required variables:");

scanf("%d %d %d %d %d %d",&a,&b,&c,&d,&e,&f,&g);

h=a+b*c+(d+e)+f*g;

printf("first expression =%d\n",h);

k=a/b*c-b+a*d/3;

printf("second expression %d\n",k);

l=a++ +b- --a;

printf("third expression=%d \n",l);

j=(i++)+(++i);

printf("j=%d\n",j);

}

ii. Write a C program on Find the maximum of three numbers using conditional operator?

#include<stdio.h>

#include<conio.h>

void main()

{

clrscr();

int a,b,c,big;

printf("\n enter the values of a,b,c, values respectively:");

```
scanf("%d %d %d",&a,&b,&c);
big=(a>b&&a>c)?a:((b>c)?(b):(c));
printf("the greatest value among these three numbers is %d\n",big);
getch();
}
```

iii. Write a C program on take marks of 5 subjects in integers, and find the total, average in float.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
clrscr();
int i,n,sub,total=0;
float avg;
printf("enter the no.of subjects");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("subject  %d:",i);
scanf("%d",&sub);
total=total+sub;
}
printf("total=%d\n",total);
avg=(float)total/n;
printf("average = %.2f\n",avg);
getch();
}
```

WEEK-5: Branching and logical expressions

i. Write a C program to find the max and min of four numbers using if-else?

```
#include<stdio.h>
void main(){
int a,b,c,d;
printf("enter the values of a,b,c,d");
scanf("%d,%d,%d,%d",&a,&b,&c,&d);
if(a>b&&a>c&&a>d){
printf("%d is big",a);
}
else if(b>c&&b>d){
printf("%d is big",b);
}
else if(c>d){
printf("%d is big",c);
}
else{
printf("%d is big",d);
}
if(a<b&&a<c&&a<d){
printf("%d is small", a);
}
else if(b<c&&b<d){
printf("%d is small", b);
}
else if(c<d){
printf("%d is small", c);
}
else{
printf("%d is small", d);
}
getch();
```

```
}
```

ii. Write a C program to generate electricity bill.?

```
#include<stdio.h>
void main()
{
int initial_reading,final_reading,consumed_units;
float amount;
printf("enter the values of initial and final reading values : ");
scanf("%d,%d",&initial_reading,&final_reading);
consumed_units = final_reading -initial_reading;
if(consumed>400)
total = consumed_units *6.5;
else if(consumed_units <=400&&consumed_units >=300)
total = consumed_units *5.5;
else if(consumed_units <300&&consumed_units >=200)
total = consumed_units *4.5;
else if(consumed_units <200&&consumed_units >=100)
total = consumed_units *3.5;
else if(consumed_units <100&&consumed_units >=50)
total = consumed_units *2.5;
else if(consumed_units <50)
amount = consumed_units *2;
printf("the electricity bill for %d units is %f ",consumed_units, amount);
}
```

iii. Write a C program on Find the roots of the quadratic equation?

```
#include<stdio.h>
#include<math.h>
void main()
{
double a,b,c,root1,root2,det,real,imag;
printf("enter the values of a,b,c");
```

```
scanf("%lf,%lf,%lf",&a,&b,&c);
det = b*b-4*a*c;
if(det>0)
{
printf("roots are Not Equal”);
root1 = (-b+sqrt(det))/(2*a);
root2 = (-b-sqrt(det))/(2*a);
printf("roots of quadratic expression are %lf,%lf",root1,root2);
}
else if(det==0)
{
printf("roots are Equal”);
root1 = -b/(2*a);
printf("roots of quadratic expression are %lf,%lf",root1,root1);
}
else
{
printf("roots of the quadratic expression are complex");
}
}
```

iv. Write a C program to simulate a calculator using switch case

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
int a,b,c,d,op;
while(1)
{
printf("1.add\n2.sub\n3.mul\n.4.div\n5.exit");
printf("choose an operation :");
scanf("%d",&op);
```

```
printf("enter the values of a,b:");
scanf("%d%d",&a,&b);
switch(op)
{
case 1: c=a+b;
printf("addition=%d",c);
break;
case 2: c=a-b;
printf("subtraction=%d",c);
break;
case 3: c=a*b;
printf("multiplication=%d",c);
break;
case 4: c=a/b;
printf("division=%d",c);
break;
case 5: exit(0);
}
}
}
```

v. Write a C program to find the given year is a leap year or not.?

```
#include<stdio.h>
void main()
{
int year;
printf("enter the YEAR");
scanf("%d",&year);
if((year%4==0&&year%100!=0)||(year%400==0))
{
printf("%d is leap year", year);
}
```

```
else
{
printf("%d is not a leap year”, year);
}
}
```

WEEK-6: Loops, while and for loops

i. Write a C program to find the factorial of given number using any loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int i,n,fact=1;
printf("enter the value of n:");
scanf("%d",&n);
fact=1;
for(i=1;i<=n;i++)
{
fact=fact*i;
}
printf("the factorial of %d is %d", n,fact);
}
```

ii. Write a C program to find the given number is a prime or not.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,n,count=0;
printf("enter the value of n:");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
if(n%i==0)
{
count++;
}
```

```
}
if(count==2)
{
printf("%d is a prime no.",n);
}
else
{
printf("%d id not a prime no.",n);
}
}
```

iii. Write a C program on checking a number palindrome.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int temp,n,d,rev;
printf("enter the number");
scanf("%d",&n);
temp=n;
while(n>0)
{
d=n%10;
rev=rev*10+d;
n=n/10;
}
if(temp==rev)
{
printf("%d is a palindrome number",temp);
}
else
{
```

```
printf(“%d is not a palindrome number”,temp);
}
}
```

iv. Write a C program to construct a pyramid of numbers

```
#include <stdio.h>
void main() {
  int i, space, rows, k = 0;
  printf("Enter the number of rows: ");
  scanf("%d", &rows);
  for (i = 1; i <= rows; i++, k = 0) {
    for (space = 1; space <= rows - i; ++space) {
      printf("  ");
    }
    while (k != 2 * i - 1) {
      printf("* ");
      ++k;
    }
    printf("\n");
  }
  }
```

WEEK-7: 1D Array manipulation, linear search

i. Write a C program to find the min and max of a 1-D integer array.

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void main() {
int i,n,a[50],max,min;
clrscr();
printf("enter the size of array:");
scanf("%d",&n);
printf("\n enter the values of array:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
max=a[0];
min=a[0];
for(i=0;i<n;i++)
{
if(a[i]>max)
{
max=a[i];
}
if(a[i]<min)
{
min=a[i];
}
}
printf("\nthe max no. in given array is %d",max);
printf("\nthe min no. in given array is %d",min);
getch();
}
```

ii. Write a C program to perform linear search on1D array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[50],i,n,key,flag=0;
printf("Enetr size of array:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("Enter key element to search:");
scanf("%d",&key);
for(i=0;i<n;i++)
{
if(key==a[i])
{
flag=1;
break;
}
}
if(flag==1)
{
printf("%d is found", key);
}
else
{
printf("%d is not found", key);
}
}
```

iii. Write a C program on the reverse of a 1D integer array?

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void main()
{
clrscr();
int i,n,a[5];
printf(" the size of array A :");
scanf("%d",&n);
printf("enter elements into an array:");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("Array elements are\n :");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
printf("\n Array in rev order:");
for(i=n-1;i>=0;i--)
{
printf("%d\t",a[i]);
}
printf("array in reverse order is done!!");
getch();
}
```

iv. Write a C program to eliminate duplicate elements in an array.

```
#include <stdio.h>
#include <conio.h>
```

```
void main (){
int a[20], i, j, k, n;
printf (" Enter array size: ");
scanf (" %d", &n);
printf ("Enter elements of an array: \n ");
for ( i = 0; i < n; i++)
{
scanf (" %d", &a[i]);
}
for ( i = 0; i < n; i ++)
{
for ( j = i + 1; j < n; j++)
{
if (a[i] == a[j])
{
        for ( k = j; k < size - 1; k++) {
arr[k] = arr [k + 1];
}
n--;
j--;
}
}
}
printf (" \n Array elements after deletion of the duplicate elements: ");
for ( i = 0; i < n; i++)
{
printf (" %d \t", arr[i]);
}
}
```

WEEK-8: 2 D arrays, sorting and Strings

i. Write a C program on Addition of two matrices.

```
#include<stdio.h>
#include<conio.h>
void main() {
int m, n, i, j;
int a[m][n], b[m][n], c[m][n];
printf("Enter the number of rows and columns of the matrix: ");
scanf("%d%d", &m, &n);
printf("Enter the elements of matrix A: \n");
for (i = 0; i < m; i++) {
for (j = 0; j < n; j++) {
scanf("%d", &a[i][j]);
}
}
printf("Enter the elements of matrix B: \n");
for (i = 0; i < m; i++) {
for (j = 0; j < n; j++) {
scanf("%d", &b[i][j]);
}
}
for (i = 0; i < m; i++)
{
for (j = 0; j < n; j++)
{
c[i][j] = a[i][j] + b[i][j];
}
}
printf("The sum of the two matrices is: \n");
for (i = 0; i < m; i++) {
for (j = 0; j < n; j++) {
printf("%d ", c[i][j]);
```

```
}
printf("\n");
}
}
```

ii. Write a C program on Multiplication two matrices?

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main(){
int a[10][10],b[10][10],c[10][10],r,c,i,j,k;
printf("Enter the number of rows and columns of the matrix: ");
scanf("%d%d", &r, &c);
printf("enter the first matrix elements\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("enter the second matrix element\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
scanf("%d",&b[i][j]);
}
}
printf("multiply of the matrix=\n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
```

```
{
c[i][j]=0;
for(k=0;k<c;k++)
{
c[i][j]+=a[i][k]*b[k][j];
}
}
}
Printf("Matrix multiplication is \n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d\t",mul[i][j]);
}
printf("\n");
}
}
```

iii. Write a C program on Sort array elements using bubble sort.

```
#include <stdio.h>
void main() {
int i,j,n,temp,a[10];
printf("Enter array size");
scanf("%d",&n);
printf("Enter elements into an array\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
for (i = 0; i < n - 1; i++)
{
for (j = 0; j < n - i - 1; j++)
{
if (a[j] > a[j + 1])
```

```
{
temp  = a[j];
a[j] = a[j + 1];
a[j + 1] = temp;
}
}
}
printf("Sorted  array:  ");
for (int i = 0; i < n; i++)
{
printf("%d ", a[i]);
}
}
```

iv. Write a C program to concatenate two strings without built-in functions?

```
#include <stdio.h>
void main()
{
char s1[20],s2[20];
int i,j;
printf("Enter the first string");
scanf("%s",s1);
printf("\n         Enter the second string");
scanf("%s",s2);
for(i=0;s1[i]!='\0';i++);
for(j=0;s2[j]!='\0';j++,i++)
{
s1[i]=s2[j];
}
}
s1[i]='\0';
printf("After concatenation, the string would look like: %s", s1);
}
```

v. Write a C program to Reverse a string using built-in and without built-in string functions?

```
#include <stdio.h>
#include <string.h>
void main()
{
char s1[50],s2[50],s3[50];
int i,j,count=0;
printf (" \n Enter a string to be reversed:");
scanf ("%s",s1);
strcpy(s2,s1);
// use strrev() function to reverse a string
printf (" \n After the reverse of a string: %s ", strrev(s1));
// without using strrev() function to reverse a string
for(i=0;s2[i]!='\0';i++)
{
count++;
}
count--;
for(i=count,j=0;i>=0;i--,j++)
{
s3[j]=s2[i];
}
s3[j]='\0';
printf("%s",s3);
}
```

WEEK-9:

i. Write a C program to find the sum of a 1D array using malloc()?

```
#include <stdio.h>
#include <stdlib.h>
int main() {
int n, *p, sum = 0;
printf("Enter the number of elements in the array: ");
scanf("%d", &n);
p=(int*)malloc(n * sizeof(int));
printf("Enter elements:  \n");
for (int i=0;i<n; i++)
{
scanf("%d", p+i);
sum += *(p+i);
}
printf("Sum of the elements: %d\n", sum)
// Free dynamically allocated memory
free(p);
return 0;
}
```

ii. Write a C program to find the total and average of n students using structures:

```
#include <stdio.h>
struct student {
int sno;
char sname [50];
int marks[5],total;
float avg;
}s1[100];
int main() {
int n,i;
printf("Enter the number of students: ");
scanf("%d", &n);
```

```
printf(“Enter Student details\n”);
for (i = 0; i < n; i++)
{
printf("Student %d\n", i+1);
scanf("%d %s",&s1[i].sno,s1[i].name):
printf("Enter Marks: ");
s1[i].total=0;
for(j=0;j<5;j++)
{
scanf("%d", &s1[i].marks[j]);
s1[i].total += s1[i].total+s1[i].marks[j];
}
s1[i].average=(float)s1[i].total/5.0;
}
printf(“Student details are\n”);
for (i = 0; i < n; i++)
{
printf("Student %d\n", i+1);
printf("Sno=%d\nSname=%s\ntotal= %d\navg=%f",s1[i].sno,s1[i].name,s1[i].total,s1[i],avg):
return 0;
}
```

iii. Write a C program to implement realloc():

```
#include <stdio.h>
#include <stdlib.h>
int main()
int *p;
int initialSize=5; // Initial size of the array
int newsize =10; // New size to reallocate
// Allocate memory for the initial array
p=(int *)malloc(initialSize*(sizeof(int));
// Initialize the array
for (int i=0; i<initialSize; i++)
```

```
{
scanf(“%d”,p+i);
}
printf("Original array elements: ");
for (int i=0; i< initialSizes; i++){
printf("%d", *(p+i));
}
printf("\n");
// reallocate memory to expand the array
p = realloc(p, newSize * sizeof(int));
// Initialize the new elemente
for (int i=initialSize; i<newSize; i++){
scanf(“%d”,p+i);
}
printf("Expanded array elements: ");
for (int i=0: i<newSize; i++)
printf("%d",*(p+i));
}
printf("\n");

// Free dynamically allocated memory
free(arr);

return 0;
}
```

WEEK -10:

i. Create and display a singly linked list using self-referential structure.

```
#include <stdio.h>
#include <stdlib.h>
struct node {
  int data;
  struct node *next;
};
int main() {
  // Create three nodes
  struct node *head = NULL;
  struct node *second = NULL;
  struct node *third = NULL;
  head = (struct node*)malloc(sizeof(struct node));
  second = (struct node*)malloc(sizeof(struct node));
  third = (struct node*)malloc(sizeof(struct node));
  // Assign data to each node
  head->data = 1;
  second->data = 2;
  third->data = 3;
  // Link the nodes together
  head->next = second;
  second->next = third;
  third->next = NULL;
  // Traverse the linked list and display its contents
  struct node *current = head;
  while (current != NULL) {
    printf("%d ", current->data);
    current = current->next;
  }
  return 0;
} //Output: 1 2 3
```

ii. Demonstrate the Differences Between Structures and Unions:

```
#include <stdio.h>
struct ExampleStruct{
int x
char y;
float z;
};
union ExampleUnion {
int x;
char y;
float z;
int main() {
struct Examplestruct s;
union ExampleUnion u;
printf("Size of struct: %zu\n", sizeof(s));
printf("Size of union: %zu\n", sizeof(u));
return 0;
}
```

iii. Write a C Program to Copy One Structure Variable to Another:

```
#include <stdio.h>
struct Student {
char name [50];
int roll number;
int main() {
struct Student student1, student2;
// Initialize student1
printf("Enter studentl's name: ");
scanf("%s", student1.name);
printf("Enter studentl's roll number: ");
scanf("%d", student1.roll_number);
// Copy student1 to student2
student2 = student1;
```

```
// Display student2's details
printf("\nStudent2's Details (Copied from Student1):\n");
printf("Name: %s,\n", student2.name);
printf("Roll Number: %d\n", student2.roll_number);
return 0;
}
```

Week 11: Simple functions using call by value, solving differential equations using Eulers theorem.

i. Write a C function to calculate NCR value.

```
#include <stdio.h>
int fact(int  z);
void main()
{
   int n, r, ncr;
   printf("\n Enter the value for N and R \n");
   scanf("%d%d", &n, &r);
   ncr = fact(n) / (fact(r) * fact(n - r));
   printf("\n The value of ncr is: %d", ncr);
}
int fact(int z)
{
   int f = 1, i;
   if (z == 0)
   {
      return(f);
   }
   else
   {
      for (i = 1; i <= z; i++)
          {
          f = f * i;
          }
   }
   return(f);
}
```

ii. Write a C function to find the length of a string.

```
#include <stdio.h>
int length(char s[])
{
int i;
for(i=0;s[i]!='\0';i++);
return i;
}
void main()
{
char s[20],n;
printf("Enter string");
scanf("%s",s);
n=length(s);
printf("Length=%d",n);
}
```

iii. Write a C function to transpose of a matrix.

```
#include <stdio.h>
void transpose(int matrix[10][10],int m ,int n)
{
int transpose[10][10],i,j;
  for (i = 0;i < m;i++)
    for (j = 0; j < n; j++)
      transpose[j][i] = matrix[i][j];
  printf("Transpose of the matrix:");
  for (i = 0; i< n; i++) {
    for (j = 0; j < m; j++)
      printf("%d\t", transpose[i][j]);
    printf("\n");
  }
```

```
}
int main(){
   int m, n, i, j, matrix[10][10];
   printf("Enter rows and columns :");
   scanf("%d%d", &m, &n);
   printf("Enter elements of the matrix");
   for (i= 0; i < m; i++)
      for (j = 0; j < n; j++)
         scanf("%d", &matrix[i][j]);
transpose(matrix,m,n);
   return 0;
}
```

Week 12: Recursive functions

i. Write a recursive function to generate Fibonacci series.

```
#include <stdio.h>
int fibonacciSeries(i){
   if (i <= 1)
      return i;
   else
      return (fibonacciSeries(i - 1) + fibonacciSeries(i - 2));
}
int main(void) {

   int num = 10, i;
   for(i = 0; i < num; i ++){
      printf("%d ", fibonacciSeries(i));
   }
   return 0;
}
```

ii. Write a recursive function to find the LCM of two numbers.

```
#include <stdio.h>

int lcm(int n1, int n2) {
   static int lowestcm = 1;
   if(lowestcm%n1 == 0 && lowestcm%n2 == 0)
   {
      return lowestcm;
   }
   else
   {
      lowestcm++;
      lcm(n1,n2);
      return lowestcm;
```

```
    }
}
int main() {
    int n1 = 4, n2 = 8;
    printf("L.C.M of %d and %d is %d.", n1, n2, lcm(n1, n2));
    return 0;
}
```

iii. Write a recursive function to find the factorial of a number.

```
#include<stdio.h>
int fact(int);
int main()
{
    int x,n;
    printf(" Enter the Number to Find Factorial :");
    scanf("%d",&n);

    x=fact(n);
    printf(" Factorial of %d is %d",n,x);

    return 0;
}
int fact(int n)
{
    if(n==0)
        return(1);
    return(n*fact(n-1));
}
```

iv. Write a recursive function to find the sum of series.

```
#include<stdio.h>
int series(int n);
```

```
int main( )
{
        int n;
        printf("Enter number of terms : ");
        scanf("%d", &n);
        printf("\b\b Using Recursion :: \n");
        printf("\b\b = %d\n", series(n));
        printf("\n\b\b Using Recursion :: \n");
        return 0;
}
int series(int n)
{
        int i, sum=0;
        for(i=1; i<=n; i++)
        {
                printf("%d + ", i);
                sum+=i;
        }
        return sum;
}
```